Software Construction
Basics of Java

Dhammika Elkaduwe

Department of Computer Engineering
Faculty of Engineering

University of Peradeniya

ILOs

- Programming paradigms
- Key words: public, static, class
- Naming of classes, source files, methods
- Source code, byte-code, JVM
- How to write, compile and run a Java application
- Adding comments
- Basics of OOP

Programming Paradigms

Programming paradigms:

- **Imperative programming**:
  - program is written as sequence of instructions which will achieves the required task. (Ex: C, Pascal) (*done in CO222*)
- **Object Oriented Programming**:
  - program is written as interaction between objects. Objects model the *real-world*. (Ex: Java, C++, Small-talk) (*This course*)
- **Declarative**:
  - program is written as description of what has to be done. (Ex: SQL) (*in CO226: Database Systems*)
- **Functional**:
  - Everything is a function. (Ex: Haskell, OCaml)
  - *CO523: Programming Languages*
  - May be some here

Object Oriented Programming: *OOP*

- Object: an entity that models a real-world item, by capturing its state and the behaviour.
- Behaviour: is modelled via *methods*.
- State: is modelled as fields/attributes of an object. These are variables (with restrictions on who can access)
- Class: gives the blueprint for making an object. Instance of a class is an object.

Hello World in Java

First Java Program.

```java
import java.lang.*;

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

- *class*: everything must be placed inside a class. *class* is a keyword and *Hello* is the name of the class.
- If the class is *public* then file name should be the same with *.java* extension. Example, the above code should be in *Hello.java* file.

Hello World in Java

First Java Program.

```
import java.lang.*;

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

Compiling:

- Java is a compiled language.
- To compile type `javac Hello.java`, where *javac* is the Java Compiler program and *Hello.java* is the Java *source code* file.
- Output of the compilation is called the *byte-code*.
- The *byte-code* will be in a *Hello.class* file, where *Hello* is the name of the class.

Hello World in Java

```
import java.lang.*;

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

Running:

- Java runs on top of a *virtual machine* or *JVM*.
  - Byte code can run on any computer which has the same JVM
  - Became popular with the Internet; compile-once run-anywhere. You can distribute the code
- To run the code type *$ java Hello*, where *java* is the JVM program and *Hello* is the class name (without the extension).

side track: `try $ file Hello.class and $ file ‘which ls‘. Note all OS/CPU information in the latter`

Structure of a Java program

- Document section (what the code does, who wrote etc.)
- Package statements
- **Import statements**
- Interface statements
- **Class definitions** (You can have more than one class in a single file as well. Not encouraged).

Hello World: Explained

```
import java.lang.*;

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

- **public**: *access modifier*. Specifies who can access the method.
- **static**: defines the scope. Static methods belongs to a class (not to an object). Static methods can be invoked without an object. (Hence) *main* method has to be static.
- **void**: the *main* method is void return.
- **main**: name of the method. This is the entry point to your code.
- **String [] args**: Commandline arguments are passed to the main method as an array of *Strings*.

side track: `Java does not pass the file name as an argument to the main function.`

Hello World: Explained

```
import java.lang.*;

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

`System.out.println(‘‘Hello World’’);`

- *System.out* is an object which provides a method called *println*.
- *println* displays a single value on the standard output by converting it to a *String*.
- *System.out* also provides a *print* which is very similar to *printf* is C. *print* does not break the line like *println*.

Naming rules in Java

- *class* name should be a noun, with the first letter capital. If you have more than one word make first letter of each word capital.
- *method* name should be a verb, with the first letter simple. If you have more than one word make first letter of each internal word capital.

Comments

```
/**
* How to define a variable in Java
* How to add comments, single line and muti-line
*/

// This is a single line comment

class VariableExample {

  public static void main(String [] args) {
    int i = 10; // variable i, who’s value is 10

    System.out.println("Value of i = " + i);
    // value of i gets converted to string and concatenated
  } // end main

} // end class
```

Goes without saying comments are a must!

Other stuff

- You can write a Java program using any text editor that supports ASCII
- You can compile the source from the command line and generate a class file
- You would be introduced to an IDE – Integrated Development Environment that would make all this much easy!
- That would the first lab class.

ILOs: Revisited

- Programming paradigms
- Key words: public, static, class
- Naming of classes, source files, methods
- Source code, byte-code, JVM
- How to write, compile and run a Java application
- Adding comments
- Basics of OOP

Next class: Variable, conditional executions.