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:
- Introduction to Programming in Java. Ch. 1
- 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, 6, 7, 8 (versions 1.5-1.6…)
  - more powerful
Why Java?

- Java features
  - Object-Oriented
  - Widely used, widely available.
  - Embraces full set of modern abstractions
  - Variety of automatic checks for mistakes in programs

- Java economy.
  - Mars rover.
  - Cell phones.
  - Blu-ray Disc.
  - Web servers.
  - Medical devices.
  - ...

USD 100 billions
5 millions developers
Our choice

- **Java features**
  - **Object-Oriented**
  - Widely used, widely available.
  - Embraces full set of modern abstractions
  - Variety of automatic checks for mistakes in programs

- **No perfect language, we need to choose some.**

- **Our approach:**
  - Minimal subset of Java
  - Develop object-oriented thinking that applicable to many languages

- **OOP is not about the language!**
Compiling and interpreting

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

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

- **file name and class name are identical**
- **HelloWorld application**
- **this is a class**
- **class name**
- **start of the class**
- **method name**
- **public class**
- **HelloWorld**
- **public static void**
- **main**
- **(String[] args)**
- **System.out.println("Hello, world");**
- **public, so that everyone can access**
- **more about that later**
- **end of the class**

**Introduction to Java**

9
Compile and run at command-line

- To compile HelloWorld.java, type `javac HelloWorld.java`
- To run HelloWorld.main(), type `java HelloWorld`

D:\Java> javac HelloWorld.java
D:\Java> java HelloWorld
Hello, world

- Or use an IDE like IntelliJ, Eclipse…
- Or a build tool like Maven, Gradle…

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

Introduction to Java
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
  - ...

Introduction to Java
Code structure

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

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

```java
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 text books!