Introduction to Java

Object-Oriented Programming
Outline

- Brief introduction of the language and platform
- Sample programs
- Code structure
- How to compile and run a program

Readings:
- HFJ: Ch.1, Ch.5.
- GT: Ch. 2.
Java

- 1991: developed by Sun Microsystems as a small programming language for embedded household devices
  - Initially called Oak
- Java 1.0.2, 1.1
  - “Write Once, Run Anywhere”
  - very slow
  - became popular with webpages running applets
- Java 2 (versions 1.2 - 1.4)
  - much faster, powerful,
  - 3 platforms: J2ME, J2SE, J2EE
- Java 5.0, 6.0 (versions 1.5-1.6)
  - more powerful
Why Java?

- Object-oriented
- Portability
- Safe
  - Data always initialized, references always type-safe
  - Access to "private" or "package private" data and methods is rigidly controlled.
  - Addresses the weaknesses of older programming languages
- Built-in multi-threading
Compiling and interpreting

- Program source code compiled into bytecode
- Bytecode is platform-independent
- Bytecode is executed in an interpreter environment (virtual machine)

Java program ➔ Compiler ➔ Java bytecode program ➔ Java Virtual Machine

- for Windows
- for Linux
- for Mac OS
Java Virtual Machine (JVM)

- Virtual machines depend on specific platforms (hardware, OS)
- Provide Java programs with (platform-independent) run-time environments
- Ensure system security
- Normally provided as software
  - JRE - Java Runtime Environment
- Java platform: JVM + APIs
Types of Java applications

- Desktop applications – *Java Standard Edition (J2SE)*
  - Java Application: normal Java application running on desktops; console or GUI
  - Java Applet: embedded application running within Web browsers
- Server applications – *Java Enterprise Edition (J2EE)*
  - Webservices, JSP and Servlet
- Mobile applications – *Java Microedition (J2ME)*
- Java Card
Java Desktop Applications

- Focus of this course
- Complete application programs
- Console or GUI
- Launched by `java` command
HelloWorld.java:

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

- **same name with the class**
- **this is a class**
- **class name**
- **start of the class**
- **method name**
- **public, so that everyone can access**
- **a statement**
  - it says print to standard output
- **end of the class**
Compile and run

- Compile HelloWorld.java
  `javac HelloWorld.java`
- Run
  `java HelloWorld`

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

```
%> javac HelloWorld.java
%> java>HelloWorld
Hello, world
```
Application with more than one class

Two classes in separated files

**TestGreeting.java:**

```java
public class TestGreeting {
    public static void main(String[] args) {
        Greeting gr = new Greeting();
        gr.greet();
    }
}
```

**Greeting.java:**

```java
public class Greeting {
    public void greet() {
        System.out.print("Hi there!");
    }
}
```
Compile and run

- **Compile**
  
  javac TestGreeting.java
  
  Greeting.java is automatically compiled

- **Run**
  
  java TestGreeting

%> javac TestGreeting.java
%> java TestGreeting
Hi there!
JDK – Java Development Kit

- Free development and run-time environment
- Most widely used Java software development kit
- Main components:
  - `javac` compiler, converts source code into Java bytecode
  - `java` interpreter and application loader
  - `javadoc` documentation generator, automatically generates documentation from source code comments
  - `jdb` debugger
  - ...
Code structure

```java
public class Car {
    void break() {
        statement_1;
        statement_2;
    }
    void turn_right() {
    }
}
```

- **source file Car.java**
  - each file holds one class
- **class Car**
  - a class has one or more methods
- **statements**
  - statements are inside methods
- **two methods of class Car**
  - methods belong to a class

Introduction to Java
Writing a class with a main

- In Java, everything goes in a **class**.
- When you run a program, you run a class:
  - load the class then start executing the class's main() method
  - the class MUST have a main() method!
What can we do in a method...

- **Looping...**

```java
while (x > 12) {
    x = x - 1;
}
for (int x = 0; x < 10; x++) {
    System.out.print(x);
}
```

- **Conditional branching...**

```java
if (x == 2) {
    System.out.println("x must be 2");
} else {
    System.out.println("x is not 2");
}
```
What else can we do?

- do-while?
- switch?
- int, long, float, double, boolean...?
- other syntactical stuff?

Read your text books!
To dos

- Homework 1 is ready! Due next week.
  - Start installing programming tools NOW!

- See the course website

- Weekly tests starts next week.
  - Don’t be late or you’ll miss it.

- Prepare for next week test
  - Exercises at the course’s website.