Java Revisited (1)
CS212: Data Structure
1st semester 2011/12
Lecture 1

Today

• Object Oriented Programming (OOP): What, Why, How?
• Analyzing and Designing OO Programs (Objects & Classes)
• Java Syntax, Java Program Skeleton
• Analyzing and Designing a Program
• Preparing Classes.
OOP: What?

- Thinking of Objects!
- What is the form of “Things” in the world?
- Define an Object!!

It’s a thing that have a status and can perform functions

OOP: What?

- An approach to the solution of problems in which all computations are performed in the context of objects.
  - The objects are instances of classes, which:
    - are data abstractions
    - contain procedural abstractions that operate on the objects
  - A running program can be seen as a collection of objects collaborating to perform a given task
OOP: Why?

- Object-Oriented Programming consists of 3 primary ideas:
  - **Data Abstraction and Encapsulation**
    - Operations on the data are considered to be part of the data type
    - We can understand and use a data type without knowing all of its implementation details
      - Neither how the data is represented nor how the operations are implemented
      - We just need to know the interface (or method headers) – how to “communicate” with the object
    - Compare to functional abstraction with methods
  - **Inheritance**
    - Properties of a data type can be passed down to a sub-type – we can build new types from old ones
    - We can build class hierarchies with many levels of inheritance
  - **Polymorphism**
    - Operations used with a variable are based on the class of the object being accessed, not the class of the variable
    - Parent type and sub-type objects can be accessed in a consistent way
OOP vs. Procedural Programming

- **Procedural paradigm:**
  - Software is organized around the notion of *procedures*
  - *Procedural abstraction*
    - Works as long as the data is simple
  - *Adding data abstractions*
    - Groups together the pieces of data that describe some entity
    - Helps reduce the system’s complexity.
    - Such as *Records* and *structures*

- **Object oriented paradigm:**
  - Organizing procedural abstractions in the context of data abstractions
Bird, Human and Fish are all Animals
However, an Animal is not necessarily a Bird, Human or Fish
How To Define Objects in a Program?
How dose objects interact?
Classes What are they?
Skeleton of a class

Objects = nouns
Functions to be encapsulated
Objects interaction

Classes

- A class:
  - A unit of abstraction in an object oriented (OO) program
  - Represents similar objects
    - Its instances
  - A kind of software module
    - Describes its instances’ structure (properties)
    - Contains methods to implement their behavior
Class Structure

- Two Main Sections
  - Variables: can be a simple data type or another Class
    - Represent the State of the Class
    - Define Data represented in an Class
    - Associations
  - Operations: A procedural abstraction used to implement the behaviour of a class.

Skelton of a Class

```java
class Name {
    // Attributes
    Type Name;

    constructor {
    }

    setter {
    }

    getter{
    }

    operations{
    }
}
```
What is Java?

- Java is a programming language created by James Gosling from Sun Microsystems in 1991. The first public available version of Java (Java 1.0) was released 1995.
- The target of the Java programming language was that a program can be written once and then runs on multiple operating systems.
- The Java programming language consists out of a Java compiler, the Java virtual machines, and the Java class libraries.
- The Java virtual machine is a software implementation of a computer that executes programs like a real machine.
- The Java virtual machine is written specifically for a specific operating system.

Why Java?

- Java tries to deliver the promise of „Write once, run everywhere“
- Characteristics:
  - Platform independent
  - Object-orientated programming language
  - Strongly-typed programming language
  - Interpreted and compiled language
  - Automatic memory management
  - Single inheritance
- The Java programming language is actively developed via the Java Community Process (JCP)
- Watchout: Java is case-sensitive!!!
Architecture of Java Applications

- Java applications are written as text files.
- The Java compiler creates platform independent code which is called bytecode.
- Bytecode can be executed by the Java runtime environment.
- The Java virtual machine is a program which knows how to run the bytecode on the operating system the JRE is installed upon.
- The JRE translates the bytecode into native code, e.g. the native code for Linux is different than the native code for Windows.

How Dose It Look?

```java
public class Hello {
    public static void main(String args[])
    {
        System.out.println("Hello World");
    }
} /* end of program */
```
Java Rules

- name of class is same as name of file (which has .java extension)
- body of class surrounded by `{   }
- this class has one method called main
  - all Java applications must have a main method in one of the classes
  - execution starts here
  - body of method within `{   }
- all other statements end with semicolon ;

Java Rules

- keywords appear in bold
  - reserved by Java for predefined purpose
  - don’t use them for your own variable, attribute or method names!
- public
  - visibility could be private
- static
  - the main method belongs to the Hello class, and not an instance (object) of the class
- void
  - method does not return a value
Variables and data types

String name="ALi";

- name is a variable of type String
- we have to declare variables before we use them
- unlike C, variables can be declared anywhere within block
- use meaningful names numberOfBricks
- start with lower case
- capitalise first letter of subsequent words

Data types

- **int** 4 byte integer (whole number)
  - range \(-2147483648\) to \(+2147483648\)
- **float** 4 byte floating point number
  - decimal points, numbers outside range of int
- **double** 8 byte floating point number
  - 15 decimal digits (float has 7) so bigger precision and range
- **char** 2 byte letter
- **String** string of letters
- **boolean** true or false (not 1 or 0)
System output

- Java provides print methods in the class `System.out` (don’t need to import)
  - `println(name);`
    - prints out what is stored in `name`, then goes to a new line
  - `print(name);`
    - prints out what is stored in `name`, but does not start a new line
  - `print("My name is " + name);`
    - put text in quotes
    - use `+` to print more than one item

Methods in Java

- methods break down large problems into smaller ones
- your program may call the same method many times
  - saves writing and maintaining same code
- methods take parameters
  - information needed to do their job
- methods can return a value
  - must specify type of value returned
Example method

```
public static int addNums(int num1, int num2)
{
    int answer = num1 + num2;
    return answer;
}
```

Method signature

```
visibility [static] returnType methodName(parameterList)
```

- **visibility:**
  - `public`
    - accessible to other objects and classes
  - `protected`
    - accessible to classes which inherit from this one
  - `private`

- **static keyword:**
  - use when method belongs to class as whole
    - not object of the class
Method signature

visibility [static] returnType methodName(parameterList)

- **return type:**
  - specifies type of information returned
  - can be a simple type
    - int, float, double, char, String, boolean
  - or a class
  - if nothing returned, use keyword **void**

- **method name:**
  - use meaningful name which describes what method does!

- **parameter list:**
  - information needed by method
  - pairs of **type name**
  - examples:
    - addNums(int num1, int num2)
    - drawPerson(boolean isBald, String name, int numEarrings)
  - use empty brackets if method has no parameters
    - printHeadings()
Method body

- use curly brackets to enclose method body
- all your code goes in here
  - write it so the method does what you intended
- last line should return a value of appropriate type
  - must match type in method header
  - nothing is executed after `return` statement
  - if method returns `void`, can omit `return` statement
    - method will automatically return at closing }

Calling a method

- methods will not run unless called from elsewhere
  - a statement in `main()` method could call another method
  - this method could call a third method ...
- class methods are called with the form: `ClassName.methodName(parameters)`;
  - omit `ClassName` if called in same class
- method name and parameters must match the method signature
- if the method returns a value, it can be stored in a variable or passed to another method
public static void main(String args[]) {
    int input;
    input = Console.readInt("Number? ");
    System.out.print("Your number plus 3 is ");
    System.out.println(addNums(input, 3));
}

Extending Classes

- Inheritance in Java is implemented by extending a class
  public class NewClass extends OldClass {
    ...
    ◦ We then continue the definition of NewClass as normal
    ◦ However, implicit in NewClass are all data and operations associated with OldClass
      ◦ Even though we don’t see them in the definition
ToDo

- Read Chapter 1 of the Textbook.
- Install eclipse or any java editor you fancy.
- Start programming …