# Pengenalan Lingkungan

# Translasi dari C++ ke JAVA
# Praktikum Pertama JAVA

Oleh : Yohanes Nugroho

# Overview

- Tujuan
- JDK vs JRE
- Kompilator dan Interpreter
- Kesalahan Umum
- Membuat dokumentasi
- Contoh Dokumentasi Sederhana
- Mengekstrak dokumentasi

# Tujuan

- Mengenal lingkungan pemrograman Java
- Mampu mengkompilasi dan menjalankan program Java
- Mengenal dan mampu memanfaatkan fitur dokumentasi di Java dengan menggunakan Javadoc

# JDK vs JRE

- JDK (Java Development Kit) digunakan untuk mengembangkan aplikasi Java. JDK terdiri atas:
  - kompilator, interpreter, tools-tools lain
- JRE (Java Runtime Environment) digunakan hanya untuk menjalankan aplikasi Java yang sudah "jadi" (sudah dikompilasi). JRE hanya terdiri atas interpreter.
- Keduanya tidak berisi dokumentasi Java (harus didownload terpisah).

# Cross Platform

- Java bersifat cross platform.
- Source code Java dapat dikompilasi di OS apapun yang mendukung Java.
- Hasil kompilasi bisa dijalankan di OS apapun yang mendukung Java.
- Anda boleh bekerja di lingkungan Windows atau Linux.
- Cara kerja di aneka OS sifatnya serupa.

# Hello.java

```
class Hello {
  public static void main(String
    argv[]) {
     System.out.println("Hello world");
  }
}
```

# Kompilator dan Interpreter

- Program dalam bahasa Java dikompilasi, lalu dieksekusi dengan interpreter
- Jika memiliki source code bernama Hello.java, cara kompilasi adalah:
  - `javac Hello.java`
- Perintah di atas (jika sukses) akan menghasilkan `Hello.class`
- Cara menjalankan:
  - `java Hello`

# Kesalahan umum

- Java bersifat case sensitive: Nama kelas, variabel, dll harus benar, termasuk juga **Nama File harus benar (dan sama dengan nama kelas)**
  - **jika sudah memakai fitur package, nama direktori juga harus benar**
- **Perhatikan dalam kompilasi perlu extension .java, ketika interpretasi tidak perlu .class**
  - **lupa tidak memakai .java atau memakai .class akan menyebabkan error**

# Konversi Point.cc ke Point.java

- Header file dan body harus menjadi satu. Tidak ada lagi file "header"
- Contoh:

```
class Point {
   int getX() { return x;}
}
```

# Library call

- Semua library call C `scanf` (atau C++ `cin`) jangan dipakai. Penggantiannya rumit di Java
- Library call C `printf` (atau C++ `cout`) diganti dengan:

```
System.out.println("teks");
```

  - Tanpa format specifier `%d %s`, dll
- String dan integer bisa digabung

```
int jumlah = 6;
System.out.println("Ada "  + jumlah
 + " pinguin");
```

# Hanya ada konstruktor

- Desktruktor tidak ada
  - Konsep finalizer akan dibahas waktu kuliah
- Copy constructor tidak ada
- Operator= saat ini dihapus dulu
  - nanti akan diganti dengan `clone()`

# Tidak ada lagi pointer

- Semua:

```
Point *p = new Point();
```

- Menjadi

```
Point p = new Point();
```

  - Tidak perlu **delete**
- **Point p tidak lagi mendeklarasikan variabel p (tapi reference ke p)**
- **Semua parameter "Point *p" menjadi "Point p" saja**

# Access Modifier: C++

- Di C++

```
class Point {
   private:
      int x;
      int y;
   public: int getX() { return x;}
   int getY() { return y;}
}
```

- Di Java, access modifier tanpa titik dua (:) dan harus diberikan di setiap method/property

# Access Modifier: Java

```
class Point {
    private int x;
    private int y;
    public int getX() { return x;}
    public int getY() { return y;}
}
```

- Perhatikan bahwa "private" dan "public" diulang untuk setiap method dan member.

# Driver

- Di Java, tidak boleh ada fungsi di luar kelas.
- Buat kelas DriverPoint di file DriverPoint.java
- Buat satu method:

```
public static void main(String
  argv[]) {
   /*isikan main(driver) point di
    sini*/
   /*tanpa return 0; */
}
```

# Kompilasi Point dan Driver

- Satu persatu:
  - javac Point.java
  - javac DriverPoint.java
- Sekaligus:
  - javac DriverPoint.java Point.java
- Menjalankan:
  - java DriverPoint

# Membuat Dokumentasi

- Java mendukung pembuatan dokumentasi yang di-embed di dalam source code
- Dokumentasi ini disebut dengan dokumentasi javadoc
  - Dengan tools yang bernama javadoc, dokumentasi bisa diekstrak ke file .html
- Dokumentasi ditulis dalam bentuk komentar dengan format `/** komentar */`

# Mengekstrak dokumentasi

- Untuk satu file caranya:
  - `javadoc NamaFile.java`
- Sedangkan untuk banyak file:
  - `javadoc NamaFile1.java NamaFile2.java`
  - `javadoc *.java`
- Beberapa file akan dihasilkan untuk satu kelas
- Dokumentasi bisa dilihat dibrowser (buka file index.html)

# Contoh Dokumentasi Sederhana

```
/** Kelas untuk menguji kompilator Java
@author Yohanes Nugroho
*/
class Hello {
   /** main program */
   public static void main(String argv[]){
      System.out.println("Hello world");
   }
}
```

# Tambahan

- Berikan komentar dengan format JavaDoc
- Hasilkan komentar menjadi HTML
  - File HTML tidak dikirimkan ketika mengumpulkan tugas
- Catat apa saja yang menjadi kesulitan ketika porting dari C++ ke Java
  - Tanyakan di kelas