PP/018/III/R2

LABORATORIUM
TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS AHMAD DAHLAN

# PETUNJUK PRAKTIKUM

## PEMROGRAMAN BERORIENTASI OBJEK

**Penyusun:**
**Drs. Tedy Setiadi, M.T.**
**Ali Tarmuji, S.T., M.Cs.**
**Supriyanto, S.T., M.T.**
**Adhi Prahara, S.Si., M.Cs.**

2020

# HAK CIPTA

## PETUNJUK PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK



**Diterbitkan oleh:**
**Program Studi Teknik Informatika**
Fakultas Teknologi Industri
Universitas Ahmad Dahlan
Jalan Ring Road Selatan, Tamanan, Banguntapan, Bantul Yogyakarta 55166

| | |
|---|---|
| **Penulis** | : Drs. Tedy Setiadi, M.T.<br>Ali Tarmuji, S.T., M.Cs.<br>Supriyanto, S.T., M.T.<br>Adhi Prahara, S.Si., M.Cs. |
| **Editor** | : Laboratorium Teknik Informatika, Universitas Ahmad Dahlan |
| **Desain sampul** | : Laboratorium Teknik Informatika, Universitas Ahmad Dahlan |
| **Tata letak** | : Laboratorium Teknik Informatika, Universitas Ahmad Dahlan |
| **Ukuran/Halaman** | : 21 x 29,7 cm / 65 halaman |

**Didistribusikan oleh:**

**Laboratorium Teknik Informatika**
Universitas Ahmad Dahlan
Jalan Ring Road Selatan, Tamanan, Banguntapan, Bantul Yogyakarta 55166
Indonesia

# KATA PENGANTAR

Puji syukur kami panjatkan kehadirat 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, Agustus 2020

Penyusun

# DAFTAR PENYUSUN

Drs. Tedy Setiadi, M.T.

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

Supriyanto, S.T., M.T.

Adhi Prahara, S.Si., M.Cs.

# HALAMAN REVISI

Yang bertanda tangan di bawah ini:

Nama : Drs. Tedy Setiadi, M.T.

NIY : 60030475

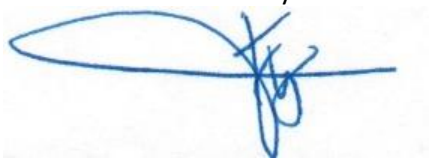
Jabatan : Koordinator Mata Kuliah Pemrograman Berorientasi Objek

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

| No | Keterangan Detail Revisi (Per Pertemuan) | Tanggal Revisi | Nomor Modul |
|---|---|---|---|
| 1 | a. Mengubah susunan materi praktikum yang sebelumnya 12 pertemuan menjadi 10 pertemuan.<br>b. Materi pertama merupakan gabungan dari materi 1 dan 2 pada modul edisi sebelumnya. Yaitu pengenalan java dan IDE dilengkapi dengan dasar pemrograman.<br>c. Materi ke 11 pada modul edisi sebelumnya juga dihilangkan. | 22 Agustus 2019 | PP/018/III/R2 |

Yogyakarta, 28 Agustus 2020

Koordinator Penyusun



**Drs., Tedy Setiadi, M.T.**

NIY. 60030475

# HALAMAN PERNYATAAN

Yang bertanda tangan di bawah ini:

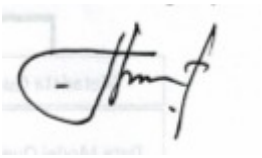
Nama : Lisna Zahrotun, S.T., M.Cs.

NIY : 60150773

Jabatan : Kepala Laboratorium Praktikum Teknik Informatika

Menerangkan dengan sesungguhnya bahwa Petunjuk Praktikum ini telah direview dan akan digunakan untuk pelaksanaan praktikum di Semester Gasal Tahun Akademik 2020/2021 di Laboratorium Praktikum Tenik Informatika, Program Studi Teknik Informatika, Fakultas Teknologi Industri, Universitas Ahmad Dahlan.

Yogyakarta, 28 Agustus 2020

| Mengetahui,<br>Ketua Kelompok Keilmuan Rekayasa Perangkat Lunak dan Data (RELATA) | Kepala Laboratorium Praktikum Teknik Informatika |
|---|---|
|  |  |
| **Drs., Tedy Setiadi, M.T.** | **Lisna Zahrotun, S.T., M.Cs.** |
| NIY. 60030475 | NIY. 60150773 |

# VISI DAN MISI PRODI TEKNIK INFORMATIKA

## VISI

Menjadi Program Studi Informatika yang diakui secara internasional dan unggul dalam bidang Informatika serta berbasis nilai-nilai Islam.

## MISI

1. Menjalankan pendidikan sesuai dengan kompetensi bidang Informatika yang diakui nasional dan internasional
2. Meningkatkan penelitian dosen dan mahasiswa dalam bidang Informatika yang kreatif, inovatif dan tepat guna.
3. Meningkatkan kuantitas dan kualitas publikasi ilmiah tingkat nasional dan internasional
4. Melaksanakan dan meningkatkan kegiatan pengabdian masyarakat oleh dosen dan mahasiswa dalam bidang Informatika.
5. Menyelenggarakan aktivitas yang mendukung pengembangan program studi dengan melibatkan dosen dan mahasiswa.
6. Menyelenggarakan kerja sama dengan lembaga tingkat nasional dan internasional.
7. Menciptakan kehidupan Islami di lingkungan program studi.

# TATA TERTIB LABORATORIUM TEKNIK INFORMATIKA

## DOSEN/KOORDINATOR PRAKTIKUM

1. Dosen harus hadir saat praktikum minimal 15 menit di awal kegiatan praktikum 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 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. 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).
5. Praktikan dilarang meninggalkan kegiatan praktikum tanpa seizin Asisten atau Laboran.
6. Praktikan harus meletakkan sepatu dan tas pada rak/loker yang telah disediakan.
7. Selama praktikum dilarang NGENET/NGE-GAME, kecuali mata praktikum yang membutuhkan atau menggunakan fasilitas Internet.
8. Praktikan dilarang melepas kabel jaringan atau kabel power praktikum tanpa sepengetahuan laboran
9. Praktikan harus memiliki FILE Petunjuk praktikum dan digunakan pada saat praktikum dan harus siap sebelum praktikum berlangsung.
10. Praktikan dilarang melakukan kecurangan seperti mencontek atau menyalin pekerjaan praktikan yang lain saat praktikum berlangsung atau post-test yang menjadi tugas praktikum.

11. 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.
12. 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.
13. 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 harus dapat bertindak secara profesional sebagai seorang asisten praktikum dan dapat menjadi teladan bagi praktikan.
7. Asisten harus dapat memberikan penjelasan/pemahaman yang dibutuhkan oleh praktikan berkenaan dengan materi praktikum yang diasisteni sehingga praktikan dapat melaksanakan dan mengerjakan tugas praktikum dengan baik dan jelas.
8. Asisten tidak diperkenankan mengobrol sendiri apalagi sampai membuat gaduh.
9. Asisten dimohon mengkoordinasikan untuk meminta praktikan agar mematikan komputer untuk jadwal terakhir dan sudah dilakukan penilaian terhadap hasil kerja praktikan.
10. Asisten wajib untuk mematikan LCD Projector dan komputer asisten/praktikan apabila tidak digunakan.
11. Asisten tidak diperkenankan menggunakan akses internet selain untuk kegiatan praktikum, seperti Youtube/Game/Medsos/Streaming Film di komputer praktikan.

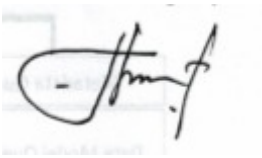
## LAIN-LAIN

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. Izin praktikum mengikuti aturan izin SIMERU/KULIAH.
4. Yang tidak berkepentingan dengan praktikum dilarang mengganggu praktikan atau membuat keributan/kegaduhan.

5. Penggunaan lab diluar jam praktikum maksimal sampai pukul 21.00 dengan menunjukkan surat ijin dari Kepala Laboratorium Prodi Teknik Informatika.

Yogyakarta, 28 Agustus 2020

Kepala Laboratorium Praktikum Teknik Informatika



**Lisna Zahrotun, S.T., M.Cs.**
NIY. 60150773

# DAFTAR ISI

# DAFTAR GAMBAR

# SKENARIO PRAKTIKUM SECARA DARING

Nama Mata Praktikum : Pemrograman Berbasis Objek
Jumlah Pertemuan : 10

**Tabel Skenario Praktikum Daring**

| Pertemuan ke | Judul Materi | Waktu (Lama praktikum sampai pengumpulan posttest) | Skenario Praktikum dari pemberian preteset, posttest dan pengumpulannya serta mencantumkan metode yang digunakan misal video, whatsapp group, Google meet atau lainnya |
|---|---|---|---|
| 1 | Dasar Bahasa Java & IDE | 1 pekan | WAG, New Elearning, Video Tutorial |
| 2 | Class dan Object pada Java | 1 pekan | WAG, New Elearning, Video Tutorial |
| 3 | Konstruktor, Enkapsulasi & Hidding Information | 1 pekan | WAG, New Elearning, Video Tutorial |
| 4 | Pewarisan (Inheritance) | 1 pekan | WAG, New Elearning, Video Tutorial |
| 5 | Polymorphisme | 1 pekan | WAG, New Elearning, Video Tutorial |
| 6 | Relasi Antar Class | 1 pekan | WAG, New Elearning, Video Tutorial |
| 7 | Abstract Class | 1 pekan | WAG, New Elearning, Video Tutorial |
| 8 | Interface | 1 pekan | WAG, New Elearning, Video Tutorial |
| 9 | Exception Handling dan I/O | 1 pekan | WAG, New Elearning, Video Tutorial |
| 10 | Aplikasi Game | 1 pekan | WAG, New Elearning, Video Tutorial |

# PRAKTIKUM 1: DASAR BAHASA JAVA & IDE

Pertemuan ke : 1

Total Alokasi Waktu : 90 menit
- Pre-Test : 15 menit
- Praktikum : 55 menit
- Post-Test : 20 menit

Total Skor Penilaian : 100%
- Pre-Test : 20 %
- Praktikum : 40 %
- Post-Test : 40 %

## 1.1. TUJUAN DAN INDIKATOR CAPAIAN

Setelah mengikuti praktikum ini mahasiswa diharapkan:
1. mampu memahami paradigma pemrograman berientasi objek
2. mampu menjelaskan teknologi yang berkaitan dengan implementasi pemrograman berorientasi objek
3. mampu menjelaskan cara kerja compiler
4. mampu menjelaskan kembali materi: struktur kendali, perulangan, dan array dalam program sederhana

Indikator ketercapaian diukur dengan:
1. mampu memahami paradigma pemrograman berientasi objek
2. mampu menggunakan IDE (Integrated Development Environmet) untuk mengimplementasikan pemrograman berorientasi objek
3. mampu memahami cara kerja compiler
4. mampu mengimplementasikan struktur kendali, perulangan, dan array dalam program sederhana

## 1.2. 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 Microsystem 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 dicompile 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, concurent, 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;
```

Setelah membahas tentang variabel, kita lanjut pengenalan operator di java. Operator di Java tidak beda dengan di C++. Berikut beberapa operator dalam pemrograman Java.

**Operator**

Operator merupakan sebuah karakter khusus yang digunakan untuk menghasilkan suatu nilai. Berikut beberapa macam operator yang ada dalam java.

**Operator Logika**

| Operator | Keterangan |
|---|---|
| + | Penjumlahan |
| - | Pembagian |
| * | Perkalian |
| / | Pembagian |
| % | Sisa |

**Operator Penugasan**

| Operator | Keterangan |
|---|---|
| = | Pemberian nilai |
| += | Penambahan bilangan |
| -= | Pengurangan bilangan |
| *= | Perkalian bilangan |
| /= | Pembagian bilangan |
| %= | Pemerolehan sisa bagi |

**Operator Pembanding**

| Operator | Keterangan |
|---|---|
| == | sama |
| != | Tidak sama |
| >= | Lebih dari sama dengan |
| <= | Kurang dari sama dengan |
| > | Lebih dari |
| < | Kurang dari |

**Operator Logika**

| Operator | Keterangan |
|---|---|
| && | Dan |
| \|\| | Atau |

**STRUKUR PERCABANGAN PADA JAVA**

**Percabangan IF...ESLE**

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

**Percabangan SWITCH**

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

**Percabangan dengan Ekspresi bersyarat**

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

| Sintaks ekspresi bersyarat | Contoh : |
|---|---|
| `exp1 ? exp2 : exp3`<br><br>*Dalam hal ini, jika exp1 adalah benar, exp2 dieksekusi.Jika exp2 adalah salah, exp3 dieksekusi.* | `int a = 50;`<br>`int b = 100;`<br>`int nilai = 3 > 2 ? a : b;`<br>`Outputnya : nilai = 50` |

**PERULANGAN PADA JAVA**

**Perulangan for**

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

**Perulangan while**

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

**Perulangan do....while**

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

**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 eleman 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.

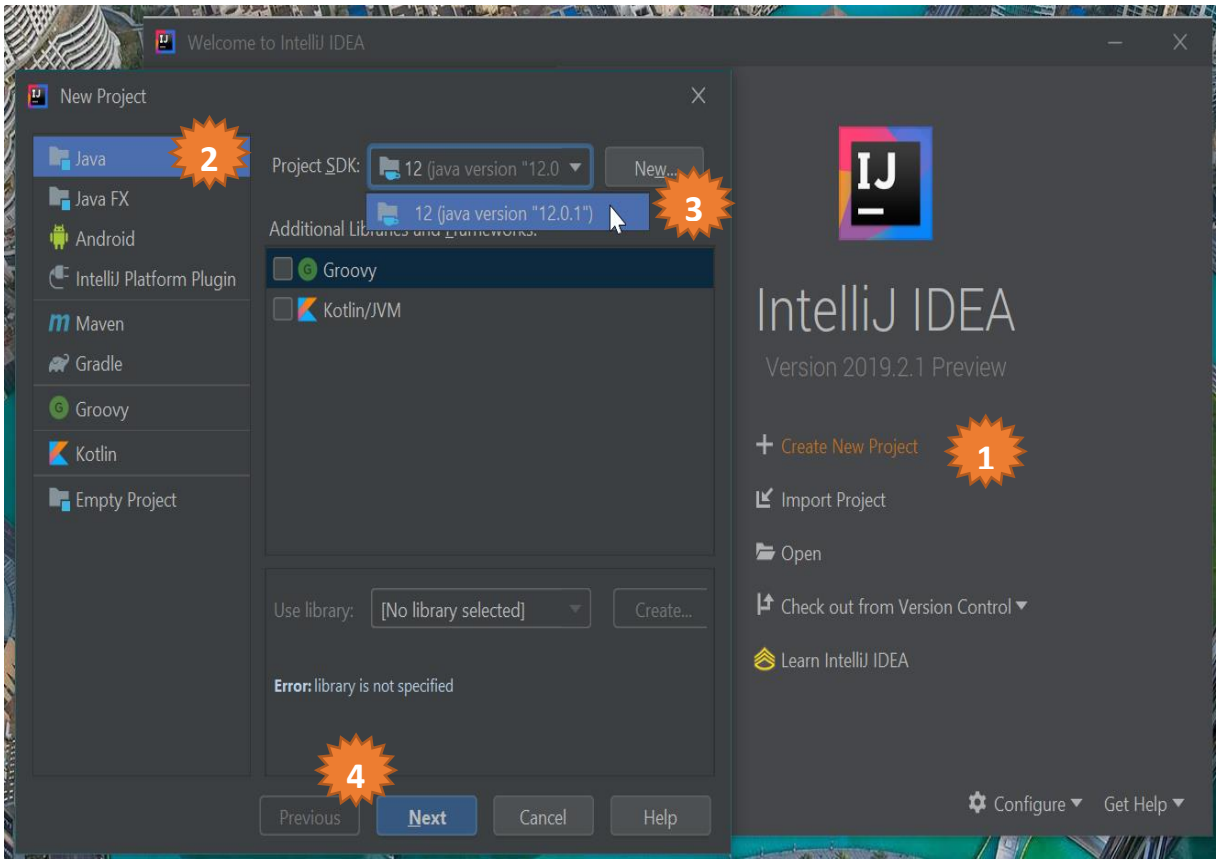
## 1.3. ALAT DAN BAHAN

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

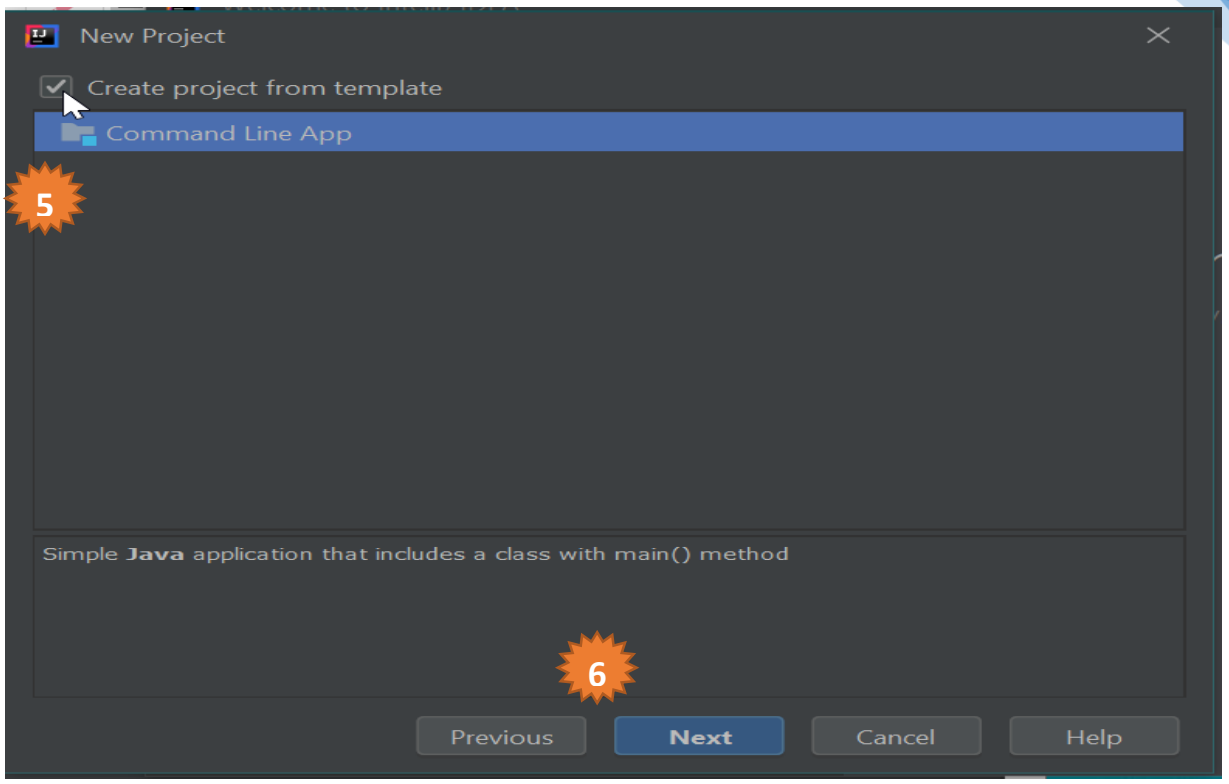
1. Komputer.
2. Intellij IDEA Community
3. Java SDK

## 1.4. LANGKAH PRAKTIKUM

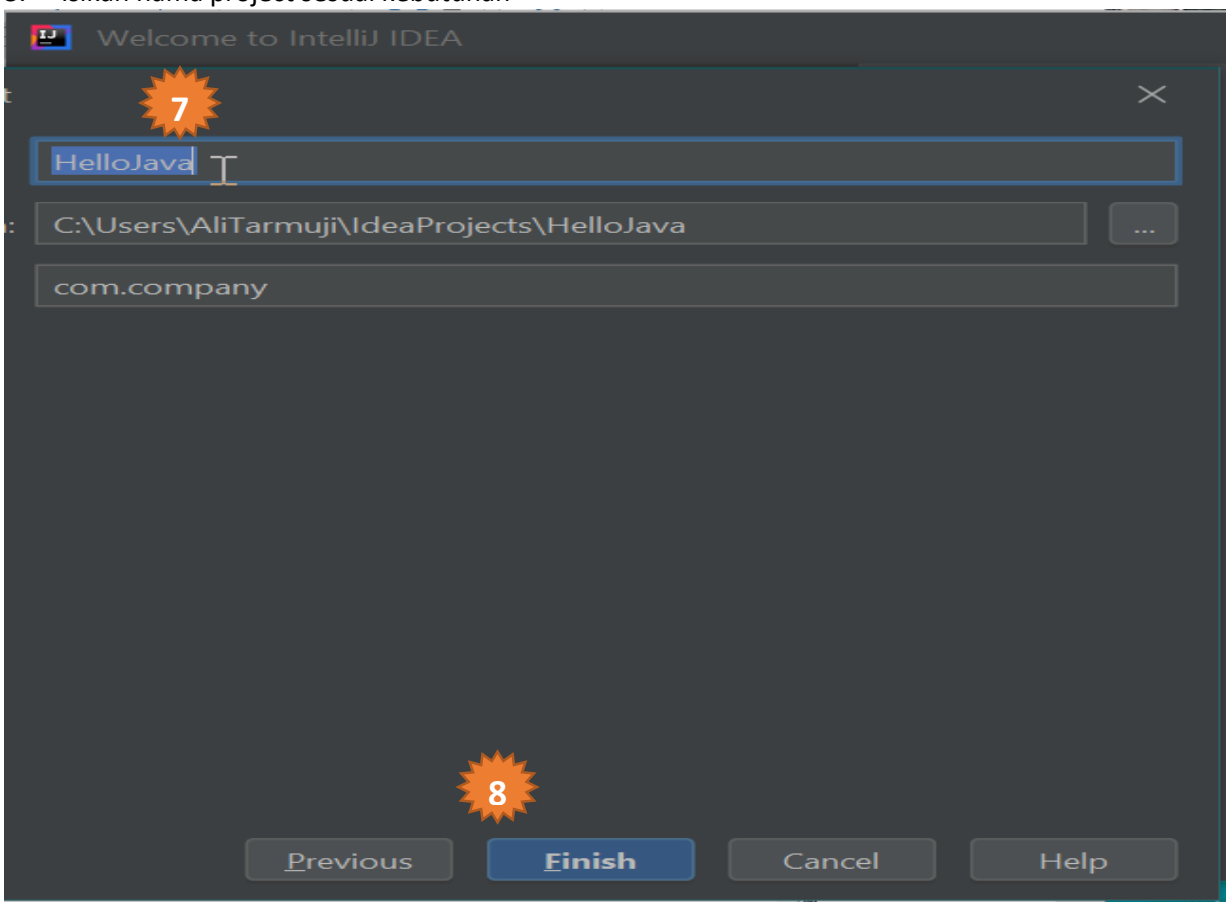
1. Memulai pemrograman Java dengan aplikasi IDE **Intellij IDEA Community** (pastikan aplikasi sudah terpasang di komputer, jika belum bisa didownload di: http://bit.ly/intellijideacommunity) Klik **Create New Project**, Klik **Java**, pilih **Project SDK**: SDK Java (versi diusahakan terbaru), klik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



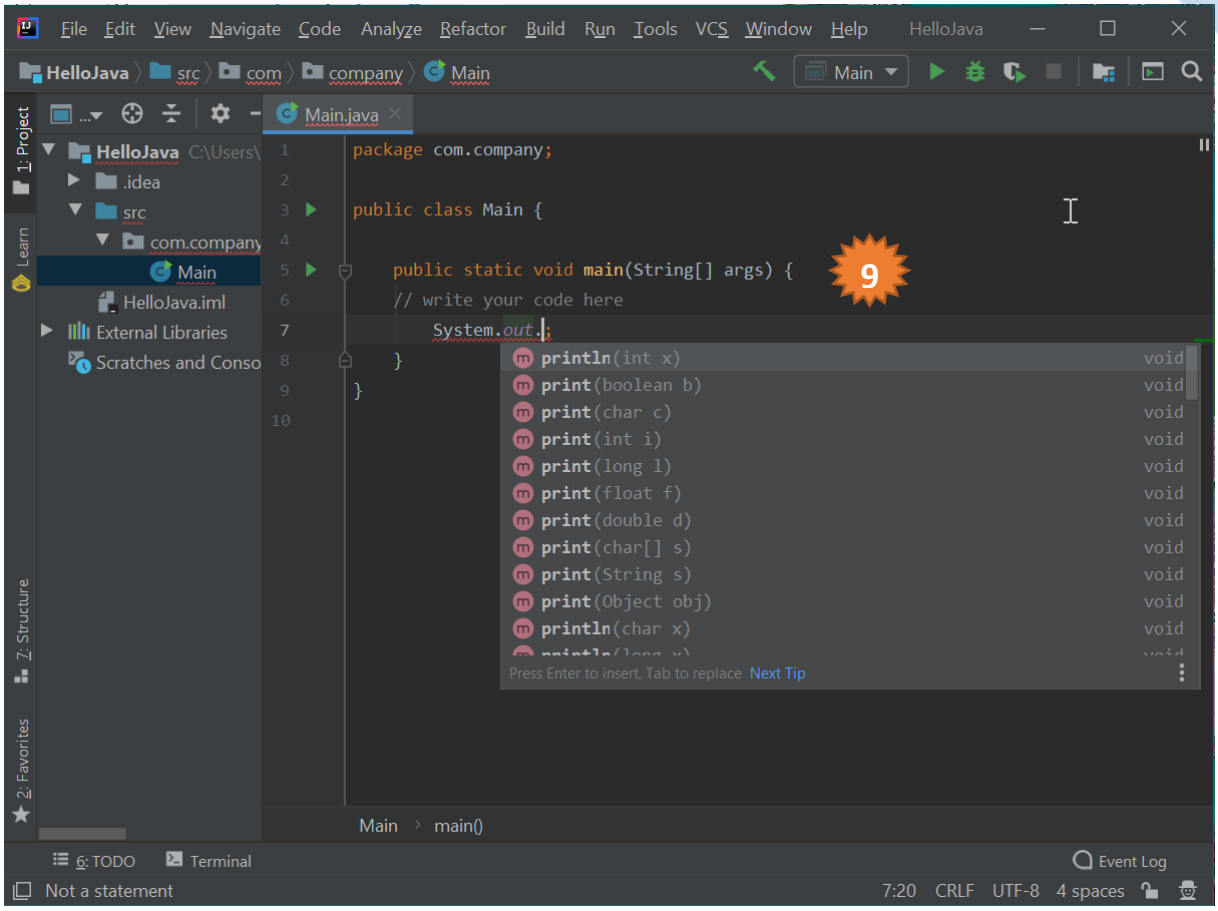

2. Aktifkan generate coding dengan memilih template (centrang)



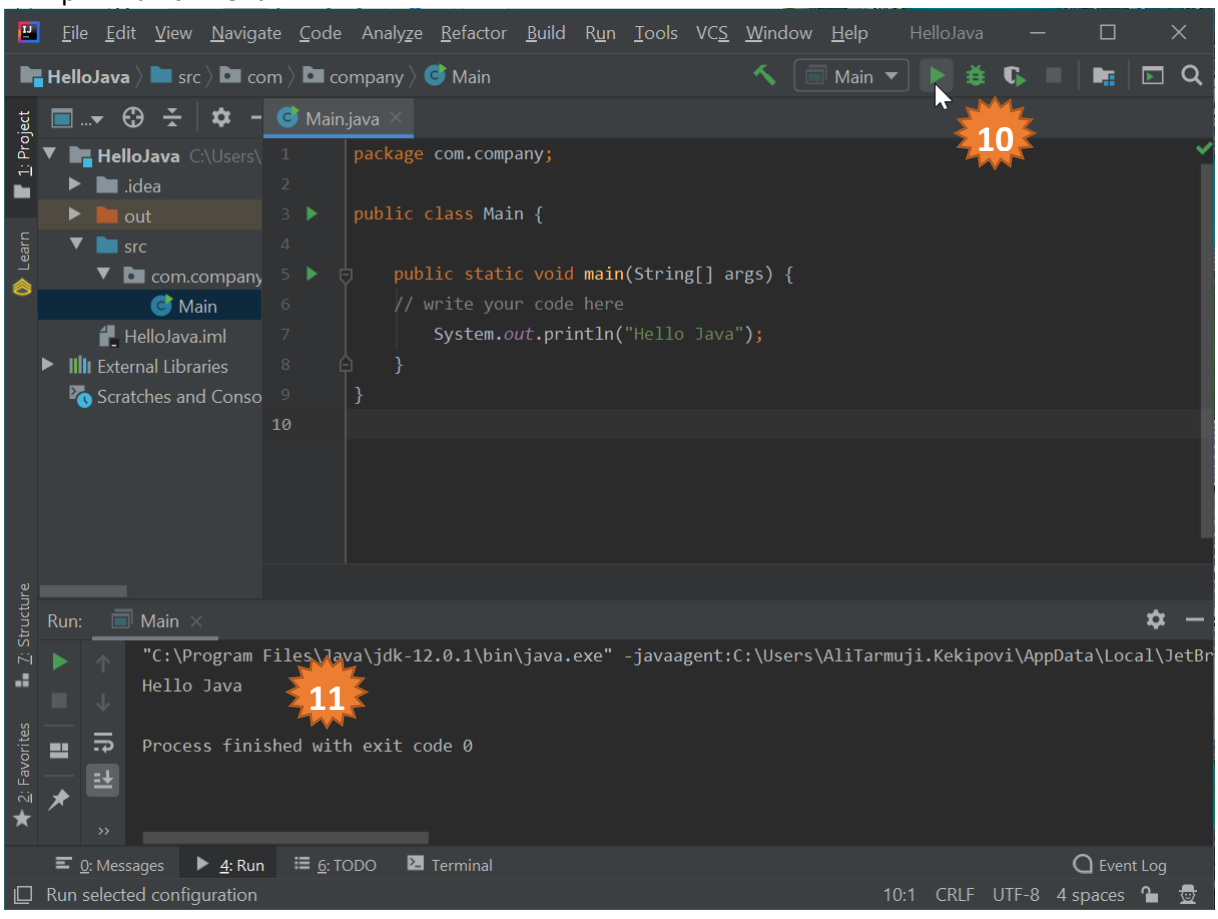

*3.* Isikan nama project sesuai kebutuhan




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



## 1.5. TUGAS

Buatlah program untuk

1. Menampilkan kalimat hello world sampai 100 baris.
2. Membaca jari-jari lingkaran kemudian menampilkan keliling dan luas lingkaran
3. Membaca tiga bilangan bulat kemudian menampilkan bilangan bulat terbesarnya dan terkecilnya
4. Membaca nilai angka kemudian menampilka n nilai hurifnya (A:80-100, B:65-79, C:55-64, D:40-54, E :0-53)
5. Buatlah sebuah program dengan menggunakan array untuk menampilkan bilangan prima antara 1 s.d. 100!
6. Buatlah sebuah program dengan menggunakan array untuk membaca N buah data bilangan bulat positif N, kemudian menampilkan
   a. Jumlah total
   b. Rata-rata
   c. Nilai tertinggi
   d. Nilai terendah.

# PRAKTIKUM 2: CLASS DAN OBJECT PADA JAVA

Pertemuan ke : 2

Total Alokasi Waktu : 90 menit
- Pre-Test : 15 menit
- Praktikum : 55 menit
- Post-Test : 20 menit

Total Skor Penilaian : 100%
- Pre-Test : 20 %
- Praktikum : 40 %
- Post-Test : 40 %

## 2.1. TUJUAN DAN INDIKATOR CAPAIAN

Setelah mengikuti praktikum ini mahasiswa diharapkan:
1. mampu menjelaskan konsep pemrograman berorientasi objek
2. mampu menjelaskan **object**
3. mampu menjelaskan **Class**
4. mampu menjelaskan **instance**
5. mampu membuat contoh kasus sederhana

Indikator ketercapaian diukur dengan:
1. kemampuan memahami konsep pemrograman berorientasi objek
2. kemampuan menyebutkan contoh objek dalam kehidupan sehari-hari/ dalam kasus aplikasi
3. kemampuan menulis *coding* materi **Class**
4. kemampuan menulis *coding* **instance** dan penerapannya.

## 2.2. 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 didefnisikan 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 getNama() {               // method brp Fungsi
        return nama;
    }
    public void setNama(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-*runing* pada saat pembuatan suatu objek.
**Contructor** mempunyunyai 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 |
|---|---|
| `public Class Prak2 {`<br>`  public static void main(String[] args) {`<br>`  Mahasiswa m= new Mahasiswa();`<br>`  m.setNim("08018222");`<br>`  m.setNama("Rudi");`<br>`  System.out.println("Nim :"+ m.getNim());`<br>`  System.out.println("Nama :"+ m.getNama());`<br>`  }`<br>`}` | Nim :08018222<br>Nama :Rudi<br>BUILD SUCCESSFUL (total time: 1 second) |

## 2.3. ALAT DAN BAHAN

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

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

## 2.4. LANGKAH PRAKTIKUM

1. Buatlah **project** baru dengan nama prak2.
2. Buatlah kelas pojo dengan nama **PersegiPanjang** seperti pada contoh di atas.

## 2.5. TUGAS

Buatlah program untuk:

1. membuat **class Titik** dengan atribut absis dan ordinat, metode/operasi yang ada menentukan sebuah titik, menghitung jarak dua titik, menentukan titik tengah dua titik
2. membuat **class Waktu** dengan atribut jam, menit, detik , metode/operasi yang ada menentukan sebuah jam, menghitung lama waktu

# PRAKTIKUM 3: KONSTRUKTOR, ENKAPSULASI & HIDDING INFORMATION

Pertemuan ke : 3

Total Alokasi Waktu : 90 menit
- Pre-Test : 15 menit
- Praktikum : 55 menit
- Post-Test : 20 menit

Total Skor Penilaian : 100%
- Pre-Test : 20 %
- Praktikum : 40 %
- Post-Test : 40 %

## 3.1 TUJUAN DAN INDIKATOR CAPAIAN

Setelah mengikuti praktikum ini mahasiswa diharapkan:
1. mampu menjelaskan **constructor** (konstruktor)
2. mampu menjelaskan **encapsulation** (enkapsulasi) dan **Information hiding**
3. mampu membuat contoh kasus penerapannya dalam program Java sederhana

Indikator ketercapaian diukur dengan:
1. menulis *coding* **constructor**
2. menjelaskan **constructor** dalam **instance object**
3. mengimplementasikan **encapsulation** dan **information hiding** dalam program Java
4. menjelaskan perbedaan penggunaan **information hiding**

## 3.2 TEORI PENDUKUNG

**Constructor** (konstruktor) merupakan sebuah **pseudo-method** yang dibuat pada sebuat 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 kontruktor didefinisikan dalam dua aturan berikut:

1. Nama konstruktor harus sama dengan nama kelasnya
2. Konstruktor harus tidak memiliki explisit 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 contructor dan method, yaitu:

| Java Constructor | Java Method |
|---|---|
| Constructor digunakan untuk menginiasialisasi 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 explisit |
| 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());
   }
}
```

## 3.3 ALAT DAN BAHAN

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

1. Komputer.

2. IDE Intellij IDEA Community
3. SDK Java

## 3.4 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.

## 3.5 TUGAS

1. Buatlah sebuah driver **Class** dengan nama Tester.java dan object **Class** dengan nama Karyawan.java.
2. Buatlah sebuah konstruktor default yang telah berisi variabel gaji tetap dan berikan nilai tertentu.
3. Buatlah sebuah konstruktor berparameter dengan sebuah parameter dan simpan parameter tersebut pada variabel gaji tetap.
4. Buatlah tiga buah variabel dengan nama variabel: nomor pegawai, nama pegawai dan gaji bonus.
5. Lakukan pembuatan method mutator dan accessor untuk tiga variabel tersebut.
6. Buat sebuah method menghitung gaji untuk karyawan baru yang berasal dari gaji tetap saja.
7. Buat sebuah method menghitung gaji untuk karyawan lama yang berasal dari gaji tetap ditambahkan bonus.
3. Tampilkan di layar berupa informasi nomor pegawai, nama pegawai dan hitungan gaji untuk objek karyawan baru dan karyawan lama.

# PRAKTIKUM 4: PEWARISAN (INHERITANCE)

Pertemuan ke : 4

Total Alokasi Waktu : 90 menit
- Pre-Test : 15 menit
- Praktikum : 55 menit
- Post-Test : 20 menit

Total Skor Penilaian : 100%
- Pre-Test : 20 %
- Praktikum : 40 %
- Post-Test : 40 %

## 4.1 TUJUAN DAN INDIKATOR CAPAIAN

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat memahami tentang pewarisan dan penerapannya.
Indikator ketercapaian diukur dengan keberhasilan mahasiswa dalam menerapkan pemrograman dengan pewarisan.

## 4.2 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 |
|---|---|
| public class Main {<br>public static void main(String[] args) {<br>Mahasiswa m=new Mahasiswa();<br>m.setNim("06018251");<br>m.setNama("Tejo");<br>m.setAlamat("Sleman");<br>System.out.println("Nim :"+m.getNim());<br>System.out.println("Nama :"+m.getNama());<br>System.out.println("Alamat :"+m.getAlamat());<br>} | Nim :06018251<br>Nama :Tejo<br>Alamat :Sleman<br>BUILD SUCCESSFUL (total time: 1 second) |

## 4.3 ALAT DAN BAHAN

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

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

## 4.4 LANGKAH PRAKTIKUM

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

## 4.5 TUGAS

Buatlah project seperti perintah langkah-langkah praktikum dengan studi kasus membuat kelas lingkaran, kelas persegi panjang disertai metodenya yang merupakan turunan dari kelas titik

# PRAKTIKUM 5: POLYMORPHISME

Pertemuan ke : 5

Total Alokasi Waktu : 90 menit
- Pre-Test : 15 menit
- Praktikum : 55 menit
- Post-Test : 20 menit

Total Skor Penilaian : 100%
- Pre-Test : 20 %
- Praktikum : 40 %
- Post-Test : 40 %

## 5.1 TUJUAN DAN INDIKATOR CAPAIAN

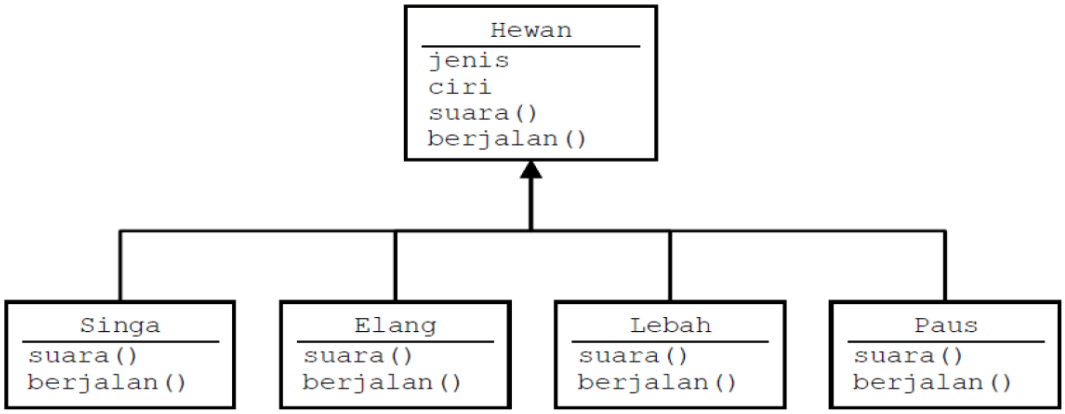
Setelah mengikuti praktikum ini mahasiswa diharapkan dapat menerapkan polimorfisme dalam pemrograman.
Indikator ketercapaian diukur dengan keberhasilan mahasiswa dalam menerapkan polimorfisme dalam pemrograman.

## 5.2 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 terganting 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.

## 5.3 ALAT DAN BAHAN

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

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

## 5.4 LANGKAH PRAKTIKUM

1. Buat project prak5 dengan menggunakan NetBeans
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 metodhmetodhnya.
5. Buatlah sebuah kelas KelompokStudi yang merupakan kelas turunan dari Mahasiswa dan buatlah metodhmetodhnya.
6. Buatlah 1 objek untuk menginstantiasi kelas HM dan mengimplementasikan beberapa metodh yang telah Anda definisikan dalam kelas HM
7. Panggilllah objek yang telah Anda buat pada kelas KelompokStudi dan HM untuk bekerja sehingga tampil efek-efek polymorfisme

## 5.5 TUGAS

Kembangkan program dari ilustrasi gambar di atas dengan mengimplementasikam method-method yang ada kemudian buatlah testernya

# PRAKTIKUM 6: RELASI ANTAR CLASS

Pertemuan ke : 6

Total Alokasi Waktu : 90 menit
- Pre-Test : 15 menit
- Praktikum : 55 menit
- Post-Test : 20 menit

Total Skor Penilaian : 100%
- Pre-Test : 20 %
- Praktikum : 40 %
- Post-Test : 40 %

## 6.1 TUJUAN DAN INDIKATOR CAPAIAN

Setelah mengikuti praktikum ini mahasiswa diharapkan:
1. mampu melakukan analisis terhadap class-class yang saling ber-relasi sesuai dengan konsep berorientasi objek
2. mampu melakukan implementasi konsep relasi antar class ke dalam program

Indikator ketercapaian diukur dengan: (sesuaikan dengan RPS)
1. Menghasilkan implementasi relasi antar class dengan menggunakan Bahasa Java.
2. Dapat menjelaskan relasi antar class dari code program yang ada.

## 6.2 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 eleman 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.

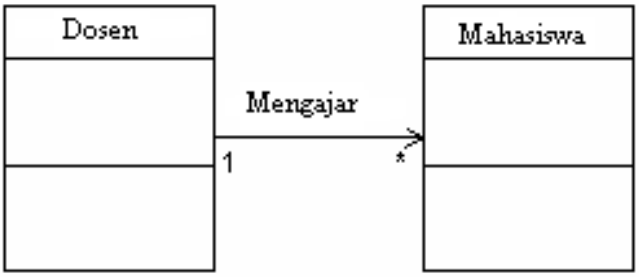
**Hubungan Antar Kelas**

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.

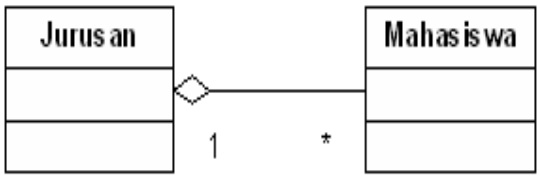
**Contoh 1:**

```
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:

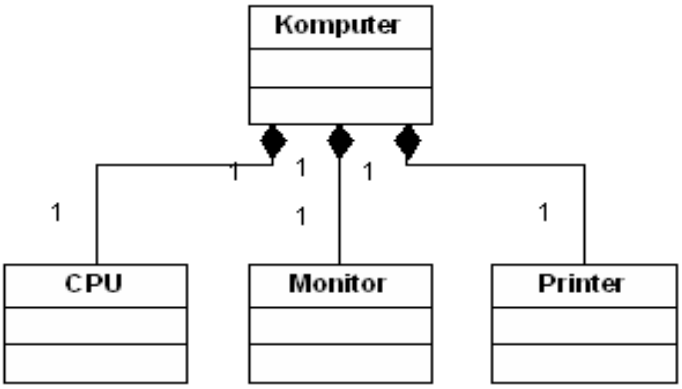
**Contoh 2:**

```
//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);
        }
}
```

## 6.3 ALAT DAN BAHAN

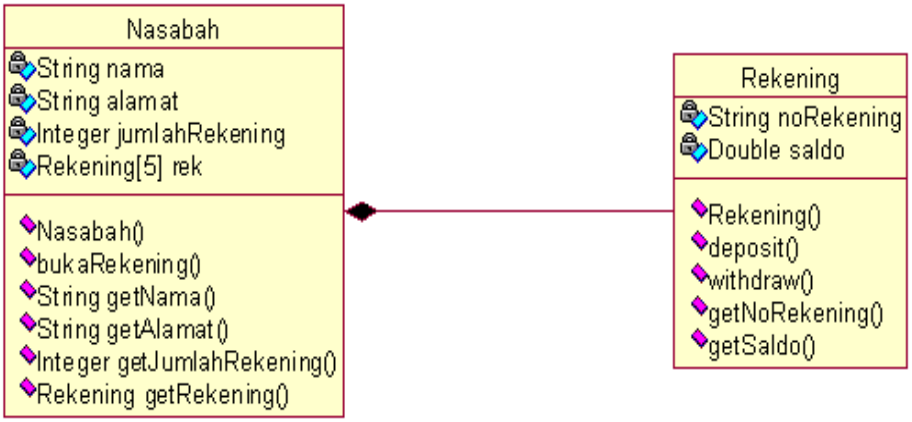
Alat dan bahan yang digunakan dalam praktikum ini yaitu:

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

## 6.4 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!

## 6.5 TUGAS




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

**Perintah:**

- Jelaskan jenis relasi antar class pada Gambar 6.4.!
- Buatlah code Java untuk mengimplementasikan class diagram Gambar 6.4!
- Buatlah class tester untuk menampilkan data seorang nasabah dan rekeningnya! (data diasumsikan sendiri)

# PRAKTIKUM 7: ABSTRACT CLASS

Pertemuan ke : 7

Total Alokasi Waktu : 90 menit
- Pre-Test : 15 menit
- Praktikum : 55 menit
- Post-Test : 20 menit

Total Skor Penilaian : 100%
- Pre-Test : 20 %
- Praktikum : 40 %
- Post-Test : 40 %

## 7.1 TUJUAN DAN INDIKATOR CAPAIAN

Setelah mengikuti praktikum ini mahasiswa diharapkan:
1. Menjelaskan kelas abstrak
2. Membuat contoh kasus kelas abstrak dan penerapannya

Indikator ketercapaian diukur dengan:
1. Mengimplementasikan kelas abstrak
2. Menulis coding kelas abstrak

## 7.2 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;
       }

}
```

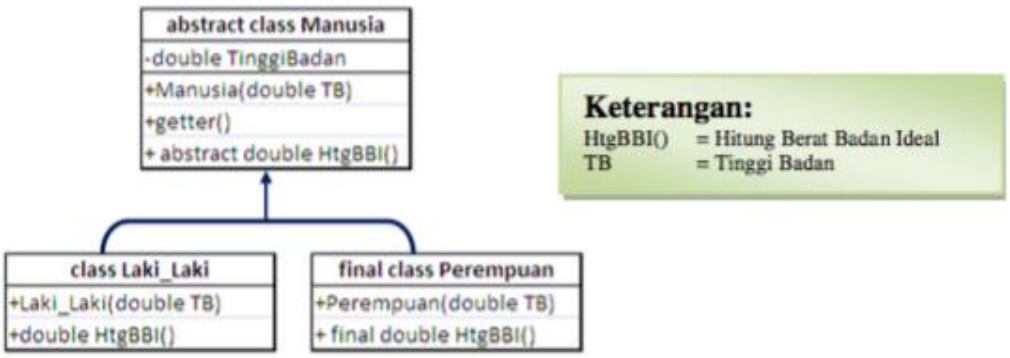
## 7.3 ALAT DAN BAHAN

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. IDE Intellij IDEA Community
3. SDK Java
4. dll.

## 7.4 LANGKAH PRAKTIKUM

1. Buat project prak8 dengan menggunakan netbeans
2. Perhatikan class diagram berikut:




Untuk menghitung berat badan ideal sesuai dengan class adalah:

a. Laki-laki = (tinggi badan(cm)-100) kg X 90%
b. 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 doule hitungBeratBadanIdeal();
}
```

4. Buatlah sebuah kelas Lakilaki seperti code dibawah ini.

```
public class Lakilaki extends Manusia{

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

## 7.5 TUGAS

1. Buatlah kelas tester dari contoh diatas.
2. Lakukan instansiasi objek untuk objek manusia kemudian berikan nilai untuk tinggi badannya dan hitung tinggi badan ideal objek manusia tersebut.
3. Lakukan instansiasi objek untuk objek laki-laki kemudian berikan nilai untuk tinggi badannya dan hitung tinggi badan ideal objek laki-laki tersebut.
4. Lakukan instansiasi objek untuk objek perempuan kemudian berikan nilai untuk tinggi badannya dan hitung tinggi badan ideal objek perempuan tersebut.
5. Buat kelas baru dengan nama PerempuanDewasa sebagai turunan dari kelas perempuan dan lakukan overriding pada method hitungBeratBadanIdeal dengan rumus menghitung berat badan ideal adalah (tinggi badan(cm)-100) kg X 85%
6. Lakukan instansiasi objek untuk objek perempuandewasa kemudian berikan nilai untuk tinggi badannya dan hitung tinggi badan ideal objek perempuandewasa tersebut.

# PRAKTIKUM 8: INTERFACE

Pertemuan ke : 8

Total Alokasi Waktu : 90 menit
- Pre-Test : 15 menit
- Praktikum : 55 menit
- Post-Test : 20 menit

Total Skor Penilaian : 100%
- Pre-Test : 20 %
- Praktikum : 40 %
- Post-Test : 40 %

## 8.1 TUJUAN DAN INDIKATOR CAPAIAN

Setelah mengikuti praktikum ini mahasiswa diharapkan: Membuat dan mendeklarasikan kelas Interface

Indikator ketercapaian diukur dengan: (sesuaikan dengan RPS)
1. Mengimplementasikan kelas interface
2. Menulis coding kelas interface

## 8.2 TEORI PENDUKUNG

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

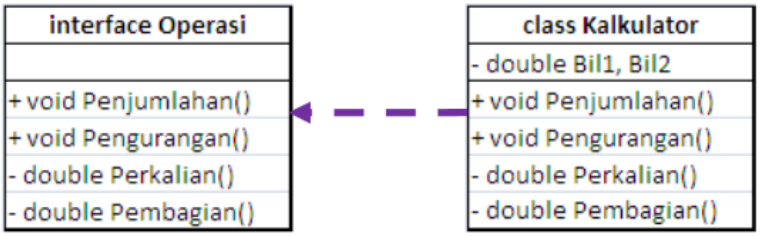
```
interface Nama_Interface
{
    //deklarasi variabel dan/atau method
}
```

Contoh:

```
interface Operasi
{
    //deklarasi variabel dan/atau method
    public void Penjumlahan();
    public void Pengurangan();
    public double Perkalian();
    public double Pembagian();
}
```

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 dideklarasi 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:

```
class Nama_Kelas implements Nama_Interface
{
    //isi kelas
}
```

Contoh implementasi:

```
class Kalkulator implements Operasi
{
    public void Penjumlahan()
    {
        //isi method Penjumlahan()
    };

    public void Pengurangan()
    {
        //isi method Pengurangan()
    };

    public double Perkalian()
    {
        //isi method Perkalian()
        return 0;
    };

    public double Pembagian()
    {
        //isi method Pembagian()
        return 0;
    };
};
```

## 8.3 ALAT DAN BAHAN

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

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

## 8.4 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!

## 8.5 TUGAS

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!

```
interface Operasi
{
    //deklarasi variabel dan/atau method
    public void Penjumlahan();
    public void Pengurangan();
    public double Perkalian();
    public double Pembagian();
}
```

# PRAKTIKUM 9: EXCEPTION HANDLING dan I/O

Pertemuan ke : 9

Total Alokasi Waktu : 90 menit
- Pre-Test : 15 menit
- Praktikum : 55 menit
- Post-Test : 20 menit

Total Skor Penilaian : 100%
- Pre-Test : 20 %
- Praktikum : 40 %
- Post-Test : 40 %

## 9.1 TUJUAN DAN INDIKATOR CAPAIAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu menerapkan exception handling dan menerapkan I/O.
Indikator ketercapaian diukur dengan kemampuan mahasiswa dalam menerapkan exception handling dan menerapkan I/O.

## 9.2 TEORI PENDUKUNG

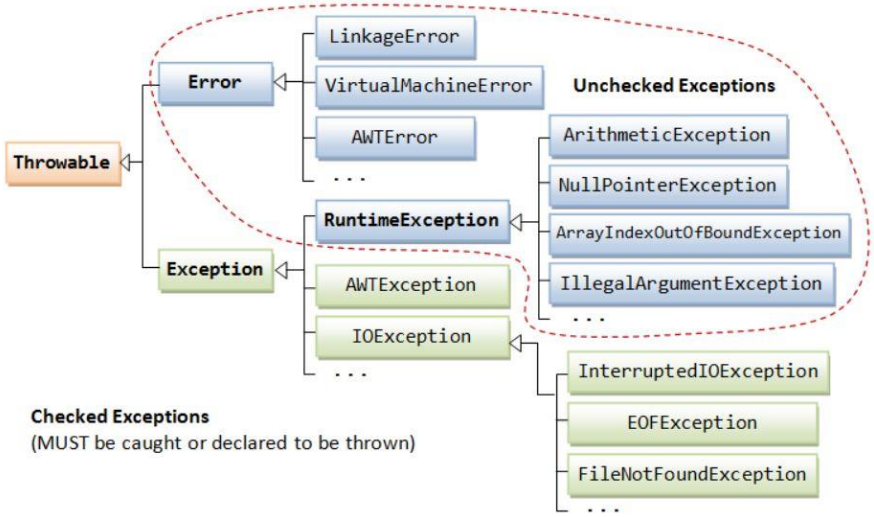
Pada program-program yang membutuhkan data-data eksternal akan diperlukan suatu proses input dan ouput (I/O). Java mendukung proses I/O ini di dalam paket java.io. Di dalam paket tersebut tersimpan banyak kelas dan interface siap pakai yang akan memudahkan programmer dalam pengambilan dan penyimpanan 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. S*tream* 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 eror. 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)

## 9.3 ALAT DAN BAHAN

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

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

## 9.4 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.

```java
package IO;

public class ContohOutput {
    public static void main(String[] args) {
        int varInt = 45;
        double varDouble = 7.98845;
        float varFloat = 6.97f;
        char varChar = 'X';
        String varString = "Ini sebuah String";

        System.out.printf("%d\n", varInt);
        System.out.printf("%e\n", varDouble);
        System.out.printf("%f\n", varFloat);
        System.out.printf("%.2f\n", varFloat);
        System.out.printf("%c\n", varChar);
        System.out.printf("%.3s\n", varString);
        System.out.printf("%s\n", varString);

        System.out.println("=========================");
        System.out.print("Ini adalah ");
        System.out.print("contoh penggunaan print");

        System.out.println();
        System.out.println("=========================");
        System.out.println("Ini adalah ");
        System.out.println("contoh penggunaan println");
    }
}
```

```
Output - Prak9 (run)
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

```java
package IO;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class ContohInput {
    public static void main(String[] args) {
        String nama = "";
        BufferedReader input = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Nama : ");
        try {
            nama = input.readLine();
        } catch(IOException e) {
            e.printStackTrace();
        }
        System.out.println("Nama anda : " + nama);
    }
}
```

```
Output - Prak9 (run)
run:
Nama :
Herman
Nama anda : Herman
```

Gambar 9.3 Hasil percobaan 2.

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

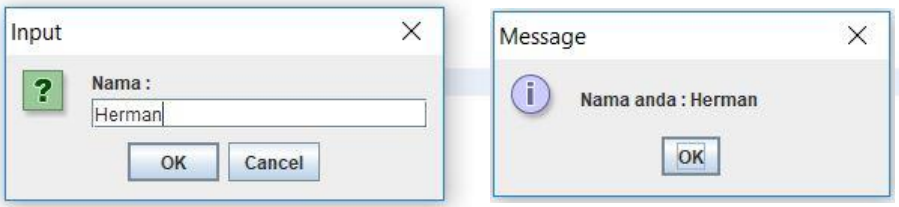
```java
package IO;

import java.util.Scanner;

public class ContohInputScanner {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Nama : ");
        String nama = input.nextLine();
        System.out.println("Nama anda : " + nama);
    }
}
```

```
Output - Prak9 (run)
run:
Nama :
Herman
Nama anda : Herman
```

Gambar 9.4 Hasil percobaan 3.

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

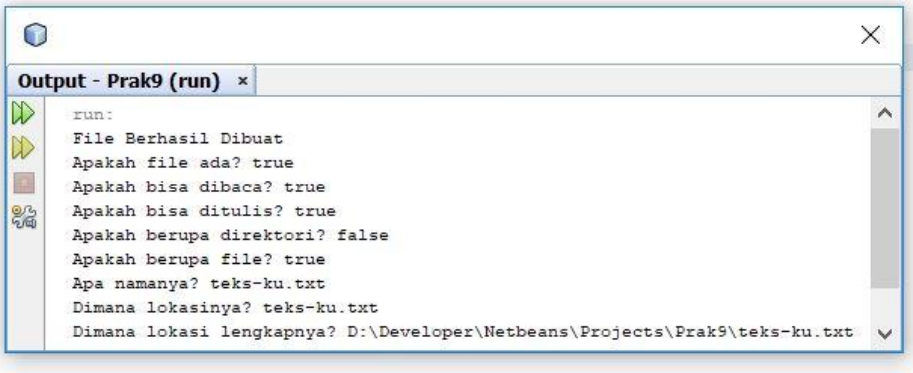
```
1   package IO;
2
3   import javax.swing.JOptionPane;
4
5   public class ContohInputJOptionPane {
6       public static void main(String[] args) {
7           String nama = JOptionPane.showInputDialog(null, "Nama : ");
8           JOptionPane.showMessageDialog(null, "Nama anda : " + nama);
9       }
10  }
11
12
13
14
```




Gambar 9.5 Hasil percobaan 4.

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

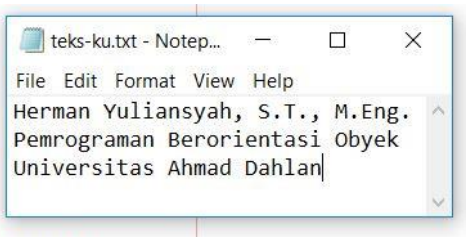
```
1   package IO;
2
3   public class ContohClassFile {
4       public static void main(String[] args) {
5           java.io.File file = new java.io.File("teks-ku.txt");
6           try {
7               if (file.createNewFile())
8                   System.out.println("File Berhasil Dibuat");
9               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.

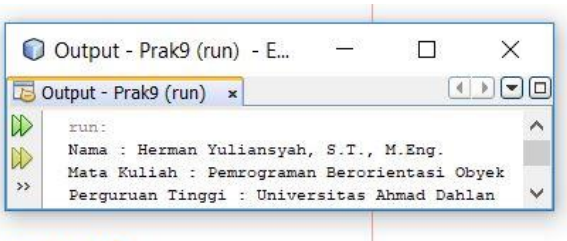
```
1  package IO;
2
3  import java.io.FileNotFoundException;
4
5  public class ContohTulisFile {
6      public static void main(String[] args) {
7
8          java.io.File file = new java.io.File("teks-ku.txt");
9
10         try {
11             java.io.PrintWriter output = new java.io.PrintWriter(file);
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 }
```




Gambar 9.8 Hasil percobaan 7.

## 9.5 TUGAS

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.

1. Hitung volume balok kayu yang Anda beli kemudian hasil perhitungannya ditampilkan dilayar dan juga disimpan dalam sebuah file.
2. Berikan *exception handling* untuk masukan supaya melakukan pengecekan terhadap kesalahan pengisian nilai panjang, lebar dan tingginya.

# PRAKTIKUM 10: APLIKASI GAME

Pertemuan ke : 10

Total Alokasi Waktu : 90 menit
- Pre-Test : 15 menit
- Praktikum : 55 menit
- Post-Test : 20 menit

Total Skor Penilaian : 100%
- Pre-Test : 20 %
- Praktikum : 40 %
- Post-Test : 40 %

## 10.1 TUJUAN DAN INDIKATOR CAPAIAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu menerapkan model OOP dalam aplikasi.
Indikator ketercapaian diukur dengan keberhasilan mahasiswa dalam menerapkan model OOP dalam aplikasi.

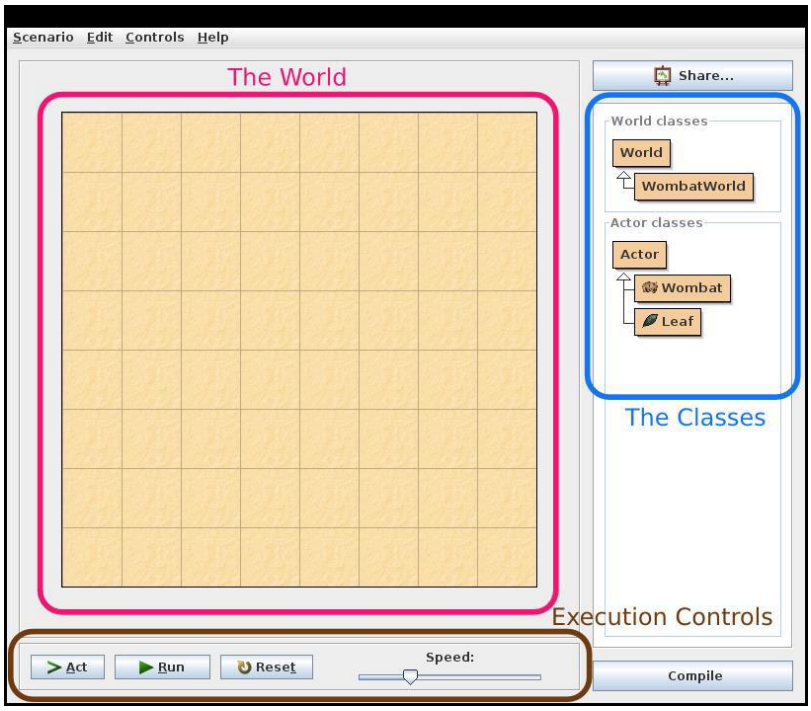
## 10.2 TEORI PENDUKUNG

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.

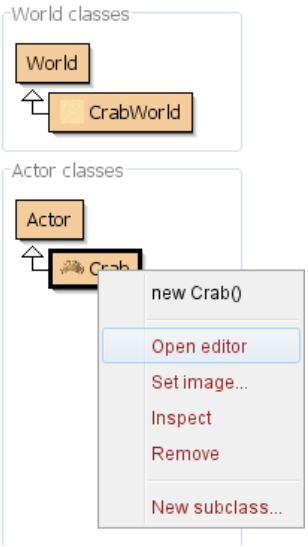
## 10.3 ALAT DAN BAHAN

Alat dan bahan yang digunakan dalam praktikum ini yaitu:

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

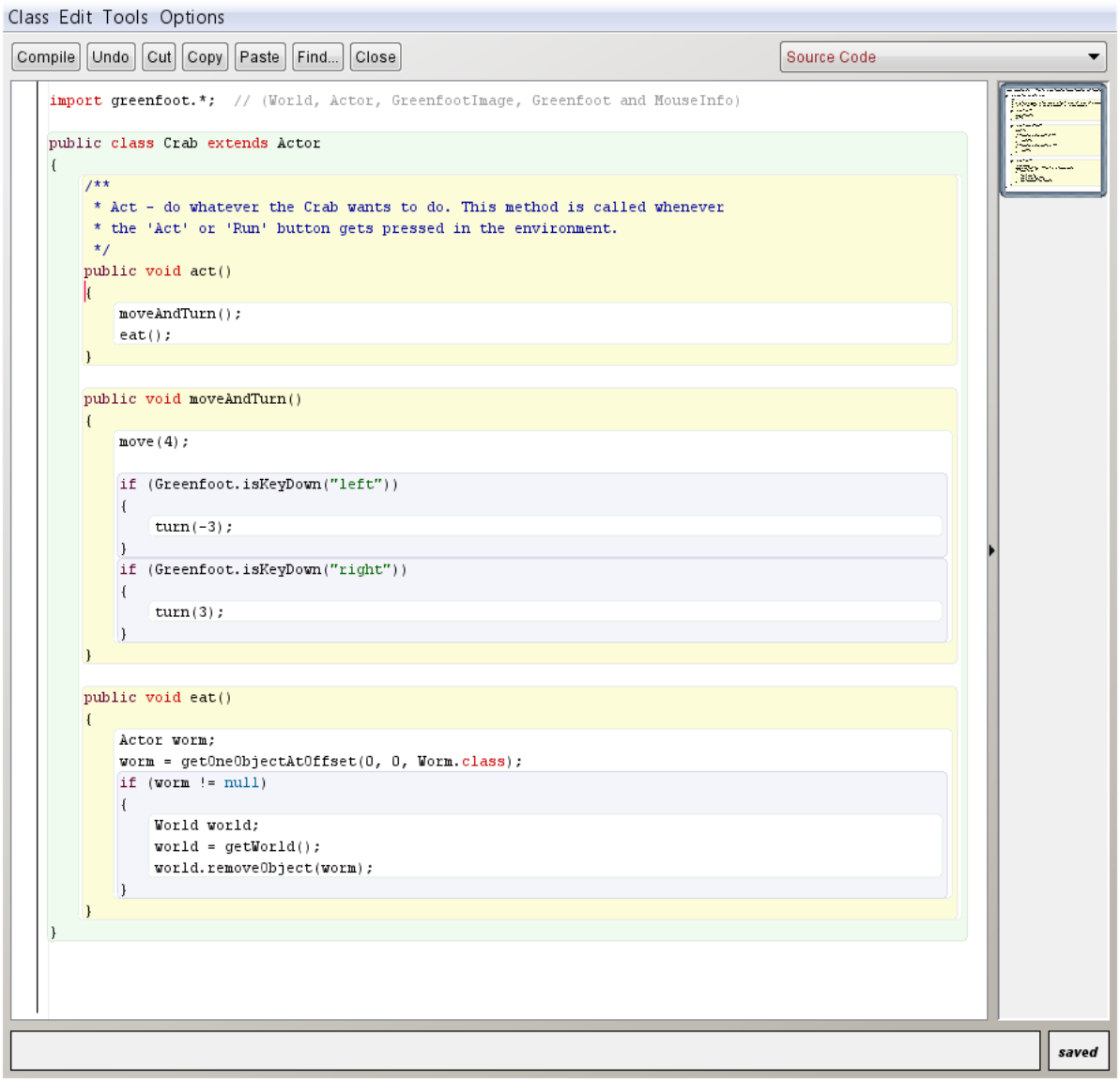
## 10.4 LANGKAH PRAKTIKUM

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.



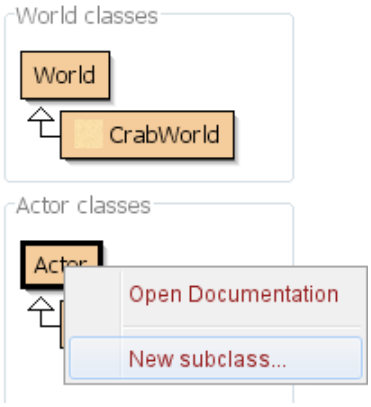

Gambar 10.2 Buka editor pada Aktor Crab.

3. Lengkapi class Crab menjadi seperti pada Gambar 10.3.



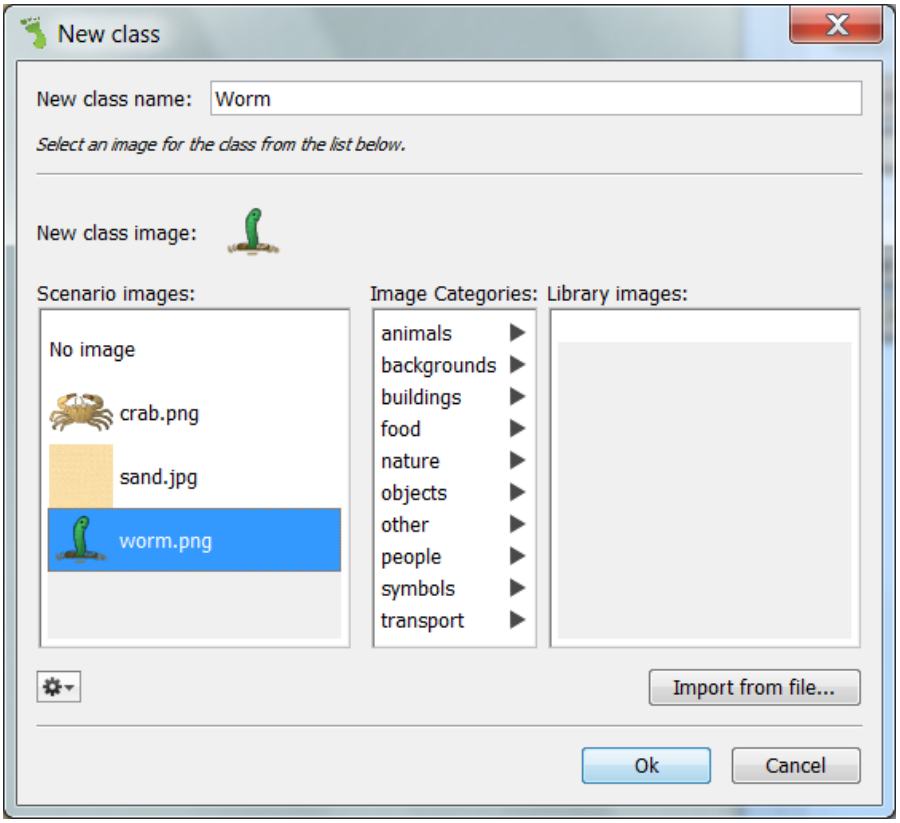

Gambar 10.3 Kode perilaku untuk Aktor Crab.

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



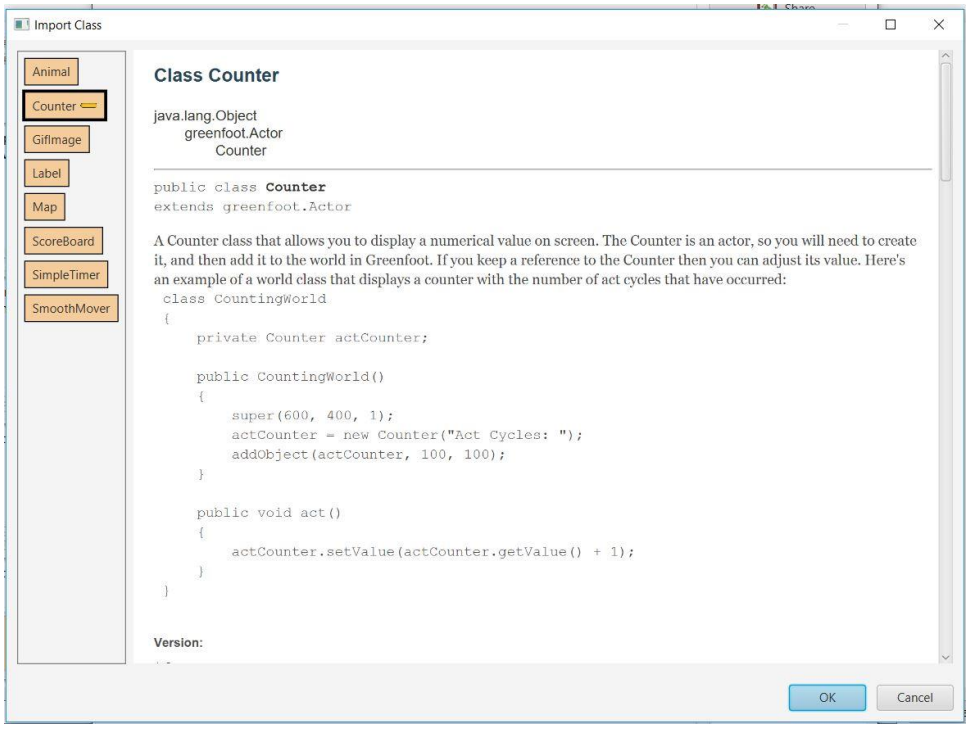

Gambar 10.4 Buat Aktor Worm.

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



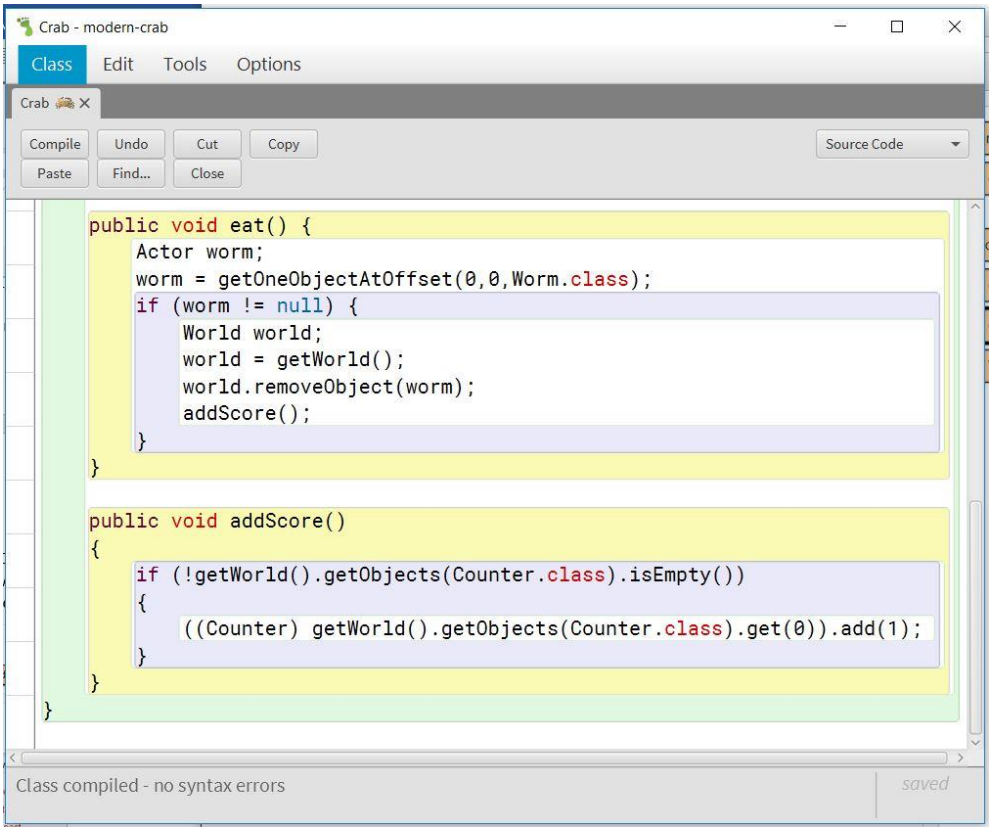

Gambar 10.5 Pilih tampilan objek Worm.

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



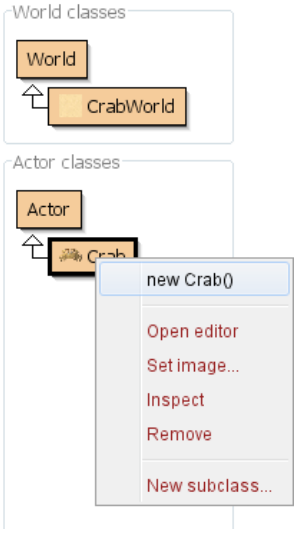

Gambar 10.6 Import class counter.

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



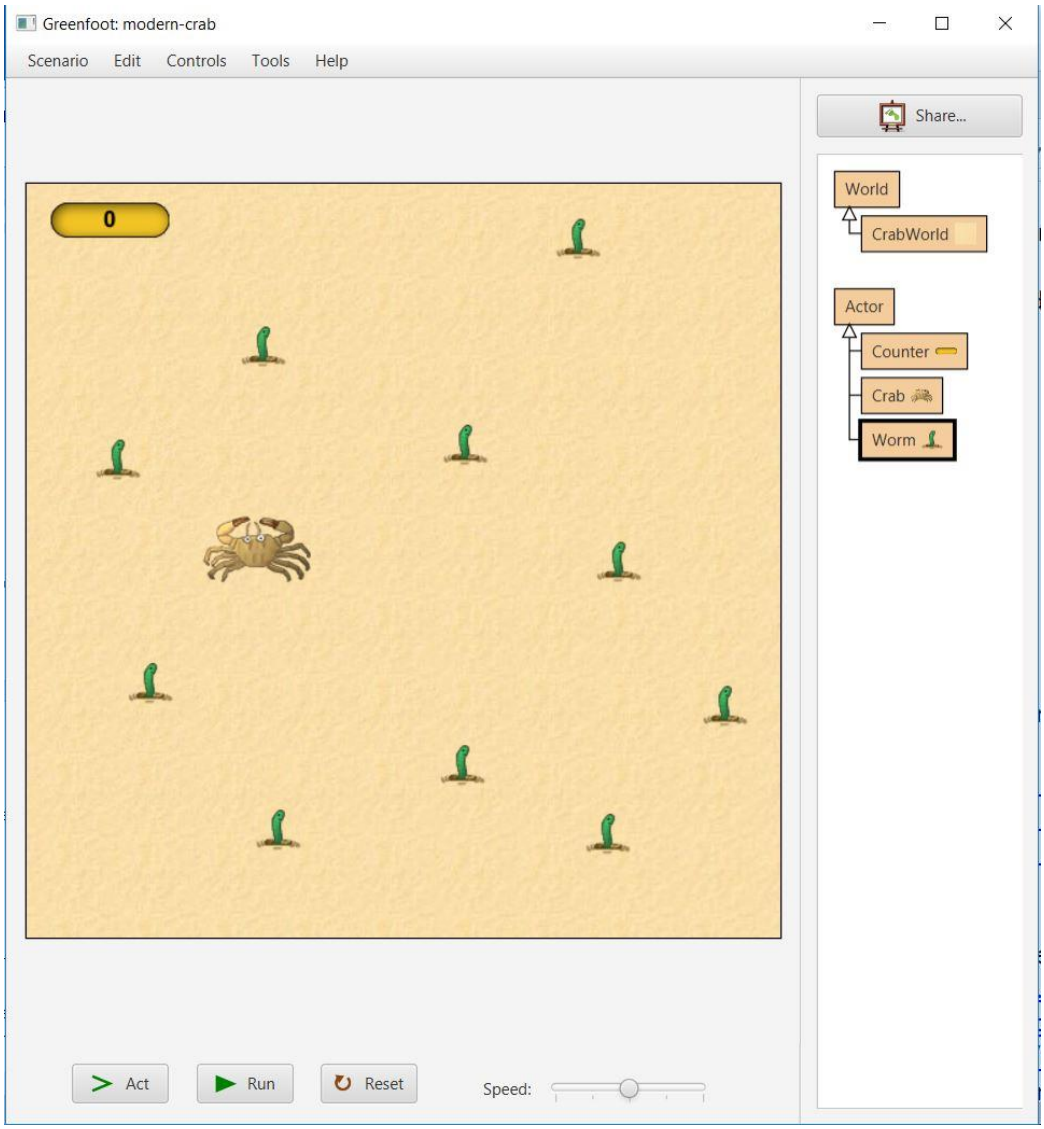

Gambar 10.7 Kode untuk meng-update Counter.

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.
11. Pilih Tombol Run untuk menjalankan greenfoot dan gunakan panah pada keyboard untuk mengatur gerak dari Crab.

## 10.5 TUGAS

Tambahkan skenario untuk game diatas dengan perilaku sebagai berikut:

1. Crab bisa bergerak cepat sejauh 3x lipat gerakan normal (Dash) apabila ditekan tombol Up.
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 **Greenfoot.getRandomNumber()** untuk membangkitkan angka acak untuk digunakan saat perpindahan tempat.
3. Hanya ada satu cara si Crab memakan Worm yaitu dengan bergerak kencang (Dash) ke arah Worm sebelum Worm itu sempat berpindah tempat.

**LEMBAR JAWABAN PRE-TEST / POST-TEST**

| | | |
|---|---|---|
| **Nama :**<br>**NIM :** | **Asisten:**<br>**Paraf Asisten:** | **Tanggal:**<br>**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
TEKNIK INFORMATIKA
UNIVERSITAS AHMAD DAHLAN


UNIVERSITAS AHMAD DAHLAN