

PP/018/V/R2



**LABORATORIUM
S1 INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS AHMAD DAHLAN**

PETUNJUK PRAKTIKUM EDISI KURIKULUM OBE

PEMROGRAMAN BERORIENTASI OBJEK



Penyusun:

Herman Yuliansyah, S.T., M.Eng., Ph.D.

Ali Tarmuji, S.T., M.Cs.

Faisal Fajri Rahani, S.Si., M.Cs.

Supriyanto, S.T., M.T.

2023

HAK CIPTA

PETUNJUK PRAKTIKUM NAMA PRAKTIKUM

Copyright© 2023,

Herman Yuliansyah, S.T., M.Eng., Ph.D.

Ali Tarmuji, S.T., M.Cs.

Faisal Fajri Rahani, S.Si., M.Cs.

Supriyanto, S.T., M.T.

Hak Cipta dilindungi Undang-Undang

Dilarang mengutip, memperbanyak atau mengedarkan isi buku ini, baik sebagian maupun seluruhnya, dalam bentuk apapun, tanpa izin tertulis dari pemilik hak cipta dan penerbit.

Diterbitkan oleh:**Program Studi Informatika**

Fakultas Teknologi Industri

Universitas Ahmad Dahlan

Jalan Ring Road Selatan, Tamanan, Banguntapan, Bantul Yogyakarta 55166

Penulis : Herman Yuliansyah, S.T., M.Eng., Ph.D.
Ali Tarmuji, S.T., M.Cs.
Faisal Fajri Rahani, S.Si., M.Cs.
Supriyanto, S.T., M.T.

Editor : Laboratorium S1 Informatika, Universitas Ahmad Dahlan
Desain sampul : Laboratorium S1 Informatika, Universitas Ahmad Dahlan
Tata letak : Laboratorium S1 Informatika, Universitas Ahmad Dahlan

Ukuran/Halaman : 21 x 29,7 cm / 84 halaman

Didistribusikan oleh:**Laboratorium S1 Informatika**

Universitas Ahmad Dahlan

Jalan Ring Road Selatan, Tamanan, Banguntapan, Bantul Yogyakarta 55166
Indonesia

KATA PENGANTAR

Puji syukur kami panjatkan kehadiran Tuhan Yang Maha Esa yang telah memberikan rahmat dan karunia-Nya, sehingga kami dapat menyelesaikan pembuatan petunjuk praktikum ini. Petunjuk praktikum ini dibuat untuk membantu praktikan dalam praktikum mata kuliah Pemrograman Berorientasi Objek. Petunjuk praktikum ini telah disusun dalam dua belas pertemuan dan disesuaikan dengan Rencana Pembelajaran Semester mata kuliah Pemrograman Berorientasi Objek. Hal ini bertujuan agar praktikan dapat mempelajari isi petunjuk praktikum dengan baik.

Dengan selesainya petunjuk praktikum ini, maka kami tidak lupa mengucapkan banyak terima kasih. Kami juga menyampaikan terima kasih kepada semua pihak yang terlibat dalam penyusunan petunjuk praktikum Pemrograman Berorientasi Objek ini.

Demikian petunjuk praktikum Pemrograman Berorientasi Objek yang telah kami buat. Kami mohon kritik dan sarannya apabila terdapat kekurangan dalam penyusunan petunjuk Praktikum ini. Semoga petunjuk Praktikum Pemrograman Berorientasi Objek ini dapat bermanfaat.

Yogyakarta, 20 September 2023

Penyusun

DAFTAR PENYUSUN

Herman Yuliansyah, S.T., M.Eng., Ph.D.



Herman Yuliansyah adalah seorang dosen berjabatan akademik Lektor di program studi Informatika Universitas Ahmad Dahlan. Herman yuliansyah telah menamatkan studi Sarjana Teknik (ST) di Teknik Elektro Universitas Muhammadiyah Yogyakarta, Master of Engineering (M.Eng) di Fakultas Teknik Universitas Gajah Mada, dan Doctor of Philofophy (Ph.D) di Faculty of Information Science & Technology Universiti Kebangsaan Malaysia. Pada petunjuk praktikum ini, Herman Yuliansyah menulis untuk pertemuan 4 sampai 6.

Ali Tarmuji, S.T., M.Cs.



Ali Tarmuji adalah seorang dosen berjabatan akademik Lektor di program studi Informatika Universitas Ahmad Dahlan. Ali Tarmuji telah menamatkan studi Sarjana Teknik (ST) di Teknik Informatika Universitas Ahmad Dahlan dan Master of Computer Science (M.Cs) di Fakultas Ilmu Komputer Universitas Gajah Mada. Pada petunjuk praktikum ini, Ali Tarmuji menulis untuk pertemuan 1 sampai 3.

Faisal Fajri Rahani, S.Si., M.Cs.



Faisal Fajri Rahani adalah seorang dosen berjabatan akademik Lektor di program studi Informatika Universitas Ahmad Dahlan. Faisal Fajri Rahani telah menamatkan studi Sarjana Teknik (ST) di Elektronika dan Instrumentasi Universitas Gajah Mada dan Master of Computer Science (M.Cs) di Fakultas Ilmu Komputer Universitas Gajah Mada. Pada petunjuk praktikum ini, Faisal Fajri Rahani menulis untuk pertemuan 9 sampai 10.

Supriyanto, S.T., M.T.



Supriyanto adalah seorang dosen berjabatan akademik Lektor di program studi Informatika Universitas Ahmad Dahlan. Supriyanto telah menamatkan studi Sarjana Teknik (ST) di Teknik Informatika Universitas Ahmad Dahlan dan Magister Teknik (M.T) di Teknik Elektro Institut Teknologi Bandung. Pada petunjuk praktikum ini, Supriyanto menulis untuk pertemuan 9 sampai 10.

KONTRIBUSI PENULIS

Nomor Pertemuan	Daftar Penulis
Pertemuan 1	Ali Tarmuji, S.T., M.Cs.
Pertemuan 2	Ali Tarmuji, S.T., M.Cs.
Pertemuan 3	Ali Tarmuji, S.T., M.Cs.
Pertemuan 4	Herman Yuliansyah, S.T., M.Eng., Ph.D.
Pertemuan 5	Herman Yuliansyah, S.T., M.Eng., Ph.D.
Pertemuan 6	Herman Yuliansyah, S.T., M.Eng., Ph.D.
Pertemuan 7	Supriyanto, S.T., M.T.
Pertemuan 8	Supriyanto, S.T., M.T.
Pertemuan 9	Faisal Fajri Rahani, S.Si., M.Cs.
Pertemuan 10	Faisal Fajri Rahani, S.Si., M.Cs.

HALAMAN REVISI

Yang bertanda tangan di bawah ini:

Nama : Herman Yuliansyah, S.T., M.Eng., Ph.D.

NIP/NIY : 60110648

Jabatan : Dosen Pengampu Mata Kuliah **Pemrograman Berorientasi Objek**

Dengan ini menyatakan pelaksanaan Revisi Petunjuk Praktikum **Pemrograman Berorientasi Objek** untuk Program Studi Informatika telah dilaksanakan dengan penjelasan sebagai berikut:

No	Keterangan Revisi	Tanggal Revisi	Nomor Modul
1	<ul style="list-style-type: none"> a. Mengubah susunan materi praktikum yang sebelumnya 12 pertemuan menjadi 10 pertemuan. b. Materi pertama merupakan gabungan dari materi 1 dan 2 pada modul edisi sebelumnya. Yaitu pengenalan java dan IDE dilengkapi dengan dasar pemrograman. a. Materi ke 11 pada modul edisi sebelumnya dihilangkan. 	22 Agustus 2019	PP/018/V/R1
2	<ul style="list-style-type: none"> a. Penyesuaian aktifitas praktikum dengan Rencana Pembelajaran Kuliah Pemrograman Berorientasi Objek berbasis versi OBE a. Penambahan Pretest dan Postest pada setiap pertemuan praktikum 	17 September 2021	PP/018/III/R2
3	<ul style="list-style-type: none"> a. Perubahan pada soal PreTest dan PostTest beserta komposisi bobot nilai b. Perubahan materi praktikum pada pertemuan ke 10 	20 September 2022	
4	<ul style="list-style-type: none"> a. Perubahan template buku petunjuk praktikum b. Perubahan contoh kasus dalam pertemuan 	20 September 2023	

Yogyakarta, 20 September 2023

Penyusun



Herman Yuliansyah, S.T., M.Eng., Ph.D.

NIPM : 19830712 201104 111 0896292

HALAMAN PERNYATAAN

Yang bertanda tangan di bawah ini:

Nama : Murein Miksa Mardhia S.T., M.T.

NIPM : 19891019 201606 011 1236278

Jabatan : Kepala Laboratorium S1 Informatika

Menerangkan dengan sesungguhnya bahwa Petunjuk Praktikum ini telah direview dan akan digunakan untuk pelaksanaan praktikum di Semester Gasal Tahun Akademik 2023/2024 di Laboratorium Praktikum S1 Informatika, Program Studi Informatika, Fakultas Teknologi Industri, Universitas Ahmad Dahlan.

Yogyakarta, 20 September 2023

Mengetahui,
Ketua Kelompok Keilmuan Sistem Cerdas



Herman Yuliansyah, S.T., M.Eng., Ph.D.
NIP/NIY. 19830712 201104 111 0896292

Kepala Laboratorium Praktikum S1
Informatika



Murein Miksa Mardhia, S.T., M.T.
NIPM : 19891019 201606 011 1236278

VISI DAN MISI PRODI INFORMATIKA

VISI

Menjadi program studi yang unggul dan inovatif dalam bidang rekayasa perangkat lunak dan sistem cerdas dengan dijiwai nilai-nilai Islam

MISI

1. Mengimplementasikan nilai-nilai AIK pada semua aspek kegiatan.
2. Memajukan ilmu pengetahuan dan teknologi Rekayasa Perangkat Lunak dan Sistem cerdas melalui pendidikan, penelitian, dan pengabdian kepada masyarakat.
3. Mengembangkan kerjasama dalam pendidikan, penelitian, dan pengabdian kepada masyarakat di tingkat lokal, nasional, maupun internasional.
4. Menyelenggarakan tata kelola program studi yang unggul dan inovatif.
5. Berperan aktif dalam kegiatan yang menunjang profesi dosen.

TATA TERTIB LABORATORIUM S1 INFORMATIKA

DOSEN/KOORDINATOR PRAKTIKUM

1. Dosen harus hadir saat praktikum minimal 15 menit di awal kegiatan praktikum untuk mengisi materi dan menandatangani presensi kehadiran praktikum.
2. Dosen membuat modul praktikum, soal seleksi asisten, pre-test, post-test, dan responsi dengan berkoordinasi dengan asisten dan pengampu mata praktikum.
3. Dosen berkoordinasi dengan koordinator asisten praktikum untuk evaluasi praktikum setiap minggu.
4. Dosen menandatangani surat kontrak asisten praktikum dan koordinator asisten praktikum.
5. Dosen yang tidak hadir pada slot praktikum tertentu tanpa pemberitahuan selama 2 minggu berturut-turut mendapat teguran dari Kepala Laboratorium, apabila masih berlanjut 2 minggu berikutnya maka Kepala Laboratorium berhak mengganti koordinator praktikum pada slot tersebut.

PRAKTIKAN

1. Praktikan harus hadir 15 menit sebelum kegiatan praktikum dimulai, dan dispensasi terlambat 15 menit dengan alasan yang jelas (kecuali asisten menentukan lain dan patokan jam adalah jam yang ada di Laboratorium, terlambat lebih dari 15 menit tidak boleh masuk praktikum & dianggap INHAL).
2. Praktikan yang tidak mengikuti praktikum dengan alasan apapun, wajib mengikuti INHAL, maksimal 4 kali praktikum dan jika lebih dari 4 kali maka praktikum dianggap GAGAL.
3. Praktikan yang akan mengikuti inhal diwajibkan mendaftarkan diri dan membayar administrasi inhal kepada laboran inhal paling lambat H-1 jadwal inhal.
4. Praktikan harus berpakaian rapi sesuai dengan ketentuan Universitas, sebagai berikut:
 - a. Tidak boleh memakai Kaos Oblong, termasuk bila ditutupi Jaket/Jas Almamater (Laki-laki / Perempuan) dan Topi harus Dilepas.
 - b. Tidak Boleh memakai Baju ketat, Jilbab Minim dan rambut harus tertutup jilbab secara sempurna, tidak boleh kelihatan di jidat maupun di punggung (khusus Perempuan).
 - c. Tidak boleh memakai baju minim, saat duduk pun pinggang harus tertutup rapat (Laki-laki / Perempuan).
 - d. Laki-laki tidak boleh memakai gelang, anting-anting ataupun aksesoris Perempuan.
5. Praktikan tidak boleh makan dan minum selama kegiatan praktikum berlangsung, harus menjaga kebersihan, keamanan dan ketertiban selama mengikuti kegiatan praktikum atau selama berada di dalam laboratorium (tidak boleh membuang sampah sembarangan baik kertas, potongan kertas, bungkus permen baik di lantai karpet maupun di dalam ruang CPU).
6. Praktikan dilarang meninggalkan kegiatan praktikum tanpa seizin Asisten atau Laboran.
7. Praktikan harus meletakkan sepatu dan tas pada rak/loker yang telah disediakan.
8. Selama praktikum dilarang *NGENET/NGE-GAME*, kecuali mata praktikum yang membutuhkan atau menggunakan fasilitas Internet.
9. Praktikan dilarang melepas kabel jaringan atau kabel power praktikum tanpa sepengetahuan laboran
10. Praktikan harus memiliki FILE Petunjuk praktikum dan digunakan pada saat praktikum dan harus siap sebelum praktikum berlangsung.
11. Praktikan dilarang melakukan kecurangan seperti mencontek atau menyalin pekerjaan praktikan yang lain saat praktikum berlangsung atau post-test yang menjadi tugas praktikum.

12. Praktikan dilarang mengubah *setting software/hardware* komputer baik menambah atau mengurangi tanpa permintaan asisten atau laboran dan melakukan sesuatu yang dapat merugikan laboratorium atau praktikum lain.
13. Asisten, Koordinator Praktikum, Kepala laboratorium dan Laboran mempunyai hak untuk menegur, memperingatkan bahkan meminta praktikan keluar ruang praktikum apabila dirasa anda mengganggu praktikan lain atau tidak melaksanakan kegiatan praktikum sebagaimana mestinya dan atau tidak mematuhi aturan lab yang berlaku.
14. Pelanggaran terhadap salah satu atau lebih dari aturan diatas maka Nilai praktikum pada pertemuan tersebut dianggap 0 (NOL) dengan status INHAL.

ASISTEN PRAKTIKUM

1. Asisten harus hadir 15 Menit sebelum praktikum dimulai (konfirmasi ke koordinator bila mengalami keterlambatan atau berhalangan hadir).
2. Asisten yang tidak bisa hadir WAJIB mencari pengganti, dan melaporkan kepada Koordinator Asisten.
3. Asisten harus berpakaian rapi sesuai dengan ketentuan Universitas, sebagai berikut:
 - a. Tidak boleh memakai Kaos Oblong, termasuk bila ditutupi Jaket/Jas Almamater (Laki-laki / Perempuan) dan Topi harus Dilepas.
 - b. Tidak Boleh memakai Baju ketat, Jilbab Minim dan rambut harus tertutup jilbab secara sempurna, tidak boleh kelihatan di jidat maupun di punggung (khusus Perempuan).
 - c. Tidak boleh memakai baju minim, saat duduk pun pinggang harus tertutup rapat (Laki-laki / Perempuan).
 - d. Laki-laki tidak boleh memakai gelang, anting-anting ataupun aksesoris Perempuan.
4. Asisten harus menjaga kebersihan, keamanan dan ketertiban selama mengikuti kegiatan praktikum atau selama berada di laboratorium, menegur atau mengingatkan jika ada praktikan yang tidak dapat menjaga kebersihan, ketertiban atau kesopanan.
5. Asisten harus dapat merapikan dan mengamankan presensi praktikum, Kartu Nilai serta tertib dalam memasukan/Input nilai secara Online/Offline.
6. Asisten mencatat dan merekap praktikan dengan status INHAL setiap minggu serta wajib mengumumkan mekanisme INHAL di awal pertemuan praktikum.
7. Asisten harus dapat bertindak secara profesional sebagai seorang asisten praktikum dan dapat menjadi teladan bagi praktikan.
8. Asisten harus dapat memberikan penjelasan/pemahaman yang dibutuhkan oleh praktikan berkenaan dengan materi praktikum yang diasistensi sehingga praktikan dapat melaksanakan dan mengerjakan tugas praktikum dengan baik dan jelas.
9. Asisten tidak diperkenankan mengobrol sendiri apalagi sampai membuat gaduh.
10. Asisten dimohon mengkoordinasikan untuk meminta praktikan agar mematikan komputer untuk jadwal terakhir dan sudah dilakukan penilaian terhadap hasil kerja praktikan.
11. Asisten wajib untuk mematikan LCD Projector dan komputer asisten/praktikan apabila tidak digunakan.
12. Asisten tidak diperkenankan menggunakan akses internet selain untuk kegiatan praktikum, seperti Youtube/Game/Medsos/Streaming Film di komputer praktikan.

LAIN-LAIN

1. Pada Saat Responsi Harus menggunakan Baju Kemeja untuk Laki-laki dan Perempuan untuk Praktikan dan Asisten.
2. Ketidakhadiran praktikum dengan alasan apapun dianggap INHAL.
3. Pelaksanaan (waktu dan metode) INHAL sama seperti praktikum mingguan/reguler.

4. Izin praktikum mengikuti aturan izin SIMERU/KULIAH.
5. Yang tidak berkepentingan dengan praktikum dilarang mengganggu praktikan atau membuat keributan/kegaduhan.
6. Penggunaan lab diluar jam praktikum maksimal sampai pukul 21.00 dengan menunjukkan surat ijin dari Kepala Laboratorium Prodi Informatika.

Yogyakarta, 20 September 2023

Kepala Laboratorium Praktikum S1
Informatika



Murein Miksa Mardhia, S.T., M.T.
NIPM : 19891019 201606 011 1236278

DAFTAR ISI

HAK CIPTA	1
KATA PENGANTAR.....	2
DAFTAR PENYUSUN.....	3
KONTRIBUSI PENULIS	4
HALAMAN REVISI.....	5
HALAMAN PERNYATAAN.....	6
VISI DAN MISI PRODI INFORMATIKA.....	7
TATA TERTIB LABORATORIUM S1 INFORMATIKA	8
DAFTAR ISI.....	11
SKENARIO PRAKTIKUM SECARA LURING.....	12
PRAKTIKUM 1: TEKNOLOGI DAN DASAR PEMROGRAMAN BERORIENTASI OBJEK.....	13
PRAKTIKUM 2: KOMPONEN DASAR PEMROGRAMAN BERORIENTASI OBJEK	25
PRAKTIKUM 3: KONSTRUKTOR DAN ENKAPSULASI	30
PRAKTIKUM 4: PEWARISAN	36
PRAKTIKUM 5: POLYMORPHISME.....	42
PRAKTIKUM 6: RELASI ANTAR CLASS.....	47
PRAKTIKUM 7: CLASS ABSTRAK	56
PRAKTIKUM 8: INTERFACE	62
PRAKTIKUM 9: EXCEPTION HANDLING	67
PRAKTIKUM 10: APLIKASI PEMROGRAMAN BERORIENTASI OBJEK.....	74
DAFTAR PUSTAKA.....	82

SKENARIO PRAKTIKUM SECARA LURING

Nama Mata Praktikum : Pemrograman Berorientasi Objek

Jumlah Pertemuan : 10 praktikum + 1 Responsi

TABEL SKENARIO PRAKTIKUM LURING

Pertemuan ke	Judul Materi	Waktu (Lama praktikum sampai pengumpulan posttest)	Skenario Praktikum <i>(Dari pemberian pre-test, post-test dan pengumpulannya serta mencantumkan metode yang digunakan misal video, whatsapp group, Google meet atau lainnya)</i>
1	Teknologi Java dan Dasar Pemrograman Java	90 menit	<ul style="list-style-type: none"> ● Pretest ● Pelaksanaan Praktikum Posttest
2	Komponen Dasar PBO	90 menit	<ul style="list-style-type: none"> ● Pretest ● Pelaksanaan Praktikum Posttest
3	Konstruktor dan Enkapsulasi	90 menit	<ul style="list-style-type: none"> ● Pretest ● Pelaksanaan Praktikum Posttest
4	Pewarisan	90 menit	<ul style="list-style-type: none"> ● Pretest ● Pelaksanaan Praktikum ● Posttest
5	Polymorphisme	90 menit	<ul style="list-style-type: none"> ● Pretest ● Pelaksanaan Praktikum ● Posttest
6	Relasi Class	90 menit	<ul style="list-style-type: none"> ● Pretest ● Pelaksanaan Praktikum ● Posttest
7	Abstract	90 menit	<ul style="list-style-type: none"> ● Pretest ● Pelaksanaan Praktikum ● Posttest
8	Interface	90 menit	<ul style="list-style-type: none"> ● Pretest ● Pelaksanaan Praktikum ● Posttest
9	Exception Handling	90 menit	<ul style="list-style-type: none"> ● Pretest ● Pelaksanaan Praktikum ● Posttest
10	Aplikasi PBO	90 menit	<ul style="list-style-type: none"> ● Pretest ● Pelaksanaan Praktikum ● Posttest
11	Responsi	90 menit	<ul style="list-style-type: none"> ● Pengerjaan soal Responsi

PRAKTIKUM 1: TEKNOLOGI DAN DASAR PEMROGRAMAN BERORIENTASI OBJEK

Pertemuan ke : 1

Total Alokasi Waktu : 90 menit

- Materi : 15 menit
- Pre-Test : 15 menit
- Praktikum : 40 menit
- Post-Test : 20 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 20 %
- Post-Test : 60 %

Pemenuhan CPL dan CPMK:

CPL-03	Mampu menerapkan konsep teoritis bidang area Informatika terkait matematika dasar dan ilmu komputer untuk memodelkan masalah dan meningkatkan produktivitas
CPMK-01	Mahasiswa mampu memahami konsep rancangan dan prinsip abstraction dan program decomposition

1.1 DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu:

1. Menjelaskan Teknologi yang berkaitan dengan implementasi pemrograman berorientasi objek dan cara kerja compiler
2. Menjelaskan kembali Percabangan, perulangan, array

1.2 INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL-03	CPMK-01	<p>mahasiswa dapat :</p> <ol style="list-style-type: none"> 1) Memahami environment development yang digunakan 2) Memahami cara kerja compiler 3) Menerapkan percabangan dan perulangan sesuai bahasa pemrograman yang digunakan 4) Membedakan penggunaan percabangan dan perulangan 5) Mengimplementasikan array
--------	---------	--

1.3 TEORI PENDUKUNG

Sebagai sebuah bahasa pemrograman, Java dapat membuat seluruh bentuk aplikasi, *desktop*, *web* dan lainnya, sebagaimana dibuat dengan menggunakan bahasa pemrograman konvensional yang lain. Java adalah bahasa pemrograman yang berorientasi objek (OOP) dan dapat dijalankan pada berbagai platform sistem operasi. Perkembangan Java tidak hanya terfokus pada satu sistem operasi, tetapi dikembangkan untuk berbagai sistem operasi dan bersifat open source.

Bahasa pemrograman Java pada awalnya dibuat oleh James Gosling pada tahun 1995 sebagai bagian dari Sun Microsystems Java Platform. Sintaks Java banyak diturunkan dari C dan C++ tetapi lebih sederhana, ketat dan mempunyai akses ke OS yang lebih terbatas. Hal ini karena Java ditujukan sebagai bahasa pemrograman yang cukup sederhana untuk dipelajari dan mudah dibaca.

Aplikasi Java ditulis sebagai file berekstensi .java yang di compile menjadi file .class. File .Class ini adalah bytecode yang bisa dijalankan di semua Java Virtual Machine, tidak peduli apapun OS-nya ataupun arsitektur processornya. Java adalah bahasa yang ditujukan untuk semua kebutuhan, concurrent, berbasis class, object oriented serta didesain agar tidak tergantung terhadap lingkungan dimana aplikasi dijalankan (OS dan processor).

Java Platform terdiri dari tiga buah profile: Java ME (Java Micro Edition) adalah java yang bisa berjalan di dalam embedded system seperti Java Card dan Handphone. Java SE (Java Standard Edition) adalah java yang bisa berjalan di dalam PC maupun server sebagai aplikasi standalone maupun aplikasi desktop. Java EE (Java Enterprise Edition) adalah profile java yang ditujukan untuk membuat aplikasi Enterprise seperti Web Application (Servlet) dan Enterprise Java Bean (EJB).

Sebelum mengenal yang namanya kelas, terlebih dahulu belajar tentang tipe data, variabel, operator, statement kontrol pada bahasa pemrograman java. Yang pertama kita harus kenal tipe data dan cara menggunakan variabel didalam lingkungan java. Berikut tipe data yang digunakan dalam Java.

Tipe Data

Terdapat beberapa tipe data primitive yang ada di Java yaitu :

Tipe Data	Keterangan
boolean	true atau false
char	Karakter
byte	-128 - 127
short	-32768 - 32767
int	-2147483648 - 2147483647
long	-9223372036854775808 - 9223372036854775807
double	4.9E-324 - 1.7976931348623157E308
float	1.4E-45 - 3.4028235E38

String bukanlah merupakan tipe data di Java, String merupakan Object. Namun string memiliki keunikan yaitu String dapat langsung dibuat tanpa harus membuat Object.

Variabel

Variabel merupakan sesuatu yang digunakan untuk menampung sebuah data. Sebuah variabel harus ada dalam sebuah kelas atau metode. Pembuatan sebuah variabel di Java terlihat pada kode dibawah ini.

```
Tipe namaVariabel; // mendeklarasikan variabel
Tipe namaVariabel= nilai; //inisialisasi variabel
```

Contoh 1.1 Pendeklarasian variabel:

```
Int nilai;
```

```
Int nilai=0;
```

STRUKUR PERCABANGAN PADA JAVA

Percabangan IF...ESLE

Sintaks if	Contohnya:
<pre>if (boolean expression) { statement or block } else if (boolean expression) { statement or block } else { statement or block }</pre>	<pre>public Class Percabangan { public static void main(String[] args) { int a = 2; if (a < 5) { System.out.println("Nilai a lebih kecil dari 5"); } else { System.out.println("Nilai a lebih besar dari 5"); } } }</pre>

Percabangan SWITCH

Sintaks switch	Contohnya:
<p>Syntax switch sebagai berikut :</p> <pre>switch (expression) { case constant1 : statements; break; case constant2 : statements; break; default : statements; break; }</pre>	<pre>switch(x){ case 1: System.out.println("Senin"); break; case 2: System.out.println("Selasa"); break; case 3: System.out.println("Rabu"); break; case 4: System.out.println("Kamis"); break; case 5: System.out.println("Jum'at"); break; case 6: System.out.println("Sabtu"); break; case 7: System.out.println("Minggu"); break; default: System.out.println("Salah input hari"); break; }</pre>

Percabangan dengan Ekspresi bersyarat

Digunakan untuk menggantikan sebuah bentuk if else. Sintaks adalah sebagai berikut :

Sintaks ekspresi bersyarat	Contoh :
<pre>exp1 ? exp2 : exp3</pre> <p><i>Dalam hal ini, jika exp1 adalah benar, exp2 dieksekusi. Jika exp2 adalah salah, exp3 dieksekusi.</i></p>	<pre>int a = 50; int b = 100; int nilai = 3 > 2 ? a : b; Outputnya : nilai = 50</pre>

PERULANGAN PADA JAVA

Perulangan for

Syntax for	Contoh
<pre>for (init_expr;boolean testexpr;alter_expr) { statement or block }</pre>	<pre>public Class PerulanganFOR { public static void main(String[] args) { for (int i = 1; i <=5; i++) { System.out.println(i+"=Belajar Javaitu mudah"); } } }</pre>

Perulangan while

Sintaks while	Contoh
<pre>while (boolean testexpr) { statement or block }</pre>	<pre>public Class PerulanganWHILE { public static void main(String[] args) { int i = 0; while (i<10) { System.out.println(i); i++; } } }</pre>

Perulangan do....while

Sintaks do/while	Contoh
<pre>do {</pre>	<pre>public Class PerulanganDOWHILE {</pre>

<pre>statement or block } while (boolean testexpr)</pre>	<pre>public static void main(String[] args) { int i = 0; do { System.out.println(i); i++; } while (i<10); }</pre>
--	--

ARRAY

Mendeklarasikan Array

Array biasanya digunakan untuk mengelompokkan data atau objek yang memiliki tipe yang sama. Array memungkinkan untuk mengacu ke sekumpulan objek dengan nama yang sama. Array dapat dideklarasikan dengan tipe apa saja, baik itu yang bertipe data primitif maupun objek. Berikut contoh deklarasi Array:

```
char s[];
point p[]; // point adalah sebuah kelas
```

Dalam bahasa pemrograman Java, Array merupakan sebuah objek meskipun ia terdiri dari elemen yang bertipe data primitif. Seperti halnya kelas yang lain, ketika mendeklarasikan Array belum dibentuk sebuah objek Array. Deklarasi Array hanya membuat sebuah referensi yang dapat digunakan untuk mengacu ke sebuah objek Array.

Besarnya memori yang digunakan oleh elemen-elemen Array akan dialokasikan secara dinamis dengan menggunakan pernyataan `new` atau dengan array initializer. Contoh deklarasi Array di atas, dengan menggunakan kurung siku setelah nama variabel, merupakan standar penulisan array dalam C, C++ dan Java. Format tersebut agak sulit untuk dibaca. Oleh karena itu, bahasa Java menggunakan bentuk deklarasi Array alternatif dengan menggunakan kurung siku di sebelah kiri seperti dalam contoh berikut:

```
char [] s;
point [] p; // point adalah sebuah kelas
```

Sebuah deklarasi Array dapat ditulis sebagai pendeklarasian tipe Array di bagian kiri dan nama variabel di bagian kanan. Kedua bentuk deklarasi Array di atas dapat digunakan, tetapi sebaiknya konsisten dalam menggunakan satu bentuk saja. Dalam deklarasi Array, tidak ditentukan ukuran aktual dari Array. Ketika mendeklarasikan Array dengan kurung siku yang berada di sebelah kiri nama variabel, kurung siku tersebut berlaku bagi semua variabel yang berada di sebelah kanannya.

Membuat Array

Array dibuat dengan menggunakan keyword `new`. Contoh di bawah ini membuat sebuah Array dengan tipe primitif `char`:

```
s = new char[26];
```

Indeks Array dimulai dengan angka 0. Batasan legal untuk nilai indeks ini adalah dari nol hingga jumlah elemen Array – 1. Apabila program berusaha mengakses Array di luar batas yang legal, maka akan muncul runtime exception.

Untuk membuat Array dengan elemen objek, gunakan cara penulisan berikut

```
p = new point[10];
```

Pernyataan tersebut tidak membuat 10 objek Point. Untuk membuat objek Point, gunakan pernyataan berikut :

```
p[0] = new point(0, 1);
p[1] = new point(1, 2);
```

Menginisialisasi Array

Java memiliki cara singkat untuk membuat sebuah objek Array dengan elemen-elemennya memiliki nilai awal :

```
String names[] = {"Joko", "Joni", "Jujuk"};
Kode di atas adalah sama dengan kode berikut ini:
String names[];
names[0] = "Joko";
names[1] = "Joni";
names[2] = "Jujuk";
```

Cara singkat ini dapat digunakan untuk Array dengan berbagai jenis tipe, seperti dalam contoh berikut :

```
MyDate dates[] = {
    new MyDate(22, 7, 1964),
    new MyDate(1, 1, 2000),
    new MyDate(22, 12, 1964),
};
```

Array Multidimensi

Dalam Java, dapat dibuat Array dari Array sehingga dinamai Array multidimensi.

Contoh berikut ini memperlihatkan cara membuat Array dua dimensi :

```
int twoDim [][] = new int[4][];
twoDim[0] = new int[5];
twoDim[1] = new int[5];
```

Array dari Array yang bersifat non-rectangular dapat dibuat seperti dalam contoh berikut :

```
twoDim[0] = new int[2];
twoDim[1] = new int[4];
twoDim[2] = new int[6];
twoDim[3] = new int[8];
```

Untuk membuat Array dari Array yang rectangular dengan cara mudah, gunakan bentuk berikut:

```
int twoDim[][] = new int[4][5];
```

Batasan Array

Dalam bahasa Java, indeks Array dimulai dengan angka nol. Jumlah elemen di dalam Array disimpan sebagai bagian dari objek Array di dalam atribut length. Atribut ini dapat digunakan untuk melakukan proses iterasi pada Array seperti dalam contoh berikut :

```
int list[] = new int[10];
for(int i = 0; i < list.length; i++) {
    System.out.println(list[i]);
}
```

Dengan menggunakan atribut length, pemeliharaan kode program menjadi mudah karena program tidak perlu mengetahui jumlah elemen Array pada saat kompilasi.

Manipulasi Array

Mengubah Ukuran Array

Setelah membuat objek Array, ukuran Array tersebut tidak dapat diubah. namun demikian, dapat digunakan variabel referensi yang sama untuk menunjuk ke sebuah objek Array baru seperti dalam contoh di bawah ini :

```
int myArray[] = new int[6];
myArray = new int[10];
```

Dalam kode program ini, objek Array yang pertama akan hilang kecuali ada variabel lain yang dibuat untuk merujuk ke objek Array tersebut.

Menyalin Array

Bahasa Java menyediakan method khusus untuk menyalin Array, yaitu dengan menggunakan method `arrayCopy()` dari kelas `System` seperti dalam contoh berikut :

```
int myArray[] = {1, 2, 3, 4, 5, 6};
// array engan elemen yang lebih banyak
int hold[] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1};
//menyalin semua elemen myArray ke hold
// dimulai dengan indeks ke 0
System.arraycopy(myArray, 0, hold, 0, myArray.length);
```

Setelah proses pengkopian di atas, maka isi dari Array `hold` adalah 1, 2, 3, 4, 5, 6, 4, 3, 2, 1.

1.4 HARDWARE DAN SOFTWARE

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. IntelliJ IDEA Community
3. Java SDK

1.5 PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-03	CPMK-01	Sebutkan perbedaan JDK dan JRE!	30
2.	CPL-03	CPMK-01	Jelaskan alur eksekusi program java dari source code sampai dengan aplikasi berjalan!	40
3	CPL-03	CPMK-01	Sebutkan tipe-tipe data primitive dengan jenis numeric!	30

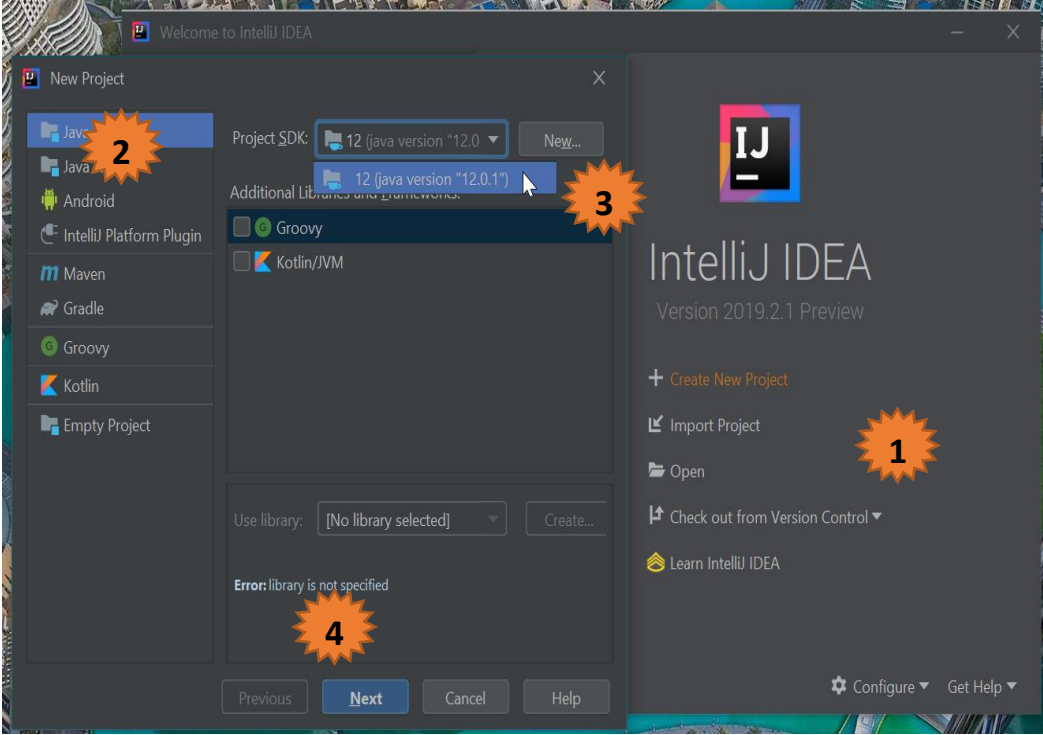
1.6 LANGKAH PRAKTIKUM

Aturan Penilaian (**Total Skor: 100**):

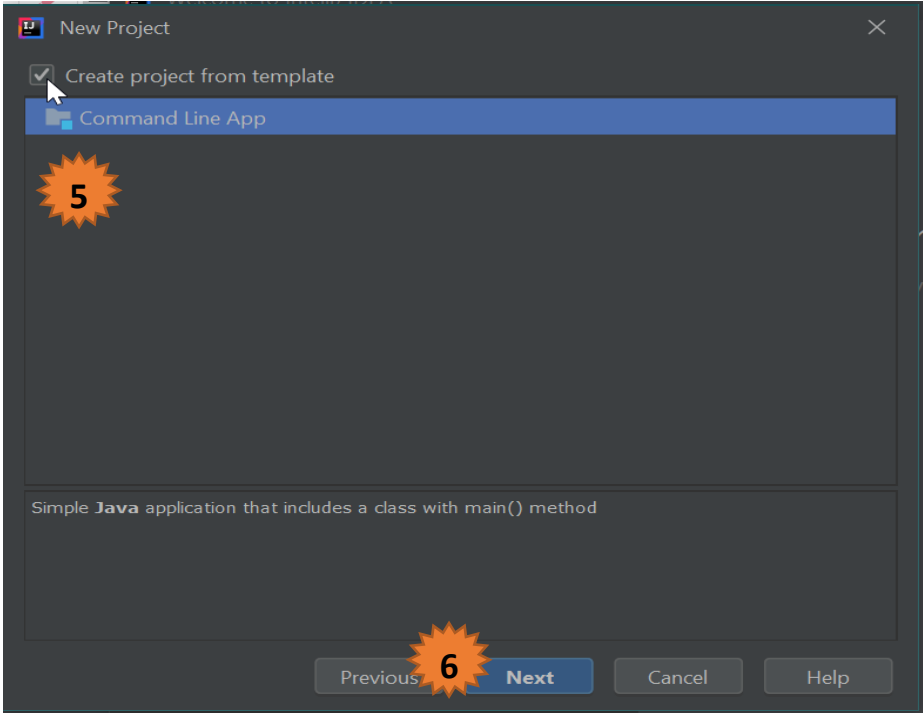
No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL-03	CPMK-01	Selesaikan langkah praktikum 1 – 5	Hasil praktikum langkah 1 – 5	100

Langkah-Langkah Praktikum:

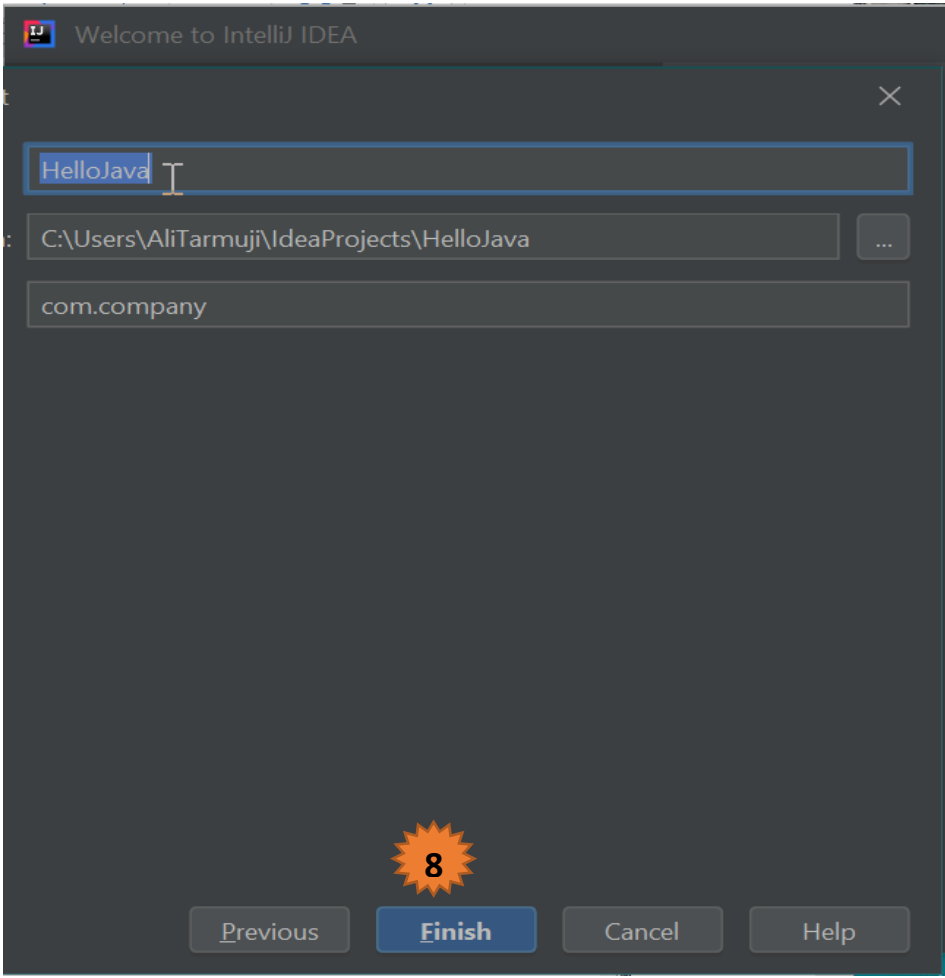
- 1. Memulai pemrograman Java dengan aplikasi IDE **IntelliJ IDEA Community** (pastikan aplikasi sudah terpasang di komputer, jika belum bisa didownload di: <http://bit.ly/intellijideacommunity>) Klik **Create New Project**, Klik **Java**, pilih **Project SDK**: SDK Java (versi diusahakan terbaru), klik **Next**



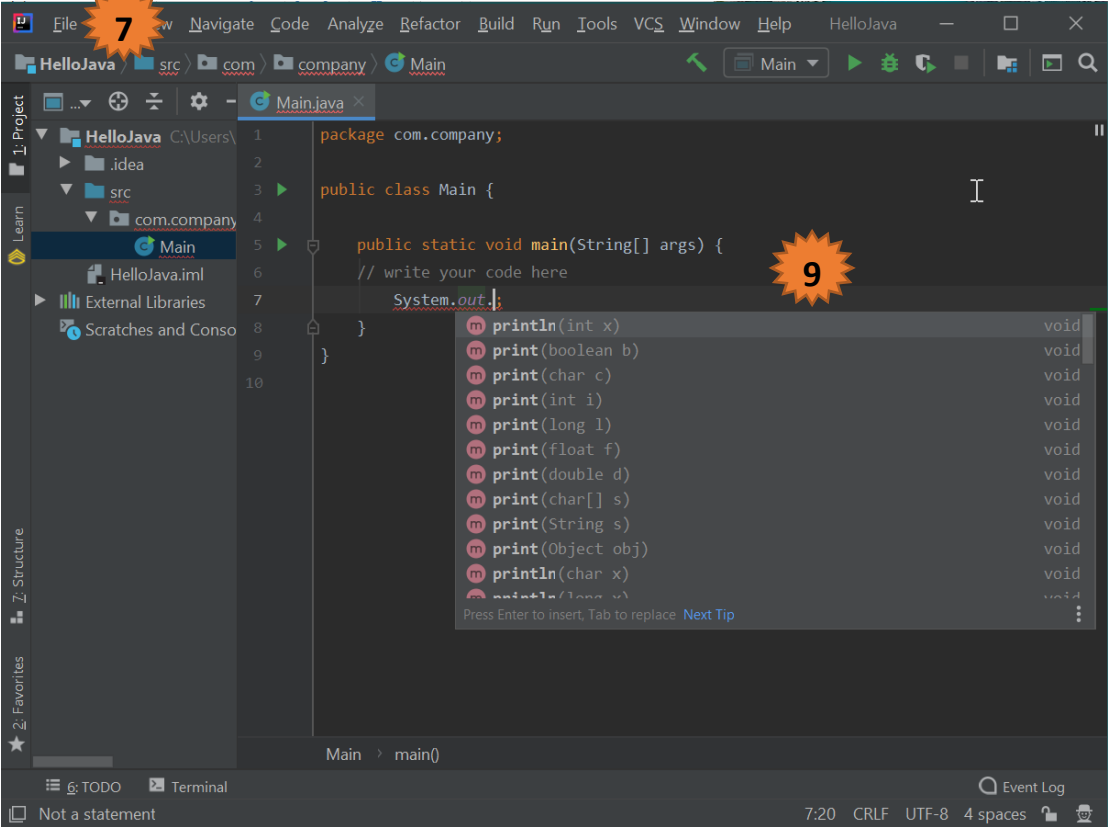
- 2. Aktifkan generate coding dengan memilih template (centrang)



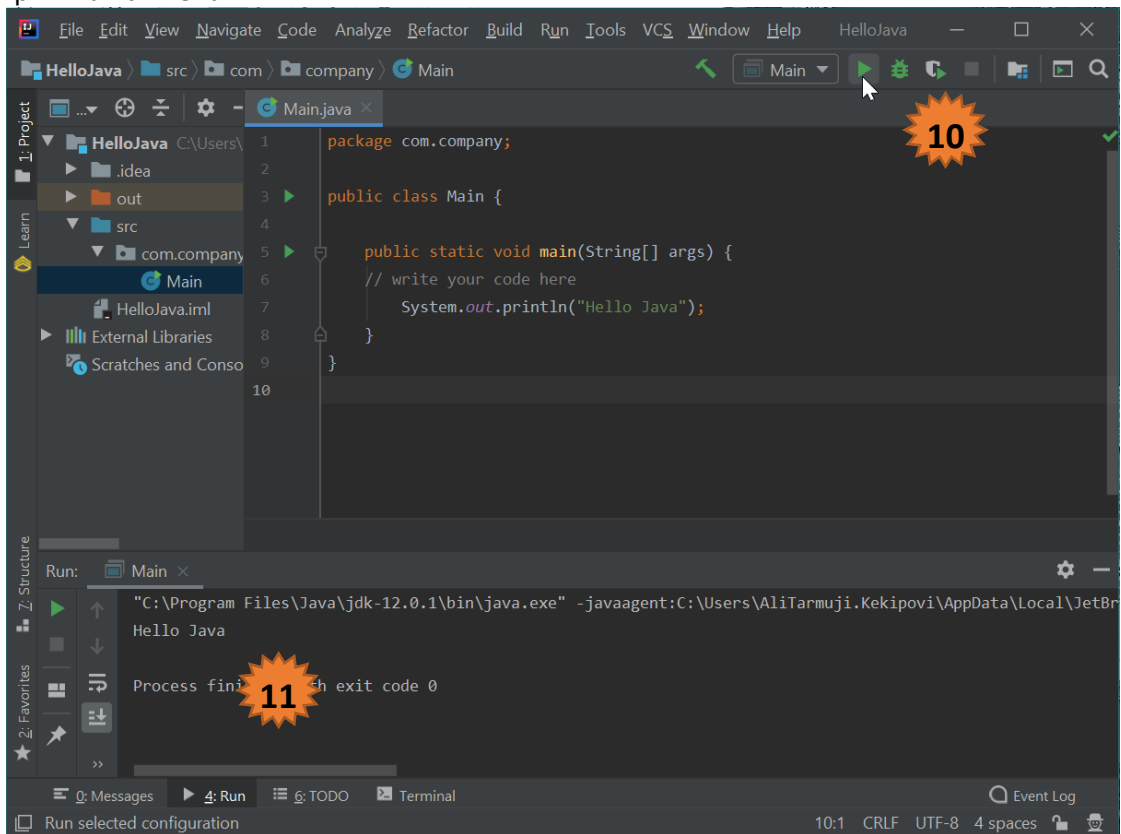
- 3. Isikan nama project sesuai kebutuhan



4. Selesai, siap ngoding



5. Cara me-running aplikasi (compile, linking, Excute). Klik tombol play di pojok kanan atas IDE atau pilih Run di menu



1.7 POST TEST

Jawablah pertanyaan berikut (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-03	CPMK-01	Buatlah program untuk membaca jari-jari lingkaran kemudian menampilkan keliling dan luas lingkaran	50
2.	CPL-03	CPMK-01	Buatlah program untuk membaca nilai angka kemudian menampilkan nilai hurufnya (A:80-100, B:65-79, C:55-64, D:40-54, E :0-53)	50

1.8 HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL-03	CPMK-01	20%		
2.	Praktik	CPL-03	CPMK-01	20%		
3.	Post-Test	CPL-03	CPMK-01	60%		

	Total Nilai
--	--------------------

LEMBAR JAWABAN PRE-TEST DAN POST-TEST PRAKTIKUM

Nama : NIM :	Asisten: Paraf Asisten:	Tanggal: Nilai:
-------------------------------	--	----------------------------------

--

PRAKTIKUM 2: KOMPONEN DASAR PEMROGRAMAN BERORIENTASI OBJEK

Pertemuan ke : 2

Total Alokasi Waktu : 90 menit

- Materi : 15 menit
- Pre-Test : 15 menit
- Praktikum : 40 menit
- Post-Test : 20 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 20 %
- Post-Test : 60 %

Pemenuhan CPL dan CPMK:

CPL-03	Mampu menerapkan konsep teoritis bidang area Informatika terkait matematika dasar dan ilmu komputer untuk memodelkan masalah dan meningkatkan produktivitas
CPL-04	Mampu berpikir logis, kritis, sistematis dan inovatif, dan mampu mengambil keputusan secara tepat di bidang keahliannya
CPMK-01	Mahasiswa mampu memahami konsep rancangan dan prinsip abstraction dan program decomposition

2.1 DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu Menjelaskan Konsep pemrograman berorientasi objek (Objek, class, instance) disertai dengan contoh kasus.

2.2 INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL-03	CPMK-01	1) Memahami konsep pemrograman berorientasi objek 2) Menyebutkan contoh objek dalam kehidupan sehari-hari/dalam contoh kasus aplikasi
CPL-04	CPMK-01	1) Menulis coding Class 2) Menulis coding instance dan penerapannya

2.3 TEORI PENDUKUNG

Sebuah **Class** mendefinisikan struktur (structure) dan tingkah laku (behaviour) sebuah objek atau sekumpulan objek. **Class** juga merupakan prototype yang mendefinisikan variabel-variabel dan method-method secara umum. Di dalam java ada aturan untuk pemberian sebuah nama class. Sebuah nama **Class** harus diawali dengan huruf besar. Hal ini untuk membedakan antara **Class** dan objek. **Class** didefinisikan dengan kata kunci class. Contoh sederhana dari deklarasi sebuah class:

```
Class Mahasiswa {
    String nim;           //deklarasi variabel atau atribut
    String nama;        //deklarasi variabel atau atribut
}
```

Sebuah Objek merupakan instansiasi dari suatu class. Kalau kita analogikan, **Class** itu sebuah cetakan sedangkan object itu adalah barang dari hasil cetakan. **Class** juga bisa dikatakan sebagai kategori, sedangkan objek adalah sesuatu yang memenuhi syarat-syarat yang harus dipenuhi agar masuk dalam kategori tersebut. Jadi bisa dibilang satu **Class** bisa mempunyai banyak objek, setiap objek mempunyai sifat yang sama persis seperti yang didefinisikan dalam **Class** tersebut. Untuk pemberian nama sebuah objek, diawali dengan huruf kecil. Pembuatan objek untuk **Class Mahasiswa** adalah sebagai berikut:

```
Mahasiswa mahasiswa;           //deklarasi objek
mahasiswa = new Mahasiswa();    //instansiasi dari kelas Mahasiswa
Mahasiswa mahasiswa= new Mahasiswa(); //dijadikan satu
```

Setiap objek mempunyai identitas yang unik, seperti halnya setiap orang mempunyai identitas yang unik. Contoh : Mahasiswa mempunyai Nim dimana nim seorang mahasiswa berbeda dengan mahasiswa yang lain. Di dalam **Class** terdapat tiga bagian utama dari sebuah kelas yang mendeklarasikan kode-kode program Java.

1. Konstruktor : digunakan untuk instansiasi objek
2. Variabel : merupakan atribut yang menyatakan keadaan dari kelas dan objek.
3. Metode (method) : berupa fungsi atau procedure.

Variabel dan metode dapat memiliki salah satu sifat berikut :

1. **Private**, tidak dapat dipanggil dari luar **Class** yang bersangkutan
2. **Protected**, hanya dapat dipanggil oleh **Class** yang bersangkutan dan anak-anak yang mewarisinya
3. **Public**, dapat dipanggil oleh siapa saja.

Dalam java terdapat dua buah metode (method) yaitu :

1. Fungsi, merupakan metode yang memiliki nilai balik jika metode tersebut dipanggil, cara pembuatan sebuah fungsi adalah dengan cara menentukan nilai baliknya, lalu membuat nama metodenya.
2. Prosedur, merupakan metode yang tidak memiliki nilai balik, cara pembuatan prosedur sama dengan fungsi namun bedanya, nilai baliknya menggunakan kata kunci **void**.

Class POJO di Java merupakan kelas dimana sebuah kelas memiliki atribut dan memiliki metode **getter** dan **setter**. Dimana atributnya bersifat private dan metode **getter** dan **setter**nya bersifat public. Metode **getter** digunakan untuk mendapatkan nilai atribut tersebut, sedangkan metode **setter** digunakan untuk mengubah nilai atribut.

Berikut contoh kelas pojo sederhana.

```
public Class Mahasiswa {
```

```

private String nim;
private String nama;
public String getName() {           // method brp Fungsi
    return nama;
}
public void setName(String nama) {   // method brp procedure
    this.nama = nama;
}
public String getNim() {             // method brp Fungsi
    return nim;
}
public void setNim(String nim) {     // method brp procedure
    this.nim = nim;
}
}

```

Constructor adalah suatu method yang pertama kali di-*running* pada saat pembuatan suatu objek. **Constructor** mempunyai ciri-ciri sebagai berikut :

- mempunyai nama yang sama persis dengan nama class
- tidak mempunyai tipe return
- digunakan untuk menginstansiasi objek
- hanya mempunyai *access modifier*, tidak ada *keyword* lain yang diletakkan sebelum nama **method** pada deklarasi constructor.

contoh Contructor

```

Class Mahasiswa {
    String nim;           //deklarasi variabel
    String nama;
    public Manusia(){}   //default constructor
    public Manusia(String nim, string nama){ //constructor
perparameter
        this.nim = nim;
        this.nama= nama;
    }
}

```

Setelah pembahasan mengenai deklarasi kelas, kelas pojo dan konstruktor, maka perlu dicoba penggunaannya dari kelas tersebut. Setiap membuat project, pasti sudah terbuat fungsi main. Berikut contoh code untuk memanggil method yang terdapat dalam kelas pojo melalui main.java. Contoh **main.java**

Source code	Hasil
<pre>public Class Prak2 { public static void main(String[] args) { Mahasiswa m= new Mahasiswa(); m.setNim("08018222"); m.setNama("Rudi"); System.out.println("Nim :"+ m.getNim()); System.out.println("Nama :"+ m.getNama()); } }</pre>	<p>Nim :08018222</p> <p>Nama :Rudi</p> <p>BUILD SUCCESSFUL (total time: 1 second)</p>

2.4 HARDWARE DAN SOFTWARE

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. IDE IntelliJ IDEA Community
3. SDK Java

2.5 PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-03	CPMK-01	Apa perbedaan object dan class?	40
2.	CPL-03	CPMK-01	Apa yang dimaksud dengan instance?	30
3.	CPL-04	CPMK-01	Apa yang menjadikan dasar bahwa suatu objek dalam dunia nyata dapat dijadikan objek dalam pemrograman?	30

2.6 LANGKAH PRAKTIKUM

Aturan Penilaian (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL-03	CPMK-01	Selesaikan langkah praktikum 1 – 2	Hasil praktikum langkah 1 – 2	100

Langkah-Langkah Praktikum:

1. Buatlah **project** baru dengan nama prak2.
2. Buatlah kelas dengan nama **PersegiPanjang** seperti pada contoh pada sesi materi..

2.7 POST TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-03 CPL-04	CPMK-01	Buatlah program untuk membuat class Titik dengan atribut absis dan ordinat, metode/operasi yang ada menentukan sebuah titik, menghitung jarak dua titik, menentukan titik tengah dua titik	50
2.	CPL-03 CPL-04	CPMK-01	Buatlah program untuk membuat class Waktu dengan atribut jam, menit, detik , metode/operasi yang ada menentukan sebuah jam, menghitung lama waktu	50

2.8 HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL-03	CPMK-01	20%		
2.	Praktik	CPL-03	CPMK-01	20%		
3.	Post-Test	CPL-03	CPMK-01	60%		
Total Nilai						

PRAKTIKUM 3: KONSTRUKTOR DAN ENKAPSULASI

Pertemuan ke : 3

Total Alokasi Waktu : 90 menit

- Materi : 15 menit
- Pre-Test : 15 menit
- Praktikum : 40 menit
- Post-Test : 20 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 20 %
- Post-Test : 60 %

Pemenuhan CPL dan CPMK:

CPL-03	Mampu menerapkan konsep teoritis bidang area Informatika terkait matematika dasar dan ilmu komputer untuk memodelkan masalah dan meningkatkan produktivitas
CPL-04	Mampu berpikir logis, kritis, sistematis dan inovatif, dan mampu mengambil keputusan secara tepat di bidang keahliannya
CPMK-02	Mahasiswa mampu memahami konsep rancangan dan prinsip encapsulation, information hiding, dan implementasinya

3.1 DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu Menjelaskan Konstruktor, enkapsulation, dan information hiding disertai dengan contoh penerapannya.

3.2 INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL-03	CPMK-02	1) Menulis coding konstruktor
CPL-04		2) Menjelaskan konstruktor dalam instance objek
		3) Mengimplementasikan enkapsulasi dan information hiding
		4) Menjelaskan perbedaan penggunaan information hiding

3.3 TEORI PENDUKUNG

Constructor (konstruktor) merupakan sebuah **pseudo-method** yang dibuat pada sebuah objek. Pada bahasa pemrograman Java, konstruktor adalah **instance method** yang memiliki nama **method** sama dengan nama **class**-nya. Sebuah konstruktor dipanggil menggunakan kata kunci **new**.

Berikut contoh konstruktor:

```
public Class Bicycle{
    public Bicycle() {
        gear = 1
        speed = 0;
    }
}
```

konstruktor ini akan dipanggil saat melakukan instansiasi seperti contoh berikut:

```
Bicycle myBike = new Bicycle();
```

`new Bicycle()` akan membuat space di memory untuk objek dan menginisialisasinya. Sebuah konstruktor dapat dilakukan overloading terhadap konstruktor lain. Overloading merupakan suatu cara untuk menggunakan satu identifier untuk merujuk kepada multiple items dalam scope yang sama. Pada bahasa pemrograman Java, sebuah method dapat di overload tetapi tidak dengan variable dan operator.

Berikut contoh overloading konstruktor:

```
public Class Bicycle{
    public Bicycle() {
        gear = 1
        speed = 0;
    }
    public Bicycle(int startSpeed, int startGear) {
        gear = startGear;
        speed = startSpeed;
    }
}
```

Untuk instansiasi dari **Class Bicycle** dapat dilakukan seperti berikut:

```
Bicycle myBike = new Bicycle(30, 8);
```

Aturan dalam pembuatan konstruktor didefinisikan dalam dua aturan berikut:

1. Nama konstruktor harus sama dengan nama kelasnya
2. Konstruktor harus tidak memiliki eksplisit return type.

Terdapat dua tipe dari konstruktor:

1. Default **Constructor** yaitu konstruktor yang tidak memiliki argument
2. Konstruktor berparameter yaitu konstruktor yang memiliki argument

Ada beberapa perbedaan antara constructor dan method, yaitu:

Java Constructor	Java Method
Constructor digunakan untuk menginisialisasi state pada sebuah objek.	Method digunakan untuk menunjukkan behavior dari object.
Constructor harus tidak mempunyai return type.	Method dapat memiliki return type.
Constructor dipanggil secara implisit.	Method dipanggil secara eksplisit

Compiler Java menyediakan default constructor jika tidak mendefinisikan constructor.	Method tidak disediakan oleh compiler
Nama constructor harus sama dengan nama Class	Nama method boleh atau tidak sama dengan nama class.

Akses modifier di Java

Ada dua macam tipe modifiers di java yaitu access modifiers dan non-access modifiers.

Access modifiers di java dikhususkan pada accessibility (scope) dari data member, method, constructor atau class.

Terdapat 4 macam java access modifiers:

1. private
2. default
3. protected
4. public

Terdapat banyak non-access modifiers seperti static, abstract, synchronized, native, volatile, transient dan lain-lain.

Berikut contoh penggunaan access modifier:

```

Class Students{
    private int data=40;
    private void msg(){
        System.out.println("Hello java");
    }
}

public Class Tester{
    public static void main(String args[]){
        Students obj = new Students();
        System.out.println(obj.data); //Compile Time Error
        obj.msg(); //Compile Time Error
    }
}

```

Penggunaan access modifier

Access Modifier	Class	Package	Subclass	Umum
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

Encapsulation di Java

Encapsulation di java adalah sebuah proses membungkus kode dan data secara bersama-sama dalam unit tunggal, sebagai contoh kapsul dalam resep racikan. Keuntungan encapsulation di java yaitu dengan menyediakan method **setter** dan **getter** dan dapat membuat **Class** menjadi read-only atau write-only. Ini menghasilkan control yang jelas terhadap akses data.

Berikut contoh penggunaan encapsulation:

Contoh 3.1 **Class** dengan nama file Student.java

```

package com.uad;

public Class Student{

    private int age=20;

    private String nama;

    public String getNama(){

        return name;

    }

    public void setName(String name){

        this.name=name

    }

}

```

Contoh 3.2 **Class** dengan nama file Test.java

```

package com.uad;

Class Test{

    public static void main(String[] args){

        Student s=new Student();

        s.setName("Romeo");

        System.out.println(s.age);

        System.out.println(s.getNama());

    }

}

```

3.4 HARDWARE DAN SOFTWARE

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. IntelliJ IDEA Community
3. Java SDK

3.5 PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-03	CPMK-02	Bagaimana cara menuliskan konstruktor pada pemrograman?	20
2.	CPL-03	CPMK-02	Apakah fungsi konstruktor dalam PBO?	40
3.	CPL-04	CPMK-02	Apa yang akan terjadi saat konstruktor yang memiliki 2 parameter hanya diberikan 1 argumen saja?	40

3.6 LANGKAH PRAKTIKUM

Aturan Penilaian (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL-03 CPL-04	CPMK-02	Selesaikan langkah praktikum 1 – 5	Hasil praktikum langkah 1 – 5	100

Langkah-Langkah Praktikum:

1. Buatlah project baru dengan nama prak3.
2. Buatlah seperti pada contoh 3.1 dan 3.2.
3. Modifikasi contoh object **Class** (contoh 4.1) dengan menambahkan konstruktor default dan konstruktor berparameter. Deklarasikan variabel bernilai tertentu di konstruktor default dan simpan nilai parameter pada sebuah variabel di konstruktor berparameter.
4. Lakukan instansiasi dengan menggunakan konstruktor default, kemudian tampilkan dilayar isi variabel dari objek hasil instansiasi tersebut.
5. Lakukan instansiasi dengan menggunakan konstruktor berparameter, kemudian tampilkan dilayar isi variabel dari objek hasil instansiasi tersebut.

3.7 POST TEST

Jawablah pertanyaan berikut (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-03 CPL-04	CPMK-02	Buatlah sebuah driver Class dengan nama Tester.java dan object Class dengan nama Karyawan.java.	20
2.	CPL-03 CPL-04	CPMK-02	Buatlah sebuah konstruktor default yang telah berisi variabel gaji tetap dan berikan nilai tertentu. Buatlah sebuah konstruktor berparameter dengan sebuah parameter dan simpan parameter tersebut pada variabel gaji tetap.	20
3.	CPL-03 CPL-04	CPMK-02	Buatlah tiga buah variabel dengan nama variabel: nomor pegawai, nama pegawai dan gaji bonus. Lakukan pembuatan method mutator dan accessor untuk tiga variabel tersebut.	30
4.	CPL-03 CPL-04	CPMK-02	Buat sebuah method menghitung gaji untuk karyawan baru yang berasal dari gaji tetap saja. Buat sebuah method menghitung gaji untuk karyawan lama yang berasal dari gaji tetap ditambahkan bonus.	30

			Tampilkan di layar berupa informasi nomor pegawai, nama pegawai dan hitungan gaji untuk objek karyawan baru dan karyawan lama.	
--	--	--	--	--

3.8 HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL-03	CPMK-02	20%		
2.	Praktik	CPL-03	CPMK-02	20%		
3.	Post-Test	CPL-03	CPMK-02	60%		
Total Nilai						

PRAKTIKUM 4: PEWARISAN

Pertemuan ke : 4

Total Alokasi Waktu : 90 menit

- Materi : 15 menit
- Pre-Test : 15 menit
- Praktikum : 40 menit
- Post-Test : 20 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 20 %
- Post-Test : 60 %

Pemenuhan CPL dan CPMK:

CPL-03	Mampu menerapkan konsep teoritis bidang area Informatika terkait matematika dasar dan ilmu komputer untuk memodelkan masalah dan meningkatkan produktivitas
CPMK-02	Mahasiswa mampu memahami konsep rancangan dan prinsip encapsulation, information hiding, dan implementasinya

4.1 DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu Menjelaskan inheritance dan contoh kasus penerapannya.

4.2 INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL-03	CPMK-02	1. Mengimplementasikan inheritance pada suatu kasus pemrograman 2. Menulis coding inheritance
--------	---------	--

4.3 TEORI PENDUKUNG

Inheritance (pewarisan) merupakan ciri utama dari pemrograman berorientasi objek dimana sifat-sifat yang terdapat pada kelas induk akan dimiliki oleh kelas turunannya. Kelas induk yang diturunkan disebut dengan superclass. Adapun kelas baru hasil turunan disebut subclass. Pada proses penurunan kelas ini, kelas turunan akan mewarisi sifat-sifat yang terdapat pada kelas induknya. Selanjutnya, kelas turunan tersebut dapat memiliki sifat-sifat spesifik yang sebelumnya tidak di miliki oleh kelas induk. Misal manusia mempunyai nama dan alamat. Mahasiswa mempunyai nim, nama dan alamat. Maka kelas manusia mewarisi sifatnya ke kelas mahasiswa. Karena sifatnya terdapat kesamaan sifat. Berikut contohnya :

Contoh 4.1 Kelas Pojo Induk (Manusia.java)

```
public class Manusia {  
    private String nama;  
    private String alamat;  
  
    public Manusia() {}  
  
    public Manusia(String nama, String alamat) {  
        this.nama = nama;  
        this.alamat = alamat;  
    }  
    public String getAlamat() {  
        return alamat;  
    }  
    public void setAlamat(String alamat) {  
        this.alamat = alamat;  
    }  
    public String getNama() {  
        return nama;  
    }  
    public void setNama(String nama) {  
        this.nama = nama;  
    }  
}
```

Contoh 4.2 Kelas Pojo Anak (Mahasiswa.java)

```
public class Mahasiswa {  
    private String nim;  
    private String nama;  
    private String alamat;  
  
    public Mahasiswa() {  
    }  
    public Mahasiswa(String nim, String nama, String alamat) {  
        this.nim = nim;  
    }  
}
```

```

        this.nama = nama;

        this.alamat = alamat;
    }

    public String getAlamat() {
        return alamat;
    }

    public void setAlamat(String alamat) {
        this.alamat = alamat;
    }

    public String getNama() {
        return nama;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }

    public String getNim() {
        return nim;
    }

    public void setNim(String nim) {
        this.nim = nim;
    }
}

```

Kode diatas bisa dibandingkan pada bagian atribut dan method. beberapa atribut dan metode yang sama, yaitu nama, alamat, setNama(), getNama(), setAlamat() dan getAlamat(). Artinya banyak terjadi duplikasi kode, oleh karena itu lebih baik kelas tersebut digabungkan menggunakan pewarisan, yaitu Manusia diturunkan menjadi Mahasiswa, karena semua atribut dan metode Manusia ada di Mahasiswa namun tidak semua atribut dan metode Mahasiswa ada di kelas Manusia. Untuk mengatakan bahwa kelas X turunan dari kelas Y kita dapat menggunakan kata kunci extends. Dengan begitu kita hanya perlu mengubah kelas Mahasiswa menjadi seperti berikut.

Contoh 4.3 Penurunan Kelas (Mahasiswa.java)

```

public class Mahasiswa extends Manusia{
    private String nim;

    public Mahasiswa() {
    }
}

```

```

public Mahasiswa(String nim) {
    this.nim = nim;
}
public String getNim() {
    return nim;
}
public void setNim(String nim) {
    this.nim = nim;
}
}

```

Walaupun kelas Mahasiswa tidak memiliki atribut dan metode untuk nama dan alamat, namun sebenarnya Mahasiswa tersebut memilikinya, karena Mahasiswa merupakan turunan dari Manusia, sehingga seluruh sifat dari Manusia ada pada Mahasiswa.

Contoh 4.4 Main.java

Source Code	Hasil
<pre> public class Main { public static void main(String[] args) { Mahasiswa m=new Mahasiswa(); m.setNim("06018251"); m.setNama("Tejo"); m.setAlamat("Sleman"); System.out.println("Nim :"+m.getNim()); System.out.println("Nama :"+m.getNama()); System.out.println("Alamat :"+m.getAlamat()); } } </pre>	<pre> Nim :06018251 Nama :Tejo Alamat :Sleman BUILD SUCCESSFUL (total time: 1 second) </pre>

4.4 HARDWARE DAN SOFTWARE

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. IntelliJ IDEA Community
3. Java SDK

4.5 PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-03	CPMK-02	Pada Inheritance, apa saja yang dapat diturunkan oleh superclass kepada subclassnya?	30
2.	CPL-03	CPMK-02	Apakah memungkinkan superclass mengambil sifat subclass nya? Jelaskan!	40
3	CPL-03	CPMK-02	Apa saja yang menjadi pertimbangan suatu class harus diturunkan ke dalam subclass?	40

4.6 LANGKAH PRAKTIKUM

Aturan Penilaian (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL-03	CPMK-02	Selesaikan langkah praktikum 1 – 2	Hasil praktikum langkah 1 – 2	100

Langkah-Langkah Praktikum:

1. Buat project baru dan beri nama project dengan Praktikum 4.
2. Silahkan mencoba contoh 4.1, 4.3 dan 4.4.

4.7 POST TEST

Jawablah pertanyaan berikut (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-03	CPMK-02	Buatlah project seperti perintah langkah-langkah praktikum dengan studi kasus membuat kelas lingkaran, kelas persegi panjang disertai metodenya yang merupakan turunan dari kelas titik	100

4.8 HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL-03	CPMK-02	20%		
2.	Praktik	CPL-03	CPMK-02	20%		
3.	Post-Test	CPL-03	CPMK-02	60%		
Total Nilai						

PRAKTIKUM 5: POLYMORPHISME

Pertemuan ke : 5

Total Alokasi Waktu : 90 menit

- Materi : 15 menit
- Pre-Test : 15 menit
- Praktikum : 40 menit
- Post-Test : 20 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 20 %
- Post-Test : 60 %

Pemenuhan CPL dan CPMK:

CPL-03	Mampu menerapkan konsep teoritis bidang area Informatika terkait matematika dasar dan ilmu komputer untuk memodelkan masalah dan meningkatkan produktivitas
CPL-04	Mampu berpikir logis, kritis, sistematis dan inovatif, dan mampu mengambil keputusan secara tepat di bidang keahliannya
CPMK-02	Mahasiswa mampu memahami konsep rancangan dan prinsip encapsulation, information hiding, dan implementasinya

5.1 DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu menjelaskan konsep overloading dan overriding dan contoh kasus penerapannya.

5.2 INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL-03	CPMK-02	mahasiswa dapat :
CPL-04		1) Menjelaskan perbedaan overloading dan overriding
		2) Mengimplementasikan penggunaan overriding dan overloading

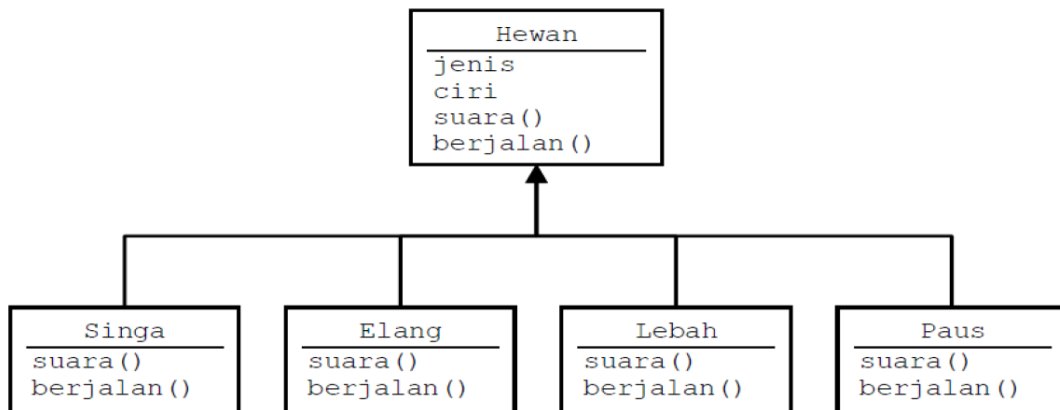
5.3 TEORI PENDUKUNG

Polimorphism berarti mempunyai banyak bentuk. Dua objek dikatakan sebagai polymorphic bila objek-objek itu mempunyai antarmuka-antarmuka yang identik namun mempunyai perilaku-perilaku berbeda. Polymorphism berupa satu nama tunggal yang menyatakan objek-objek kelas-kelas berbeda yang terhubung dengan suatu superkelas yang common di antara kelas-kelas itu. Dengan Polimorphism dapat dikenali dan dieksploitasi keserupaan-keserupaan diantara kelas-kelas berbeda

Ada 2 macam bentuk polymorphism, yaitu:

- **Overloading:** penggunaan satu nama untuk beberapa method yang berbeda dalam suatu class. Nama method yang sama dibedakan dari jumlah parameter dan tipe data parameternya.
- **Overriding:** mendeklarasikan sebuah method dengan nama dan parameter yang sama dari superclass-nya, kemudian dimodifikasi ulang (menumpuk/mendefinisi ulang).

Pada polymorphism, dikenal juga dynamic binding atau ikatan dinamis yang umumnya digunakan pada variabel bertipe class. Jika kita mempunyai variabel bertipe superclass, variabel ini dapat diisi dengan object superclass ataupun subclass-nya tanpa melakukan perubahan tipe.



Contoh Polymorphisme

Pada gambar di atas suara superclass hewan bisa berbagai jenis tergantung subclassnya.

Contoh pertama :

```

class EkspresiWajah{
    public String respons() {
        return("Perhatikan ekspresi wajah saya");
    }
}
class Gembira extends EkspresiWajah{
    public String respons() {
        return("ha ha ha...");
    }
}
class Sedih extends EkspresiWajah{
    public String respons() {
        return("hik hik ngeee ngeee ngeee");
    }
}
class Marah extends EkspresiWajah{
    public String respons() {
        return("Hai kurang ajar...!");
    }
}

class MainEkspresiWajah{
    public static void main(String args[]) {
        EkspresiWajah objEkspresi = new EkspresiWajah();
        Gembira objGembira = new Gembira();
        Sedih objSedih = new Sedih();
        Marah objMarah = new Marah();

        EkspresiWajah[] arrEkspresi = new EkspresiWajah[4];
        arrEkspresi[0] = objEkspresi;
    }
}
  
```

```

arrEkspresi[1] = objGembira;
arrEkspresi[2] = objSedih;
arrEkspresi[3] = objMarah;

System.out.println("Ekspresi[0] : "+arrEkspresi[0].respons());
System.out.println("Ekspresi[1] : "+arrEkspresi[1].respons());
System.out.println("Ekspresi[2] : "+arrEkspresi[2].respons());
System.out.println("Ekspresi[3] : "+arrEkspresi[3].respons());
    }
}

```

Contoh kedua :

```

public class Employee {
    private String name;
    private double salary;
    protected Employee(String n, double s) {
        name = n;
        salary = s;
    }
    protected String getDetails() {
        return "Name : "+name+ "\nSalary : "+salary;
    }
    public void cetak() {
        System.out.println("coba di Employee");
    }
}

public class Manager extends Employee {
    private String department;
    public Manager(String nama, double salary, String dep) {
        super(nama, salary);
        department = dep;
    }
    public String getDepartment() {
        return department;
    }
    public String getDetails() {
        return super.getDetails()+ "\nDepartment : "+getDepartment();
    }
    public void cetak() {
        System.out.println("Coba di Manager");
    }
}

public class View {
    public static void main(String[] args) {
        Employee e = new Employee("Ali",1200000);
        Employee em = new Manager("Ali",1200000,"Informatika");
        System.out.println("Data employee : \n"+e.getDetails());
        System.out.println("Data manager : \n"+em.getDetails());
        em.cetak();
    }
}

```

Catatan :

Kalau method cetak() di kelas Employee dan kelas Manager ada, maka yang dijalankan adalah method milik kelas Manager. Prioritasnya adalah kelas Manager kemudian kelas Employee.

5.4 HARDWARE DAN SOFTWARE

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. IntelliJ IDEA Community
3. Java SDK

5.5 PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-03 CPL-04	CPMK-02	Apa yang dimaksud dengan method overloading? Berikan contohnya!	30
2.	CPL-03 CPL-04	CPMK-02	Mengapa perlu adanya method Overriding?	40
3	CPL-03 CPL-04	CPMK-02	Berikan contoh kasus penggunaan method Overriding	30

5.6 LANGKAH PRAKTIKUM

Aturan Penilaian (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL-03	CPMK-02	Selesaikan langkah praktikum 1 – 7	Hasil praktikum langkah 1 – 7	100

Langkah-Langkah Praktikum:

1. Buat project prak5
2. Kerjakan dua contoh diatas dan amati hasilnya
3. Buatlah sebuah kelas pojo dengan nama Mahasiswa seperti code dibawah ini.

```
public class Mahasiswa {
    private String nim;
    private String nama;

    public Mahasiswa() {
    }

    public Mahasiswa(String nim, String nama) {
        this.nim = nim;
        this.nama = nama;
    }

    public String getNama() {
        return nama;
    }

    public void setNama(String nama) {
```

```

        this.nama = nama;
    }
    public String getNim() {
        return nim;
    }
    public void setNim(String nim) {
        this.nim = nim;
    }
}

```

4. Buatlah sebuah kelas HM yang merupakan kelas turunan dari Mahasiswa dan buatlah methodmethodnya.
5. Buatlah sebuah kelas KelompokStudi yang merupakan kelas turunan dari Mahasiswa dan buatlah methodnya
6. Buatlah 1 objek untuk menginstantiasi kelas HM dan mengimplementasikan beberapa method yang telah Anda definisikan dalam kelas HM
7. Panggilah objek yang telah Anda buat pada kelas KelompokStudi dan HM untuk bekerja sehingga tampil efek-efek polymorfisme

5.7 POST TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-03 CPL-04	CPMK-02	Kembangkan program dari ilustrasi gambar di atas dengan mengimplementasikan method-method yang ada kemudian buatlah testernya	100

5.8 HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL-03	CPMK-02	20%		
2.	Praktik	CPL-03	CPMK-02	20%		
3.	Post-Test	CPL-03	CPMK-02	60%		
Total Nilai						

PRAKTIKUM 6: RELASI ANTAR CLASS

Pertemuan ke : 6

Total Alokasi Waktu : 90 menit

- Materi : 15 menit
- Pre-Test : 15 menit
- Praktikum : 40 menit
- Post-Test : 20 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 20 %
- Post-Test : 60 %

Pemenuhan CPL dan CPMK:

CPL-03	Mampu menerapkan konsep teoritis bidang area Informatika terkait matematika dasar dan ilmu komputer untuk memodelkan masalah dan meningkatkan produktivitas
CPMK-02	Mahasiswa mampu memahami konsep rancangan dan prinsip encapsulation, information hiding, dan implementasinya

6.1 DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu menjelaskan desain class diagram dan hubungan antar class

6.2 INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL-01	CPMK-02	<ol style="list-style-type: none"> 1) Menerapkan desain class diagram 2) Mengimplementasikan hubungan antar class 3) Membangkitkan kode program dari class diagram
--------	---------	---

6.3 TEORI PENDUKUNG

Pengantar tentang Array

Untuk mengawali hubungan antar kelas perlu mereview lagi tentang array (larik). Array digunakan untuk menggolongkan data atau obyek yang bertipe sejenis. Array dideklarasikan dengan tipe apa saja, baik itu yang bertipe data primitif maupun obyek. Berikut contoh deklarasi Array :

```
char c[];
titik t[]; // t adalah sebuah kelas
```


Dalam bahasa pemrograman Java, Array merupakan sebuah obyek meskipun ia terdiri dari elemen yang bertipe data primitif. Seperti halnya kelas yang lain, ketika mendeklarasikan Array belum dibentuk sebuah obyek Array. Deklarasi Array hanya membuat sebuah referensi yang dapat digunakan untuk mengacu ke sebuah obyek Array. Besarnya memori yang digunakan oleh elemen-elemen Array akan dialokasikan secara dinamis dengan menggunakan pernyataan `new` atau dengan array initializer.

Contoh deklarasi Array di atas, dengan menggunakan kurung siku setelah nama variabel, merupakan standar penulisan Array dalam C, C++ dan Java. Format demikian agak sulit untuk dibaca. Oleh karenanya, bahasa Java menggunakan bentuk deklarasi Array alternatif dengan menggunakan kurung siku di sebelah kiri seperti dalam contoh berikut :

```
char [] c;
titik [] t; // titik adalah sebuah kelas
```

Sebuah deklarasi Array dapat ditulis sebagai pendeklarasian tipe Array di bagian kiri dan nama variabel di bagian kanan. Kedua bentuk deklarasi Array di atas dapat digunakan, tetapi sebaiknya konsisten dalam menggunakan satu bentuk saja. Dalam deklarasi Array, tidak ditentukan ukuran aktual dari Array. Ketika mendeklarasikan Array dengan kurung siku yang berada di sebelah kiri nama variabel, kurung siku tersebut berlaku bagi semua variabel yang berada di sebelah kanannya.

Membuat Array

Array dibuat dengan menggunakan keyword `new`. Contoh di bawah ini membuat sebuah Array dengan tipe primitif `char`:

```
c = new char[26];
```

Indeks Array dimulai dengan angka 0. Batasan legal untuk nilai indeks ini adalah dari nol hingga jumlah elemen array-1. Apabila program berusaha mengakses Array di luar batas yang legal, maka akan muncul runtime exception.

Untuk membuat Array dengan elemen obyek, gunakan cara penulisan berikut

```
t = new titik[5];
```

Pernyataan tersebut tidak membuat 5 obyek Titik. Untuk membuat obyek Titik, gunakan pernyataan berikut :

```
t[0] = new titik(1, 5);
t[1] = new titik (2, 4);
```

Menginisialisasi Array

Java memiliki cara singkat untuk membuat sebuah obyek Array dengan elemen-elemennya memiliki nilai awal :

```
String names[] = {"Anni", "Syauqi", "Savitri"};
```

Kode di atas adalah sama dengan kode berikut ini:

```
String names[];
names[0] = "Anni";
names[1] = "Syauqi";
names[2] = "Savitri";
```

Array Multidimensi

Dalam Java, dapat dibuat Array dari Array sehingga dinamai Array multidimensi. Contoh berikut ini memperlihatkan cara membuat Array dua dimensi :

```
int twoDim [][] = new int[4][];
twoDim[0] = new int[5];
twoDim[1] = new int[5];
```

Batasan Array

Dalam bahasa Java, indeks Array dimulai dengan angka nol. Jumlah elemen di dalam Array disimpan sebagai bagian dari obyek Array di dalam atribut `length`. Atribut ini dapat digunakan untuk melakukan proses iterasi pada Array seperti dalam contoh berikut :

```
int list[] = new int[10];
for(int i = 0; i < list.length; i++) {
    System.out.println(list[i]);
}
```

Dengan menggunakan atribut `length`, pemeliharaan kode program menjadi mudah karena program tidak perlu mengetahui jumlah elemen Array pada saat kompilasi.

Mengubah Ukuran Array

Setelah membuat obyek Array, ukuran Array tersebut tidak dapat diubah. namun demikian, dapat digunakan variabel referensi yang sama untuk menunjuk ke sebuah obyek Array baru seperti dalam contoh di bawah ini :

```
int myArray[] = new int[6];
myArray = new int[10];
```

Menyalin Array

Bahasa Java menyediakan method khusus untuk menyalin Array, yaitu dengan menggunakan method `arrayCopy()` dari kelas `System` seperti dalam contoh berikut :

```
// array semula
int myArray[] = {1, 2, 3, 4, 5, 6};
// array engan elemen yang lebih banyak
int hold[] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1};
//menyalin semua elemen myArray ke hold
// dimulai dengan indeks ke 0
System.arraycopy(myArray, 0, hold, 0, myArray.length);
```

Setelah proses pengkopian di atas, maka isi dari Array `hold` adalah 1, 2, 3, 4, 5, 6, 4, 3, 2, 1.

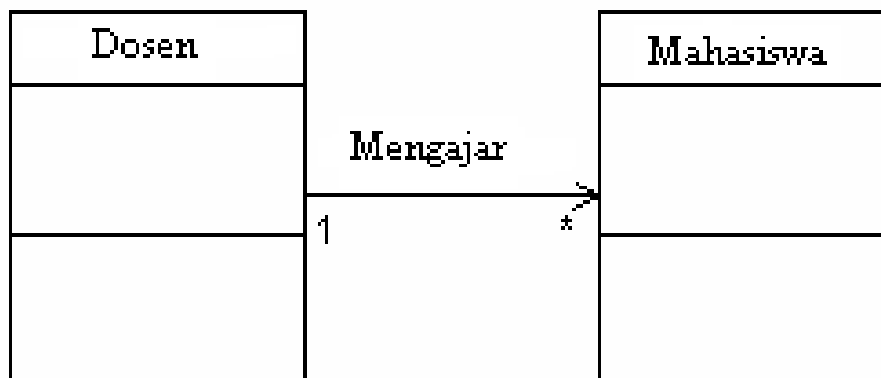
Hubungan Antar Kelas

Dalam Object Oriented Programming, kelas-kelas yang terbentuk dapat memiliki hubungan satu dengan yang lainnya, sesuai dengan kondisi dari kelas-kelas yang bersangkutan. Ada beberapa jenis hubungan yang dapat terjadi antara kelas yang satu dengan kelas yang lainnya, antara lain:

1. Asosiasi
2. Agregasi
3. Komposisi
4. Generalisasi (terkait dengan pewarisan)
5. Spesialisasi (terkait dengan pewarisan)

1. Asosiasi

Asosiasi merupakan hubungan antara dua kelas di yang merupakan hubungan struktural yang menggambarkan himpunan link antar obyek. Contoh dari hubungan asosiasi ini adalah:



Bentuk implementasi dari diagram kelas tersebut dapat dilihat pada Contoh 1.

Contoh 1:

```

class Mahasiswa {
    private String nim;
    private String nama;

    public Mahasiswa(String nim, String nama)
    {
        this.nim=nim;
        this.nama=nama;
    }
    public void setnama (String nama) {
        this.nama = nama;
    }
    public void setnim (String nim) {
        this.nim = nim;
    }
    public String getNim() {
        return this.nim;
    }
    public String getNama () {
        return this.nama;
    }
}

class Dosen {
    private String Kddosen;
    private String[] nimMHS=new String[5];
    private int jmlMahasiswa=0;

    public Dosen(String kode)
    {
        this.Kddosen=kode;
    }
    public void setKddosen (String Kddosen) {
        this.Kddosen = Kddosen;
    }
    public void setNimMahasiswa (String nim) {
        nimMHS[jmlMahasiswa]=nim;
        jmlMahasiswa++;
    }
}
  
```

```

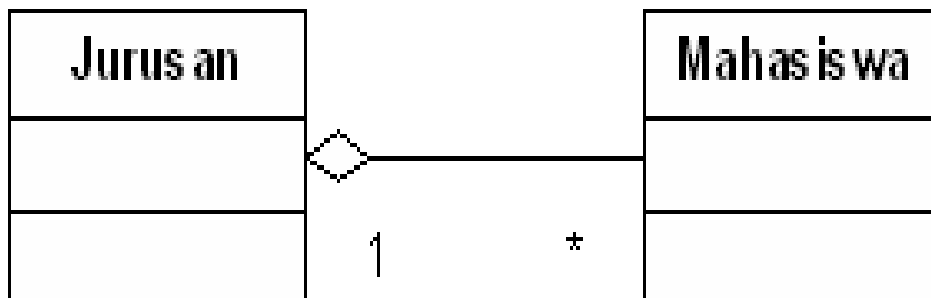
public int getJmlMahasiswa () {
return this.jmlMahasiswa;
}
public String getKddosen () {
return this.Kddosen;
}
public void daftarMahasiswa() {
System.out.println("Kode Dosen "+Kddosen);
System.out.println("Daftar Mahasiswa");
for (int i=0;i<jmlMahasiswa;i++)
{
System.out.println(nimMHS[i]);
}
}
}

class MahasiswaDosen
{
public static void main(String[] args)
{
Mahasiswa mhs1 = new Mahasiswa("30107998","Abdul Kadir");
Mahasiswa mhs2 = new Mahasiswa("30107999","Asep Sumarta");
Dosen dsn = new Dosen("SKS");
dsn.setNimMahasiswa(mhs1.getNim());
dsn.setNimMahasiswa(mhs2.getNim());
dsn.daftarMahasiswa();
}
}

```

2. Agregasi

Agregasi merupakan hubungan antara dua kelas di mana kelas yang satu merupakan bagian dari kelas yang lain namun kedua kelas ini dapat berdiri sendiri-sendiri. Agregasi sering juga disebut relasi “part of” atau relasi “whole-part”. Contoh hubungan agregasi ini adalah:



Implementasi dari diagram kelas tersebut dalam Java adalah sebagai berikut:

Contoh 2:

```

//mahasiswa.java
public class mahasiswa {
private String NIM, Nama;
public mahasiswa(String no, String nm) {
this.NIM = no;
this>Nama = nm;
}
public String GetNIM() {
return (NIM);
}
public String GetNama() {
return (Nama);
}
}

```

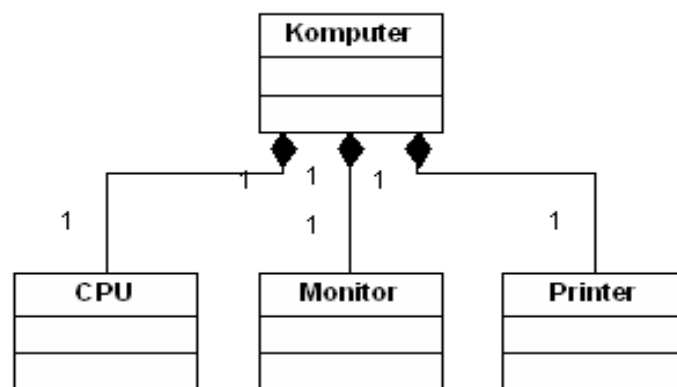
```

}
//jurusan.java
public class Jurusan {
    private String KodeJurusan, NamaJurusan;
    private Mahasiswa[] Daftar = new Mahasiswa[10];
    public Jurusan(String kode, String nama) {
        this.KodeJurusan = kode;
        this>NamaJurusan = nama;
    }
    private static int JmlMhs = 0;
    public void AddMahasiswa (Mahasiswa m) {
        this.Daftar[JmlMhs] = m;
        this.JmlMhs++;
    }
    public void DisplayMahasiswa() {
        int i;
        System.out.println("Kode Jurusan : "+this.KodeJurusan);
        System.out.println("Nama Jurusan : "+this>NamaJurusan);
        System.out.println("Daftar Mahasiswa :");
        for (i=0;i<JmlMhs;i++)
        System.out.println(Daftar[i].GetNIM()+" "+Daftar[i].GetNama());
    }
}

```

3. Komposisi

Komposisi merupakan bentuk khusus dari agregasi di mana kelas yang menjadi part (bagian) baru dapat diciptakan setelah kelas yang menjadi whole (seluruhnya) dibuat dan ketika kelas yang menjadi whole dimusnahkan, maka kelas yang menjadi part ikut musnah. Contoh hubungan komposisi adalah sebagai berikut:



Implementasi dari diagram kelas tersebut dalam Java adalah sebagai berikut:

```

//CPU.java
public class CPU {
    private String Merk;
    private int Kecepatan;
    public CPU(String m, int k) {
        this.Merk = m;
        this.Kecepatan = k;
    }
    public void DisplaySpecCPU() {
        System.out.println(this.Merk + ", " + this.Kecepatan);
    }
}

//Monitor.java
public class Monitor {
    private String Merk;
    public Monitor(String m) {

```

```

this.Merk = m;
    }
public void DisplaySpecMonitor() {
    System.out.println(this.Merk);
    }
}

//Printer.java
public class Printer {
private String Merk, Type;
public Printer(String m, String t) {
    this.Merk = m;
    this.Type = t;
    }
public void DisplaySpecPrinter() {
    System.out.println(this.Merk + ", " + this.Type);
    }
}

//Komputer.java
public class Komputer {
private String Kode;
private long Harga;
    private CPU Proc;
    private Monitor Mon;
    private Printer Prn ;

public Komputer(String k, long h) {
    this.Kode = k;
    this.Harga = h;
    if (k == "PC-01") {
        Proc = new CPU("Pentium IV", 500);
        Mon = new Monitor("Sony Multiscan 15sf");
        Prn = new Printer("Canon BJC-210SP", "Color");
    }
}

    public void DisplaySpec() {
        System.out.println("Kode      : " + this.Kode);
        System.out.print("Processor: ");
        Proc.DisplaySpecCPU();
        System.out.print("Monitor  : ");
        Mon.DisplaySpecMonitor();
        System.out.print("Printer  : ");
        Prn.DisplaySpecPrinter();
        System.out.println("Harga : Rp. " + this.Harga);
    }
}

```

6.4 HARDWARE DAN SOFTWARE

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. IntelliJ IDEA Community
3. Java SDK

6.5 PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-03	CPMK-02	Jelaskan hubungan agregasi antar class	20
2.	CPL-03	CPMK-02	Jelaskan hubungan komposisi antar class	20
3.	CPL-03	CPMK-02	Jelaskan hubungan komposisi antar class	20
4.	CPL-03	CPMK-02	Adakah ciri atau sintak khusus yang menandakan hubungan antar class? Jelaskan jawaban anda!	40

6.6 LANGKAH PRAKTIKUM

Aturan Penilaian (**Total Skor: 100**):

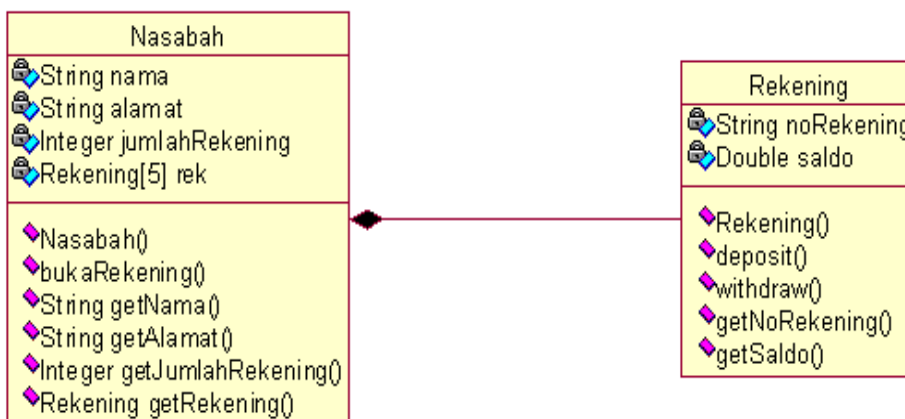
No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL-03	CPMK-02	Selesaikan langkah praktikum 1	Hasil praktikum langkah 1	100

Langkah-Langkah Praktikum:

Implementasikan dan jalankan contoh relasi antar class asosiasi, agregasi dan komposisi. Berikan komentar pada code yang ditulis, bagian mana yang merupakan implemntasi relasi antar class!

6.7 POST TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):



Gambar 6.4. Diagram Nasabah

Diketahui class diagram pada Gambar 6.4 menggambarkan hubungan antara kelas nasabah dan kelas rekening. Seorang nasabah bisa membuat maksimal 5 rekening pada sebuah bank.

Pada kelas nasabah:

- Terdapat atribut nama untuk menyimpan nama nasabah, alamat untuk menyimpan alamat nasabah, jumlahRekening untuk menyimpan jumlah rekening nasabah, dan rek untuk menyimpan data rekening nasabah
- Terdapat konstruktor nasabah untuk mengeset nama dan alamat nasabah
- Terdapat method bukaRekening untuk mengeset rekening baru

Pada kelas rekening:

- Terdapat atribut noRekening untuk menyimpan nomor rekening dan saldo untuk menyimpan nilai saldo
- Terdapat method deposit untuk menambahkan nilai saldo
- Terdapat method withdraw untuk mengurangi nilai saldo

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-03	CPMK-02	Jelaskan jenis relasi antar class pada Gambar 6.4.!	20
2.	CPL-03	CPMK-02	Buatlah code Java untuk mengimplementasikan class diagram Gambar 6.4!	50
3.	CPL-03	CPMK-02	Buatlah class tester untuk menampilkan data seorang nasabah dan rekeningnya! (data diasumsikan sendiri)	30

6.8 HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL-03	CPMK-02	20%		
2.	Praktik	CPL-03	CPMK-02	20%		
3.	Post-Test	CPL-03	CPMK-02	60%		
Total Nilai						

PRAKTIKUM 7: CLASS ABSTRAK

Pertemuan ke : 7

Total Alokasi Waktu : 90 menit

- Materi : 15 menit
- Pre-Test : 15 menit
- Praktikum : 40 menit
- Post-Test : 20 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 20 %
- Post-Test : 60 %

Pemenuhan CPL dan CPMK:

CPL-03	Mampu menerapkan konsep teoritis bidang area Informatika terkait matematika dasar dan ilmu komputer untuk memodelkan masalah dan meningkatkan produktivitas
CPMK-02	Mahasiswa mampu memahami konsep rancangan dan prinsip encapsulation, information hiding, dan implementasinya

7.1 DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu Menjelaskan kelas abstrak, contoh kasus kelas abstrak dan penerapannya.

7.2 INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL-03	CPMK-02	1) Mengimplementasikan kelas abstrak 2) Menulis coding kelas abstrak
--------	---------	---

7.3 TEORI PENDUKUNG

Abstract Class merupakan kelas yang berada pada posisi tertinggi dalam sebuah hierarki kelas. Sesuai dengan namanya, abstract class dapat didefinisikan pada class itu sendiri. Berikut adalah cara mendeklarasikan abstract class:

```
abstract class NamaKelas{
    //isi abstract class
}
```

Contoh

```
abstract class Manusia{
    //isi abstract class
}
```

Didalam abstract class dapat juga diberi abstract method (Optional). Penggunaan abstract method tidak diperlukan statement dalam method tersebut. Berikut cara mendeklarasikan abstract method pada abstract class:

```
abstract class NamaKelas{
    //untuk method berjenis prosedur
    [access_modifier] abstract void [namaMethod]();

    //untuk method berjenis fungsi
    [access_modifier] abstract [tipe_data] [namaMethod]();
}
```

Contoh:

```
abstract class Manusia{
    //untuk method berjenis prosedur
    public abstract void setNama();

    //untuk method berjenis fungsi
    public abstract String getNama();
}
```

Catatan:

1. Apabila dalam abstract class terdapat abstract method dan kelas tersebut diturunkan ke kelas turunannya, maka method tersebut harus dideklarasikan ulang (overriding method) dengan diberi statement pada isi methodnya.
2. Apabila class tersebut merupakan abstract class, maka class tersebut bisa terdapat abstract method atau tidak (optional). Sedangkan apabila kelas tersebut, terdapat abstract method, maka kelas tersebut wajib berbentuk abstract class.

Keyword “final”

Keyword “final” digunakan untuk mencegah suatu class diturunkan atau suatu method dilakukan pendeklarasian ulang (overriding method). Berikut adalah cara mendeklarasikan *final* dalam class:

```
final class NamaKelas{
    //isi class
}
```

Contoh

```
final class Manusia{
```

```

        //isi class
    }

```

Sedangkan cara mendeklarasikan “final” pada method adalah sebagai berikut:

```

final class NamaKelas{
    //untuk method berjenis prosedur
}

```

Contoh

```

final class Manusia{
    //method berjenis prosedur
    public final void cetakBeratBadanIdeal(){
        //isi method
    }
    //method berjenis prosedur
    public final double hitungBeratBadanIdeal(){
        //isi method
        return 0;
    }
}

```

7.4 HARDWARE DAN SOFTWARE

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. IntelliJ IDEA Community
3. Java SDK

7.5 PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-03	CPMK-02	Jelaskan perbedaan antara abstract class dan superclass!	50
2.	CPL-03	CPMK-02	Kapan atau dalam kondisi apa Perlu dibuat class abstrak?	50

7.6 LANGKAH PRAKTIKUM

Aturan Penilaian (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL-03	CPMK-02	Selesaikan langkah praktikum 1 – 5	Hasil praktikum langkah 1 – 5	
2.	dst				

Langkah-Langkah Praktikum:

1. Buat project prak8
2. Perhatikan class diagram berikut:



Untuk menghitung berat badan ideal sesuai dengan class adalah:

- Laki-laki = (tinggi badan(cm)-100) kg X 90%
- Perempuan = (tinggi badan(cm)-100) kg X 80%

3. Buatlah sebuah kelas pojo dengan nama Manusia seperti code dibawah ini.

```

public class Manusia {
    private double tinggiBadan;

    public Manusia () {
    }

    public Manusia (double tb) {
        this.tinggiBadan = tb;
    }

    public double getTinggi() {
        return tinggiBadan;
    }

    public abstract double hitungBeratBadanIdeal();
}
  
```

4. Buatlah sebuah kelas Lakilaki seperti code dibawah ini.

```

public class Lakilaki extends Manusia{

    public Lakilaki (double tinggiBadan) {
        super(tinggiBadan);
    }

    public double hitungBeratBadanIdeal(){
        return (super.getTinggiBadan()-100)*0.9;
    }

}
  
```

5. Buatlah sebuah kelas Perempuan seperti code dibawah ini.

```
final public class Perempuan extends Manusia {  
    public Perempuan (double tinggiBadan) {  
        super(tinggiBadan);  
    }  
    public final double hitungBeratBadanIdeal(){  
        return (super.getTinggiBadan()-100)*0.8;  
    }  
}
```

Keyword “final” pada kelas perempuan digunakan untuk mencegah pembuatan kelas baru dari kelas turunan perempuan. Sedangkan keyword “final” pada method digunakan untuk mencegah pendeklarasian (ulang method) pada kelas turunannya.

7.7 POST TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-03	CPMK-02	Buatlah kelas tester dari contoh pada Langkah praktikum. Lakukan instansiasi objek untuk objek manusia kemudian berikan nilai untuk tinggi badannya dan hitung tinggi badan ideal objek manusia tersebut.	25
2.	CPL-03	CPMK-02	Lakukan instansiasi objek untuk objek laki-laki kemudian berikan nilai untuk tinggi badannya dan hitung tinggi badan ideal objek laki-laki tersebut.	20
3.	CPL-03	CPMK-02	Lakukan instansiasi objek untuk objek perempuan kemudian berikan nilai untuk tinggi badannya dan hitung tinggi badan ideal objek perempuan tersebut	20
4.	CPL-03	CPMK-02	Buat kelas baru dengan nama PerempuanDewasa sebagai turunan dari kelas perempuan dan lakukan overriding pada method hitungBeratBadanIdeal dengan rumus menghitung berat badan ideal adalah $(\text{tinggi badan(cm)} - 100) \text{ kg} \times 85\%$. Lakukan instansiasi objek untuk objek perempuandewasa kemudian berikan nilai untuk tinggi badannya dan hitung tinggi badan ideal objek perempuandewasa tersebut	35

7.8 HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL-03	CPMK-02	20%		
2.	Praktik	CPL-03	CPMK-02	20%		
3.	Post-Test	CPL-03	CPMK-02	60%		
Total Nilai						

PRAKTIKUM 8: INTERFACE

Pertemuan ke : 8

Total Alokasi Waktu : 90 menit

- Materi : 15 menit
- Pre-Test : 15 menit
- Praktikum : 40 menit
- Post-Test : 20 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 20 %
- Post-Test : 60 %

Pemenuhan CPL dan CPMK:

CPL-03	Mampu menerapkan konsep teoritis bidang area Informatika terkait matematika dasar dan ilmu komputer untuk memodelkan masalah dan meningkatkan produktivitas
CPMK-02	Mahasiswa mampu memahami konsep rancangan dan prinsip encapsulation, information hiding, dan implementasinya

8.1 DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu Menjelaskan kelas interface, contoh kasus kelas interface dan penerapannya

8.2 INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL-03	CPMK-02	1) Mengimplementasikan kelas interface 2) Menulis coding kelas interface
--------	---------	---

8.3 TEORI PENDUKUNG

Interface adalah kumpulan method yang hanya memuat deklarasi dan struktur method tanpa detail implementasinya. Cara mendeklarasikan interface adalah sebagai berikut:.

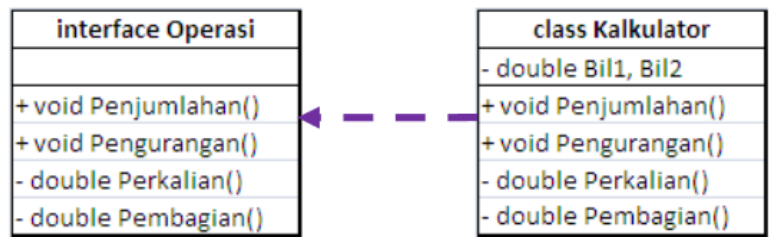
```

1 interface Nama_Interface
2 {
3     //deklarasi variabel dan/atau method
4 }
```

Contoh:

```
1 interface Operasi
2 {
3     //deklarasi variabel dan/atau method
4     public void Penjumlahan();
5     public void Pengurangan();
6     public double Perkalian();
7     public double Pembagian();
8 }
```

Pada contoh di atas, method yang dideklarasikan pada interface Operasi tidak terdapat statement apapun, baik itu rumus atau hanya sebuah nilai balik di dalamnya. Hal ini dikarenakan interface hanyalah sebuah berisi kumpulan konstanta maupun method tanpa menyertakan/menuliskan body methodnya. Perlu diketahui pula, bahwa an interface is not a class and classes can only implement interfaces (sebuah interface bukanlah sebuah kelas dan kelas hanya bisa mengimplementasi interface). Sehingga jangan anda menganggap bahwa interface adalah super class dimana memiliki kelas turunan. Penggunaan (implementasi) interface dalam sebuah kelas dapat anda lihat melalui skema OOP di bawah ini:



Anak panah putus-putus merupakan gambaran bahwa class Kalkulator merupakan implementasi dari interfaces Operasi, dimana method-method yang terdapat pada interface Operasi harus dideklarasikan ulang (overriding method) pada kelas Kalkulator. Interface dilambangkan dengan anak panah dengan garis putus-putus, sedangkan inheritance dilambangkan dengan anak panah dengan garis lurus (—).

Dalam pemrograman OOP, implementasi interfaces menggunakan keyword implements. Berikut adalah cara mengimplementasikan interface ke dalam pemrograman Java:

```
1 class Nama_Kelas implements Nama_Interface
2 {
3     //isi kelas
4 }
```


Contoh implementasi:

```

1 class Kalkulator implements Operasi
2 {
3     public void Penjumlahan()
4     {
5         //isi method Penjumlahan()
6     };
7
8     public void Pengurangan()
9     {
10        //isi method Pengurangan()
11    };
12
13    public double Perkalian()
14    {
15        //isi method Perkalian()
16        return 0;
17    };
18
19    public double Pembagian()
20    {
21        //isi method Pembagian()
22        return 0;
23    };
24 };

```

8.4 HARDWARE DAN SOFTWARE

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. IntelliJ IDEA Community
3. Java SDK

8.5 PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-03	CPMK-02	Apa yang dimaksud Interface dalam PBO?	30
2.	CPL-03	CPMK-02	Jelaskan perbedaan antara Interface dan Abstract Class dalam konteks pemanfaatannya!	70

8.6 LANGKAH PRAKTIKUM

Aturan Penilaian (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL-03	CPMK-02	Selesaikan langkah praktikum 1 –3	Hasil praktikum langkah 1 – 3	100

Langkah-Langkah Praktikum:

1. Buat project prak8 dengan menggunakan IntelliJ IDEA
2. Kerjakan contoh di bawah ini dan amati hasilnya!

```

/*interface berisi abstract method atau method header*/
interface MyComparable {
boolean greaterThan(Object obj); boolean lessThan(Object obj);
boolean equal (Object obj);
}
/*interface berisi konstanta*/
interface Constants {
int min = 1000;
int max = 9999;
}
/*class mengimplementasikan dua interface*/
class FourDigitsNumber implements Constants, MyComparable {
private int value;
public FourDigitsNumber(int initValue) {
/*latihan 1: atur agar nilai yang diinputkan dalam constructor hanya
berada di antara min - max
*/
}
/*latihan 2: tambahkan method get untuk mengakses value*/
/*overriding method greaterThan*/
public boolean greaterThan(Object obj) {
/*casting from superclass to subclass*/ FourDigitsNumber number =
(FourDigitsNumber)obj; return ( value > number.getValue() );
}
/*latihan 3: lengkapi overriding method interface*/
}
/*Contoh penggunaan class*/
class ExInterface {
public static void main(String [] args) { FourDigitsNumber number1 =
new FourDigitsNumber(700); FourDigitsNumber number2 = new
FourDigitsNumber(1700); FourDigitsNumber number3 = new
FourDigitsNumber(70000); System.out.println(number1.getValue());
System.out.println(number2.getValue());
System.out.println(number3.getValue());
System.out.println(number1.greaterThan(number2));
System.out.println(number1.lessThan(number2));
System.out.println(number1.equal(number2));
}}

```

3. Kerjakan soal latihan yang ada dalam contoh di atas!

8.7 POST TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-03	CPMK-02	<p>Buatlah program baru yang menggambarkan implementasi interface Operasi di bawah ini, ke dalam class Calculator agar dapat mengoperasikan 2 bilangan Bil1 dan Bil2 menggunakan method yang ada dalam interface Operasi!</p> <pre> 1 interface Operasi 2 { 3 //deklarasi variabel dan/atau method 4 public void Penjumlahan(); 5 public void Pengurangan(); 6 public double Perkalian(); 7 public double Pembagian(); 8 }</pre>	100

8.8 HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL-03	CPMK-02	20%		
2.	Praktik	CPL-03	CPMK-02	20%		
3.	Post-Test	CPL-03	CPMK-02	60%		
Total Nilai						

PRAKTIKUM 9: EXCEPTION HANDLING

Pertemuan ke : 9

Total Alokasi Waktu : 90 menit

- Materi : 15 menit
- Pre-Test : 15 menit
- Praktikum : 40 menit
- Post-Test : 20 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 20 %
- Post-Test : 60 %

Pemenuhan CPL dan CPMK:

CPL-03	Mampu menerapkan konsep teoritis bidang area Informatika terkait matematika dasar dan ilmu komputer untuk memodelkan masalah dan meningkatkan produktivitas
CPMK-03	Mahasiswa mampu merangkai Modern programming environments

9.1 DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu Setelah mengikuti praktikum ini mahasiswa diharapkan mampu menerapkan exception handling dan menerapkan I/O.

9.2 INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL-01	CPMK-03	kemampuan mahasiswa dalam menerapkan exception handling dan menerapkan I/O.
--------	---------	---

9.3 TEORI PENDUKUNG

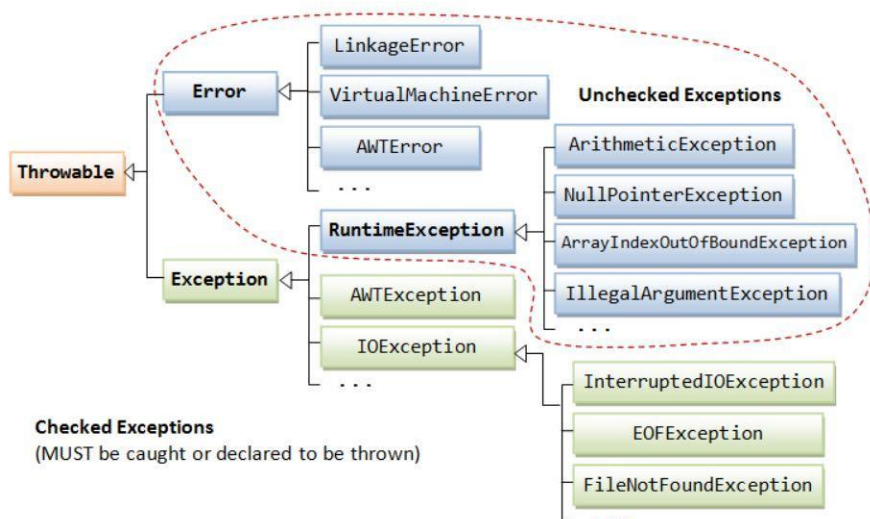
Pada program-program yang membutuhkan data-data eksternal akan diperlukan suatu proses input dan output (I/O). Java mendukung proses I/O ini di dalam paket java.io. Di dalam paket tersebut tersimpan banyak kelas dan interface siap pakai yang akan memudahkan programmer dalam pengambilan dan penyimpanan informasi dari/ke media lain (misalnya file).

Program Java melakukan proses I/O melalui *stream*, yaitu sebuah abstraksi yang dapat memberikan atau mendapatkan informasi. *Stream* dapat dihubungkan dengan peralatan fisik yang terdapat dalam sistem I/O Java, seperti *keyboard*, *file*, layar console, soket jaringan, dan lainnya. Walaupun dihubungkan dengan peralatan fisik yang berbeda, cara kerja *stream* selalu sama, sehingga

kode program yang ditulis juga sama untuk masing-masing peralatan fisik. Misalnya, untuk melakukan penulisan sebuah teks ke layar console maupun ke dalam file, maka dapat digunakan kelas dan *method* yang sama. *Stream* ada dua jenis, yaitu *stream byte* dan *stream karakter*. *Stream byte* digunakan untuk memberikan atau menyimpan informasi data dalam bentuk *byte*. Misalnya untuk menulis dan membaca *file* biner. *Stream karakter* pada proses I/O melibatkan data-data berbentuk karakter. Misalnya proses baca/tulis ke suatu *file* teks, dengan menggunakan karakter Unicode. Pendefinisian *stream* dilakukan dengan menggunakan empat kelas abstrak, yaitu *InputStream* dan *OutputStream*, sebagai *superclass* untuk kelas-kelas dalam kategori *stream byte*, dan kelas abstrak *Reader* dan *Writer* untuk kategori *stream karakter*. Melalui proses pewarisan (*inheritance*), semua kelas yang diturunkan dari *InputStream* maupun *Reader* akan memiliki *method read()*, yang digunakan dalam proses pembacaan data. Adapun untuk proses penulisan data digunakan *method write()* dalam semua kelas yang diturunkan dari *OutputStream* maupun *Writer*.

Exception merupakan *runtime error*. *Exception* terjadi jika program berjalan tidak sesuai dengan yang seharusnya. Artinya, alur program berubah dan aplikasi berhenti dalam keadaan error. Eksepsi dapat terjadi karena tidak ada antisipasi terhadap kesalahan penggunaan program. Misalnya program didesain untuk menghitung angka yang dimasukkan. Jika pengguna memasukkan karakter yang tidak dapat dihitung, misalnya huruf, maka akan terjadi *exception*. Di dalam Java ada mekanisme untuk mengatasi *runtime error* yaitu *exception handling* dan kelas khusus untuk *Exception* yang hierarkinya ditunjukkan pada Gambar 9.1. Terdapat dua jenis *exception* dalam java, yaitu:

1. *Checked Exception*. Merupakan eksepsi yang harus ditangkap atau dilempat secara eksplisit. Jika tidak ada metode yang dikonfigurasi untuk menangkap eksepsi ini, maka kode program tidak dapat dikompilasi
2. *Unchecked Exception*. Merupakan eksepsi yang dapat dibuat agar program benar-benar dapat diandalkan. Namun tidak harus ada.



Gambar 9.1 Hierarki kelas Exception (Sumber: https://www.ntu.edu.sg/home/ehchua/programming/java/J5a_ExceptionAssert.html)

9.4 HARDWARE DAN SOFTWARE

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. IntelliJ IDEA
3. Java SDK

9.5 PRE-TEST

Jawablah pertanyaan berikut (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-03	CPMK-03	Kapan Exception Handling digunakan?	40
2.	CPL-03	CPMK-03	Sintak apa yang digunakan untuk menangani exception dalam pemrograman? Tuliskan dengan contoh exception divide by zero!	60

9.6 LANGKAH PRAKTIKUM

Aturan Penilaian (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL-03	CPMK-03	Selesaikan langkah praktikum 1 – 8	Hasil praktikum langkah 1 – 8	100

Langkah-Langkah Praktikum:

1. Buat project Prak9 dengan menggunakan NetBeans.
2. Buatlah sebuah kelas POJO dengan nama ContohOutput seperti kode pada Gambar 9.2 dan amati manfaat sintaksnya.

```

1 package IO;
2
3 public class ContohOutput {
4     public static void main(String[] args) {
5         int varInt = 45;
6         double varDouble = 7.98845;
7         float varFloat = 6.97f;
8         char varChar = 'X';
9         String varString = "Ini sebuah String";
10
11         System.out.printf("%d\n", varInt);
12         System.out.printf("%e\n", varDouble);
13         System.out.printf("%f\n", varFloat);
14         System.out.printf("%.2f\n", varFloat);
15         System.out.printf("%c\n", varChar);
16         System.out.printf("%.3s\n", varString);
17         System.out.printf("%s\n", varString);
18
19         System.out.println("=====");
20         System.out.print("Ini adalah ");
21         System.out.print("contoh penggunaan print");
22
23         System.out.println();
24         System.out.println("=====");
25         System.out.println("Ini adalah ");
26         System.out.println("contoh penggunaan println");
27     }
28 }

```

Output - Prak9 (run) x

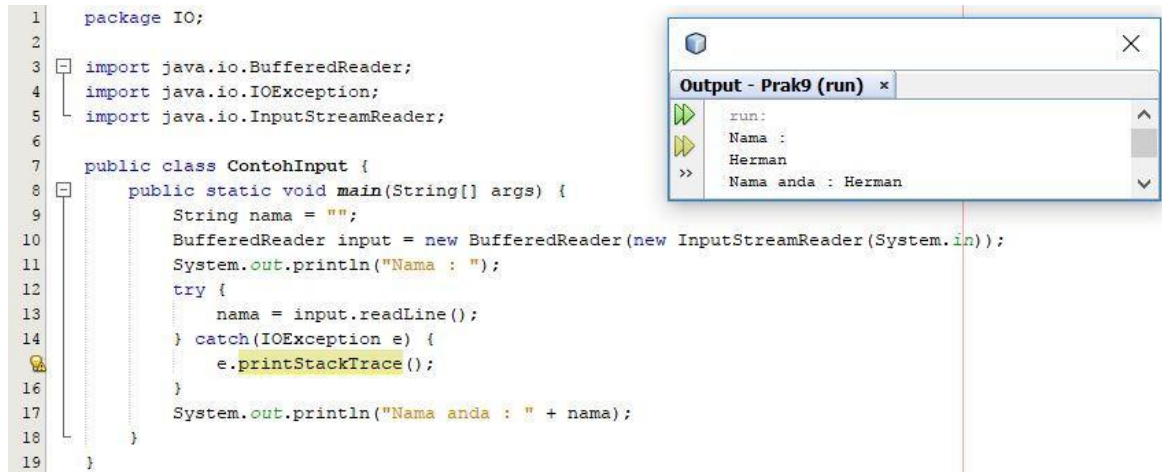
```

run:
45
7.988450e+00
6.970000
6.97
X
Ini
Ini sebuah String
=====
Ini adalah contoh penggunaan print
=====
Ini adalah
contoh penggunaan println

```

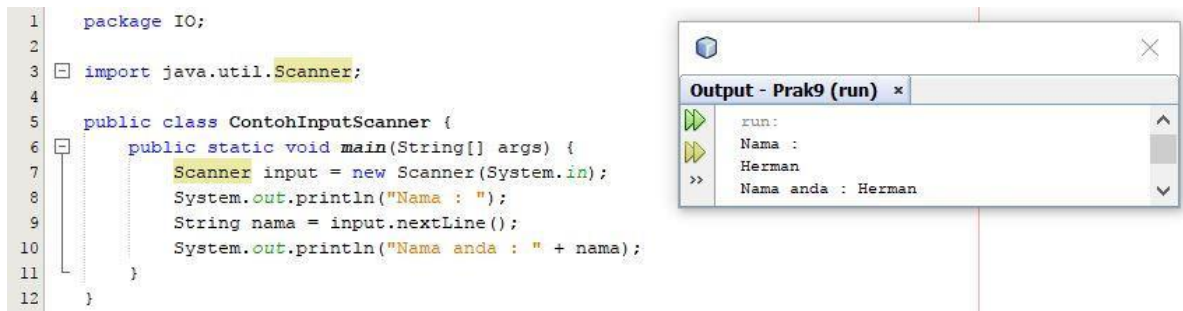
Gambar 9.2 Hasil percobaan 1.

3. Buatlah sebuah kelas POJO dengan nama ContohInput seperti kode pada Gambar 9.3 dan amati manfaat sintaksnya



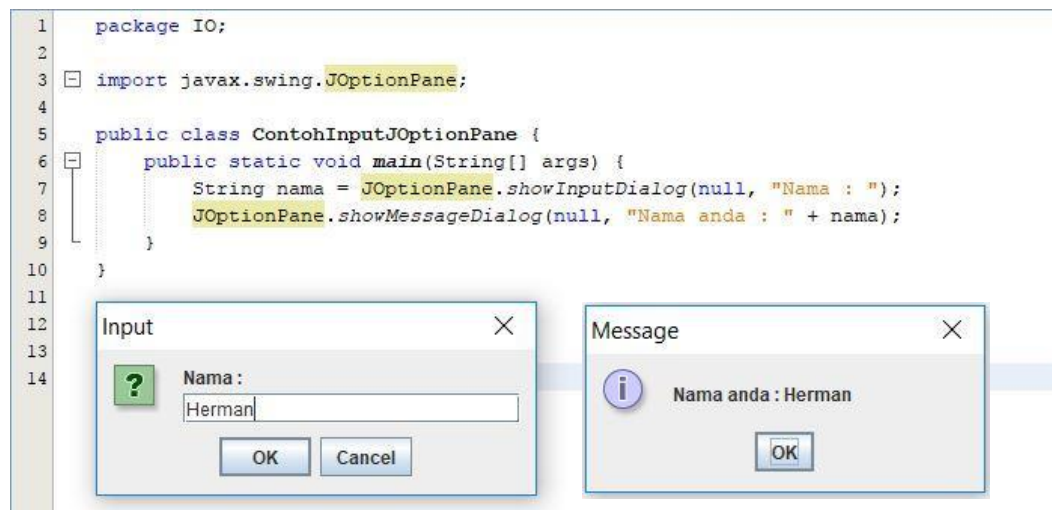
Gambar 9.3 Hasil percobaan 2.

4. Buatlah sebuah kelas POJO dengan nama `ContohInputScanner` seperti kode pada Gambar 9.4 dan amati manfaat sintaksnya.



Gambar 9.4 Hasil percobaan 3.

5. Buatlah sebuah kelas POJO dengan nama `ContohInputJOptionPane` seperti kode pada Gambar 9.5 dan amati manfaat sintaksnya.



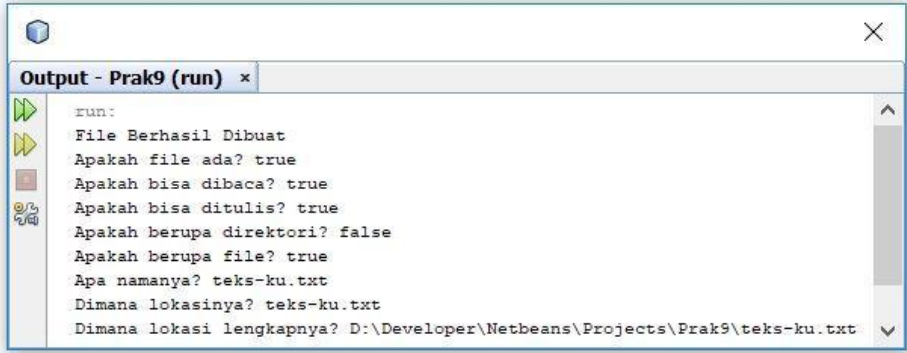
Gambar 9.5 Hasil percobaan 4.

6. Buatlah sebuah kelas POJO dengan nama `ContohClassFile` seperti kode pada Gambar 9.6 dan amati manfaat sintaksnya.


```

1 package IO;
2
3 public class ContohClassFile {
4     public static void main(String[] args) {
5         java.io.File file = new java.io.File("teks-ku.txt");
6         try {
7             if (file.createNewFile())
8                 System.out.println("File Berhasil Dibuat");
9             else
10                System.out.println("File Gagal Dibuat");
11        } catch (Exception e) {
12            System.out.println("Error");
13        }
14
15        System.out.println("Apakah file ada? " + file.exists());
16        System.out.println("Apakah bisa dibaca? " + file.canRead());
17        System.out.println("Apakah bisa ditulis? " + file.canWrite());
18        System.out.println("Apakah berupa direktori? " + file.isDirectory());
19        System.out.println("Apakah berupa file? " + file.isFile());
20        System.out.println("Apa namanya? " + file.getName());
21        System.out.println("Dimana lokasinya? " + file.getPath());
22        System.out.println("Dimana lokasi lengkapnya? " + file.getAbsolutePath());
23    }
24 }
25
26
27

```



Gambar 9.6 Hasil percobaan 5.

7. Buatlah sebuah kelas POJO dengan nama ContohTulisFile seperti kode pada Gambar 9.7 dan amati manfaat sintaksnya.

```

1 package IO;
2
3 import java.io.FileNotFoundException;
4
5 public class ContohTulisFile {
6     public static void main(String[] args) {
7
8         java.io.File file = new java.io.File("teks-ku.txt");
9
10        try {
11            java.io.PrintWriter output = new java.io.PrintWriter(file);
12            output.println("Herman Yuliansyah, S.T., M.Eng.");
13            output.println("Pemrograman Berorientasi Obyek");
14            output.println("Universitas Ahmad Dahlan");
15            output.close();
16        } catch (FileNotFoundException e) {
17            e.printStackTrace();
18        }
19    }
20 }

```



Gambar 9.7 Hasil percobaan 6.

8. Buatlah sebuah kelas POJO dengan nama ContohBacaFile seperti kode pada Gambar 9.8 dan amati manfaat sintaksnya


```

1 package IO;
2
3 import java.io.FileNotFoundException;
4
5 import java.util.Scanner;
6
7 public class ContohBacaFile {
8     public static void main(String[] args) {
9
10        java.io.File file = new java.io.File("teks-ku.txt");
11        try {
12            Scanner input = new Scanner(file);
13            input.useDelimiter("\n");
14            while(input.hasNext()) {
15                String nama = input.next();
16                String mk = input.next();
17                String pt = input.next();
18                System.out.println("Nama : " + nama);
19                System.out.println("Mata Kuliah : " + mk);
20                System.out.println("Perguruan Tinggi : " + pt);
21            }
22            input.close();
23        } catch (FileNotFoundException e) {
24            e.printStackTrace();
25        }
26    }
27 }

```

Output - Prak9 (run) - E...
 Output - Prak9 (run) x
 run:
 Nama : Herman Yuliansyah, S.T., M.Eng.
 Mata Kuliah : Pemrograman Berorientasi Obyek
 Perguruan Tinggi : Universitas Ahmad Dahlan

Gambar 9.8 Hasil percobaan 7.

9.7 POST TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

Di sebuah toko kayu, Anda ingin membeli sebuah balok kayu dengan ukuran yang anda inputkan sendiri. Akan tetapi disana hanya tersedia balok kayu dengan nilai panjang harus lebih besar dari nilai lebar dan nilai lebar harus lebih besar dari nilai tinggi

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-03	CPMK-03	Hitung volume balok kayu yang Anda beli kemudian hasil perhitungannya ditampilkan dilayar dan juga disimpan dalam sebuah file.	50
2.	CPL-03	CPMK-03	Berikan exception handling untuk masukan supaya melakukan pengecekan terhadap kesalahan pengisian nilai panjang, lebar dan tingginya.	50

9.8 HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL-03	CPMK-03	20%		
2.	Praktik	CPL-03	CPMK-03	20%		
3.	Post-Test	CPL-03	CPMK-03	60%		

	Total Nilai
--	--------------------

PRAKTIKUM 10: APLIKASI PEMROGRAMAN BERORIENTASI OBJEK

Pertemuan ke : 10

Total Alokasi Waktu : 90 menit

- Materi : 15 menit
- Pre-Test : 15 menit
- Praktikum : 40 menit
- Post-Test : 20 menit

Total Bobot Penilaian : 100%

- Pre-Test : 20 %
- Praktik : 20 %
- Post-Test : 60 %

Pemenuhan CPL dan CPMK:

CPL-07	Mampu memilih, membuat dan menerapkan teknik, sumber daya, penggunaan perangkat teknik modern dan implementasi teknologi informasi untuk memecahkan masalah
CPMK-01	Mahasiswa mampu merangkai Modern programming environments

10.1 DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu Menjelaskan enviroment development pemrograman games 2 D dan penerapan pemrograman berorientasi objek dengan pengembangan games 2D.

10.2 INDIKATOR KETERCAPAIAN PEMBELAJARAN

Indikator ketercapaian diukur dengan:

CPL-07	CPMK-03	1) Memahami enviroment development pemrograman 2D 2) Mengimplementasikan konsep pemrograman berorientasi objek dalam pemrograman games 2D
--------	---------	--

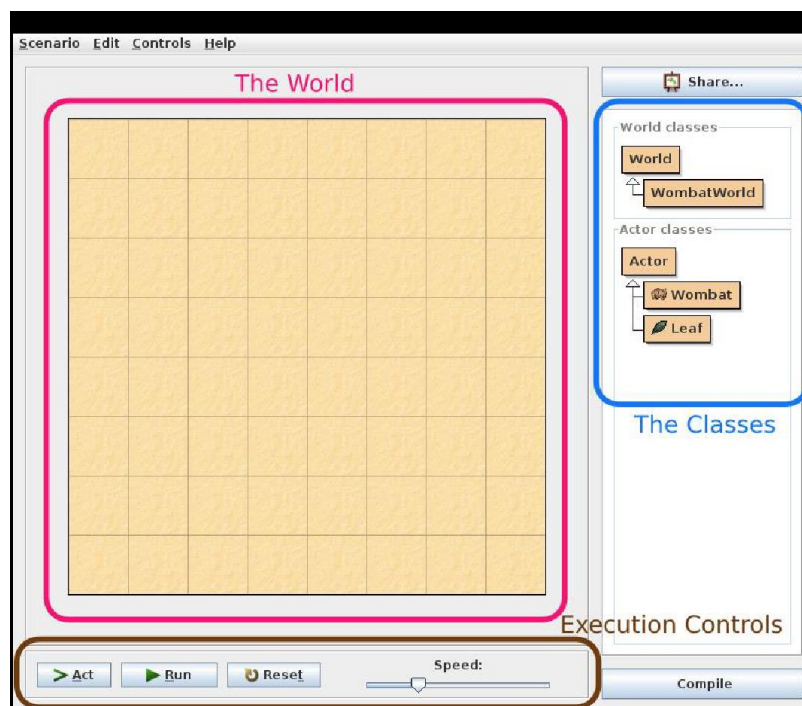
10.3 TEORI PENDUKUNG

Greenfoot merupakan lingkungan pengembangan interaktif Java yang dirancang terutama untuk tujuan pendidikan di sekolah tinggi dan tingkat sarjana. Hal ini memungkinkan pengembangan yang mudah dari aplikasi grafis dua dimensi, seperti simulasi dan permainan interaktif. Greenfoot sedang dikembangkan dan dipelihara di University of Kent, dengan dukungan dari Oracle. Ini adalah perangkat lunak bebas, dirilis di bawah lisensi GPL. Greenfoot tersedia untuk Windows, OS X, Linux, Solaris, dan setiap JVM. Proyek Greenfoot ini diprakarsai oleh Michael Kölling pada tahun 2003, dan prototipe

pertama dibangun oleh Poul Henriksen (mahasiswa master) dan Michael Kölling (supervisor) pada tahun 2003/2004. Dari tahun 2005 pembangunan dilanjutkan melibatkan anggota lain dari Grup BlueJ di University of Kent dan Deakin University.

Model pemrograman Greenfoot terdiri dari kelas World (diwakili oleh area layar persegi panjang) dan sejumlah objek Actor yang hadir di World dan dapat diprogram untuk bertindak independen. World dan Actor yang diwakili oleh objek Java dan didefinisikan oleh kelas Java. Greenfoot menawarkan metode untuk dengan mudah memprogram para Actor ini, termasuk metode untuk gerakan, rotasi, perubahan penampilan, tabrakan, dll. Pemrograman di Greenfoot di dasar yang paling terdiri dari subclassing dua built-in kelas, World dan Actor seperti pada Gambar 10.1. Sebuah contoh dari subclass dunia mewakili World di mana eksekusi Greenfoot akan terjadi. subclass Actor adalah objek yang bisa eksis dan bertindak di World. Sebuah contoh dari subclass dunia secara otomatis dibuat oleh lingkungan.

Eksekusi di Greenfoot terdiri dari loop utama built-in yang berulang kali memanggil metode tindakan masing-masing Actor. Pemrograman skenario, oleh karena itu, sebagian besar terdiri dari menerapkan metode tindakan bagi Actor skenario ini. Implementasi dilakukan di Java standar. Greenfoot menawarkan metode API untuk berbagai tugas umum, seperti animasi, suara, pengacakan, dan manipulasi gambar. Semua Java perpustakaan standar dapat digunakan juga, dan fungsi canggih dapat dicapai.



Gambar 10.1 Interface Greenfoot.

Greenfoot bertujuan untuk memotivasi peserta didik cepat dengan menyediakan akses mudah ke grafis animasi, suara dan interaksi. Lingkungan sangat interaktif dan mendorong eksplorasi dan eksperimen. Pedagogis, desain didasarkan pada pendekatan konstruktivis dan magang. Kedua, lingkungan dirancang untuk menggambarkan dan menekankan abstraksi penting dan konsep pemrograman berorientasi objek. Konsep seperti hubungan kelas/objek, metode, parameter, dan interaksi objek yang disampaikan melalui visualisasi dan interaksi dipandu. Tujuannya adalah untuk membangun dan mendukung model mental yang benar merupakan sistem pemrograman berorientasi objek modern.

10.4 HARDWARE DAN SOFTWARE

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. Greenfoot <https://www.greenfoot.org>.

10.5 PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-07	CPMK-03	Sebutkan Environment yang harus disiapkan untuk membangun game 2D dengan Bahasa pemrograman Java	100

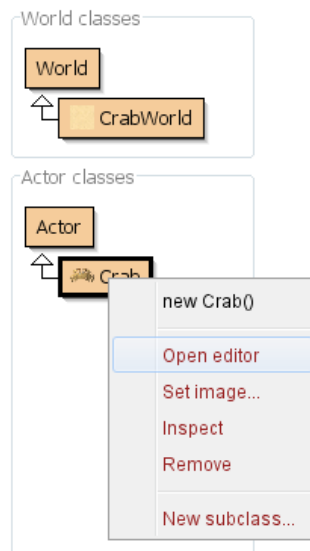
10.6 LANGKAH PRAKTIKUM

Aturan Penilaian (**Total Skor: 100**):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL-07	CPMK-03	Selesaikan langkah praktikum 1 – 11	Hasil praktikum langkah 1 – 11	100

Langkah-Langkah Praktikum:

1. Download contoh scenario greenfoot di url: <http://www.greenfoot.org/tutorial-files/modern-crab.zip>
2. Buka Aktor Crab dengan klik kanan pilih open editor seperti pada Gambar 10.2.



Gambar 10.2 Buka editor pada Aktor Crab.

3. Lengkapi class Crab menjadi seperti pada Gambar 10.3.

```
Class Edit Tools Options
Compile Undo Cut Copy Paste Find... Close Source Code
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

public class Crab extends Actor
{
    /**
     * Act - do whatever the Crab wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        moveAndTurn();
        eat();
    }

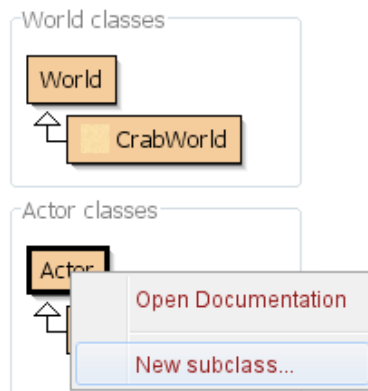
    public void moveAndTurn()
    {
        move(4);

        if (Greenfoot.isKeyDown("left"))
        {
            turn(-3);
        }
        if (Greenfoot.isKeyDown("right"))
        {
            turn(3);
        }
    }

    public void eat()
    {
        Actor worm;
        worm = getOneObjectAtOffset(0, 0, Worm.class);
        if (worm != null)
        {
            World world;
            world = getWorld();
            world.removeObject(worm);
        }
    }
}
saved
```

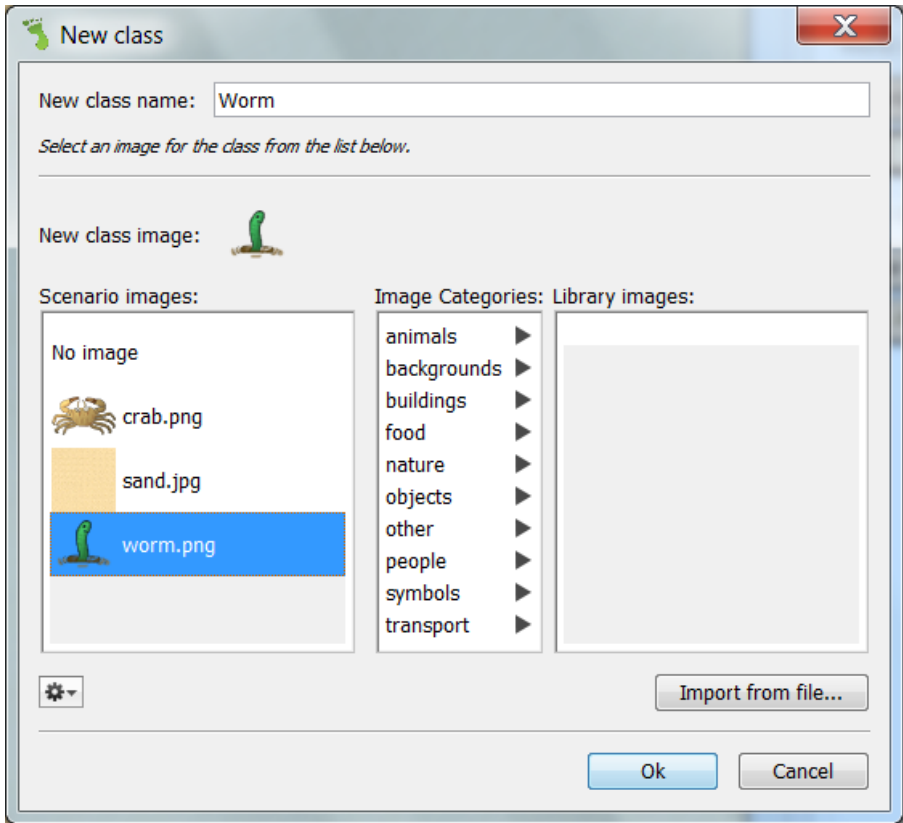
Gambar 10.3 Kode perilaku untuk Aktor Crab.

- 4. Buat sebuah class dengan nama Worm dengan cara klik kanan pada kelas actor dan pilih New subclass seperti pada Gambar 10.4.



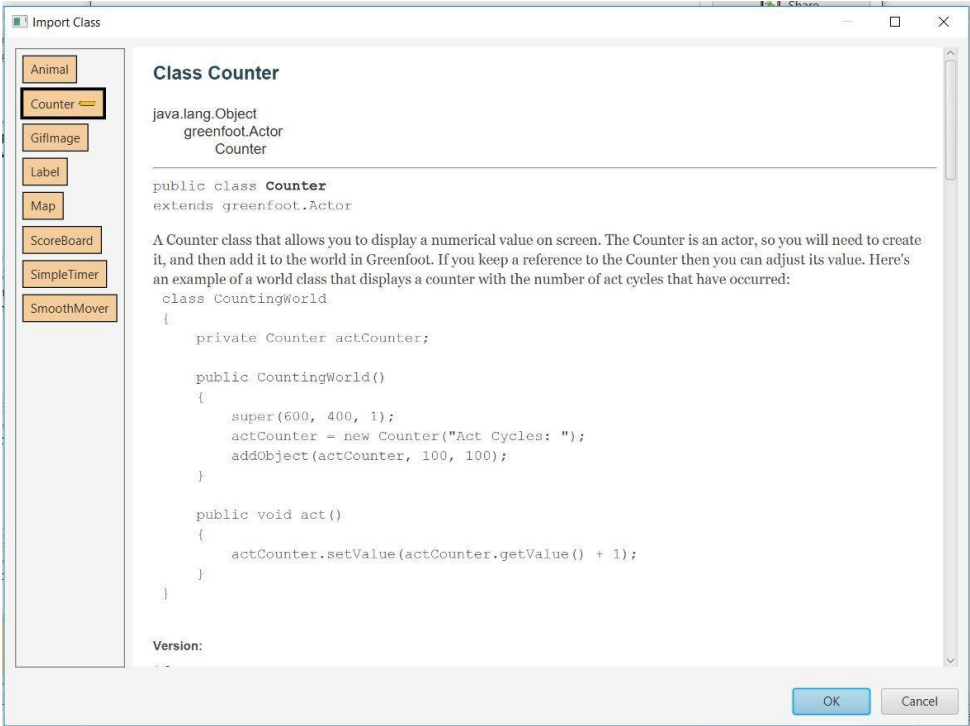
Gambar 10.4 Buat Aktor Worm.

- 5. Pilih image worm sesuai dengan visualisasinya kemudian pilih tombol OK seperti Gambar 10.5.



Gambar 10.5 Pilih tampilan objek Worm.

- 6. Buatlah sebuah counter untuk menampilkan score. Klik Edit – Import Class – Counter – OK seperti pada Gambar 10.6.



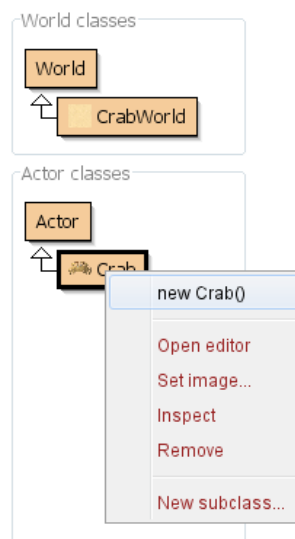
Gambar 10.6 Import class counter.

- 7. Klik kanan Aktor Crab lalu tambahkan kode addScore() seperti Gambar 10.7 untuk meng-update skor pada Counter.

```
public void eat() {  
    Actor worm;  
    worm = getOneObjectAtOffset(0,0,Worm.class);  
    if (worm != null) {  
        World world;  
        world = getWorld();  
        world.removeObject(worm);  
        addScore();  
    }  
}  
  
public void addScore()  
{  
    if (!getWorld().getObjects(Counter.class).isEmpty())  
    {  
        ((Counter) getWorld().getObjects(Counter.class).get(0)).add(1);  
    }  
}
```

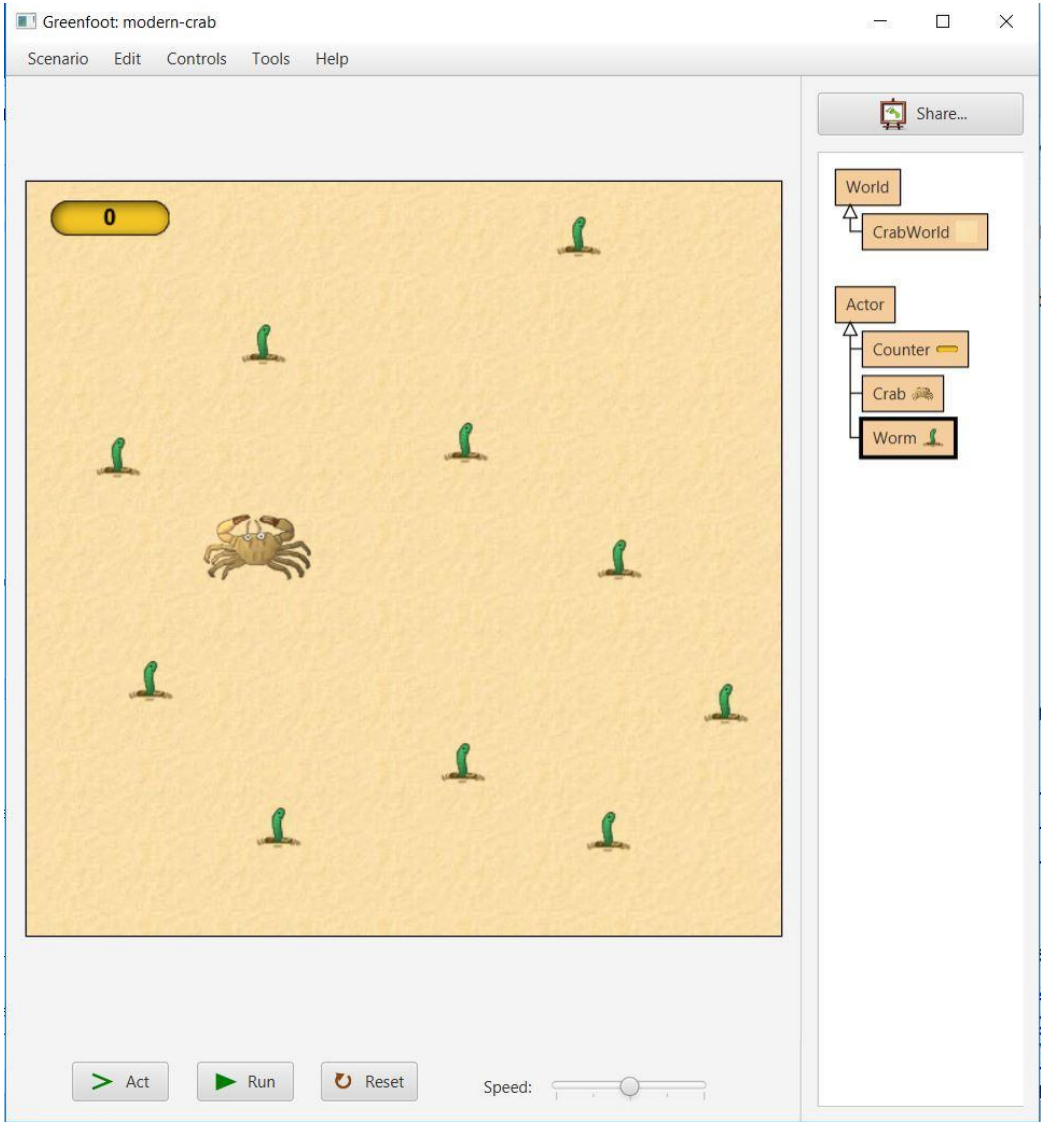
Gambar 10.7 Kode untuk meng-update Counter.

- 8. Buatlah beberapa object Worm, satu object Crab dan satu object Counter kemudian letakkan di World dengan cara klik kanan pada class Worm dan Class Crab, pilih new Crab() untuk membuat objek Crab, pilih new Worm() untuk membuat objek Worm, dan pilih new Counter() untuk membuat objek Counter seperti pada Gambar 10.8.. Letakkan keduanya di World.



Gambar 10.8 Letakkan Aktor Crab di World.

- 9. Letakkan beberapa Worm, satu Counter, dan satu Crab sehingga akan tertampil seperti Gambar 10.9.



Gambar 10.9 Tampilan Crab dan Worms di World.

- 10. Klik kanan pada World kemudian pilih Save the World untuk menyimpan sebaran objek yang telah dibuat.
- Pilih Tombol Run untuk menjalankan greenfoot dan gunakan panah pada keyboard untuk mengatur gerak dari Crab.

10.7 POST TEST

Jawablah pertanyaan berikut (Total Skor: 100):

Tambahkan skenario untuk game diatas dengan perilaku sebagai berikut:

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-07	CPMK-03	1. Crab bisa bergerak cepat sejauh 3x lipat gerakan normal (Dash) apabila ditekan tombol Up	30
2.	CPL-07	CPMK-03	2. Worm yang didekati Crab akan berpindah tempat secara acak apabila Crab mendekatinya dalam jarak 50. Gunakan getObjectsInRange() untuk mengecek jarak antara Crab dengan Worm dan gunakan	35

			Greenfoot.getRandomNumber() untuk membangkitkan angka acak untuk digunakan saat perpindahan tempat	
3.	CPL-07	CPMK-03	3. Hanya ada satu cara si Crab memakan Worm yaitu dengan bergerak kencang (Dash) ke arah Worm sebelum Worm itu sempat berpindah tempat.	35

10.8 HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL-07	CPMK-03	20%		
2.	Praktik	CPL-07	CPMK-03	20%		
3.	Post-Test	CPL-07	CPMK-03	60%		
Total Nilai						

DAFTAR PUSTAKA

- Brown, B. (2005). *A Guide to Programming in Java*. Lawrenceville Press.
- Deitel, P., & Deitel, H. (2012). *Java How to Program 9th. Journal of Chemical Information and Modeling* (Vol. 53). Pearson.
- Schildt, H. (2007). *The Complete Reference Java Seventh Edition*. Mc Graw Hill.
<https://doi.org/10.1192/bjp.112.483.211-a>



**LABORATORIUM
S1 INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS AHMAD DAHLAN**



2023