JAVA

Lecture 1 - Fundamentals

LECTURES

- Monday and Tuesday. Paradigms, Applications, OOP (theory), Concepts, Language, OOP (application), Exam prep

- Based on text book
-

- Exam is mostly knowledge based
- Coursework is mostly comprehension and application

ROUGHLY 7 LECTURE TOPICS

- 1. Fundamentals
- 2. Classes, Objects, Methods
- 3. Inheritance
- 4. Interfaces
- 5. Exception Handling and IO
- 6. Packages
- 7 Generics? IDE? GUI?

- Chapters 1-11,14? of "Java Programming: A Comprehensive Introduction, Herbert Schildt and Dale Skrien, 2013."

LEARN BY DOING - LABS

- Lectures will mostly be **knowledge** and **theory**
- Labs provide the **application** and **comprehension**
- Please come to your designated lab!
- First 2 weeks of labs simple introduction,
- We will have 3-4 support staff in the lab (or online!)

JAVA ON INTERNET

- This is a **basic** introduction to Java and OOP, assumes no Java experience
  - If you already have experience, then use the time to advance it
- Everything **basic** has already been done and is available on the internet
- Everything you get asked to do in labs will probably be google-able..
- ..but you will learn more by trying it yourself

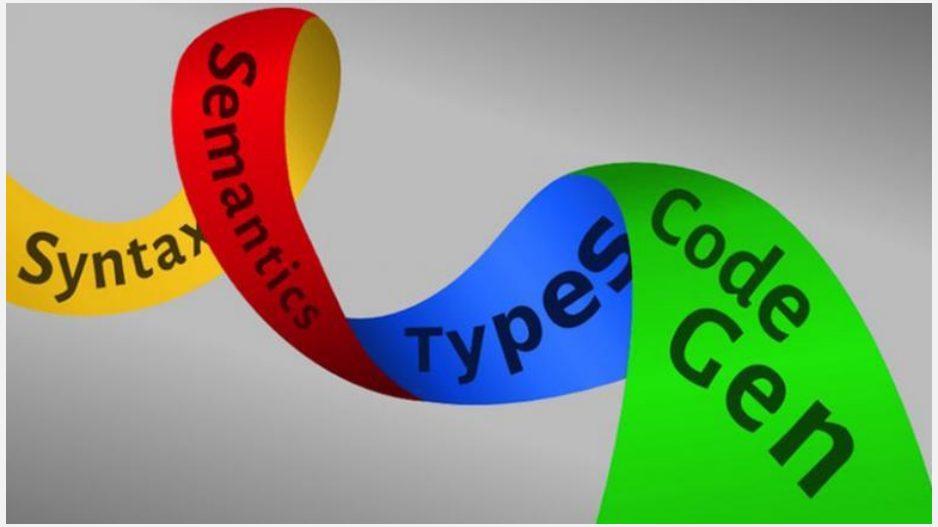
- Coursework (15%) will be 'open book' assignment, short delivery,…(but not googleable!)

JAVA

- Java is a high-level language conceived by James Gosling in 1991. (Originally called Oak and then Green)...*I was doing Fortran and Hypertext*
- It was named "Java" in 1995 (apparently after the coffee they drunk)
- Its syntax is similar to the older languages C and especially C++
- It is an important language for the Internet (Servlets, Apps…more recently Machine learning, simulation)
- Used be 'memory hungry' and slow…
  - Applets transmit full JAR file over the network and load the VM
  - Interpreted not compiled
  - Classloading was slow
- ..…less so these days, Just In Time (JIT) and Java version iteration has improved speed and memory consumption, Memory cheaper, end of Moores law

JAVA COMPILERS AND THE JVM

- Java compilers do not generate machine code for a CPU but for a Java Virtual Machine (JVM)
- The JVM machine code (called bytecode) is executed by a JVM interpreter program on each computer
- Compiling to Java bytecode is easier than compiling to machine code
  - only one format to target
  - JVM is a much simpler machine
  - more convenient to program than real CPUs
    - designed in tandem with the Java language
  - No optimization (this is left to the JIT compiler)
  - No real "Middle end"



OBJECT-ORIENTED PROGRAMMING

- "Code is organized around the data"
- You define the data and the routines that are permitted to act on the data.
- This is one way that Java is object-oriented.
- Object is a software bundle of related state and behavior
- In Java, a class encapsulates the data and the routines acting on the data
- A class is also the blueprint or prototype from which objects are created
- In real life objects might be a Cat, Car or a Pen
  - They have state and behavior (eg. Cats have state (name, color, breed, hungry))
  - and behavior (meowing, stalking, eating..)
- ….more (and more) later

BENEFITS?

- Allows object to remain in control of how it is used by the outside world
- Methods can reject illegal values (eg. Speed that is too high)
- Benefits of bundling code and data together (encapsulation) –
  - Modular/reuse - The source code for an object can be written and maintained independently of other objects. Objects can be used inside the system.
  - Information-hiding. By only interacting with an object's methods, the details of its internal implementation remain hidden
  - Replacing 'faulty' objects is facilitated
  - …..more later….

FIRST PROGRAM

```
/*
This is a simple Java program.
Call this file Example.java.
*/
class Example {
        // A Java program begins with a call to main().
        public static void main(String[] args) {
                System.out.println(“Hello Ningbo");
        }
}
```

WHAT DO WE LEARN FROM PROGRAM 1

- **Commenting** /* comment */ // comment
- **Indenting**
- **{}**
  - Code blocks limit the scope of variables
  - Show hierarchy making the code easier to read, write and maintain.
- **Public** – This means that you can call this method from outside of the current class.
- **Static** - When the JVM makes a call to the main method there is no object existing for the class being called therefore it has to have static method to allow invocation from class.
- **Void** - No particular type of data has to be returned through the function. different things to different platforms.
- **Main** – Method has to be static. The name of the method
- **String args[] or (String [] args)** – These are the arguments of type String that your Java application accepts when you run it.

More Later!

MAIN IN JAVA

- When the program starts up we have no objects
  - We need a starting function/method
- We create a method which is associated with aclass rather than an object (instance of the class)
- So we don’t need an object to exist
- Then we tell Java to ‘run the main() method forou r specific class’

EDITING A JAVA PROGRAM

- Write your program in an Editor of your choice (eg. Notepad++)
- Filename must match class name
- So, the previous program must be saved to a file named Example.java
- For now, make sure the file is saved in plain text (ASCII) format with .java suffix
    - Although Java allows Unicode in the source code.
    - You could potentially use Chinese characters as your variable names

(Java Netbeans and Eclipse are good open source IDEs, but for this module they shouldn't be needed)

COMPILING AND RUNNING THE PROGRAM

- To compile the program, on the command line, type in:
  - *javac Example.java*

- To run the compiled bytecode, type in:
  - *java Example*

A SECOND EXAMPLE

```
class Example2 {
        public static void main(String[] args) {
                int var1; // this declares a variable
                int var2; // this declares another variable
                var1 = 1024; // this assigns 1024 to var1
                System.out.println("var1 contains " + var1);
                var2 = var1 / 2;
                System.out.print("var2 contains var1 / 2: ");
                System.out.println(var2);
        }
}
```

THIRD EXAMPLE

```
class Example3 {
	public static void main(String[] args) {
		int w; // declare an int variable
		double x; // declare a floating-point variable
		w = 10; // assign w the value 10
		x = 10.0; // assign x the value 10.0
		System.out.println("Original value of w: " + w);
		System.out.println("Original value of x: " + x);
		System.out.println(); // print a blank line then now, divide both by 4
		w = w / 4;
		x = x / 4;
		System.out.println("w after division: " + w);
		System.out.println("x after division: " + x);
	}
}
```

SIMILARITIES WITH C

More Later!

- The if Statement
  - Simplest form:
  - if (condition) statement;
- Example:
  - if(3 < 4) System.out.println("yes");
- Relational operators:
  - <, >, <=, >=, ==, !=

EXAMPLE OF IF STATEMENTS

```
class IfDemo {
        public static void main(String[] args) {
                int a, b;
                a = 2;
                b = 3;
                if(a < b) System.out.println("a is less than b");
                // this won't display anything
                if(a == b)
                System.out.println("you won't see this");
        }
}
```

THE **FOR** LOOP

- **General form:**

  for(initialization ; condition ; iteration)

  statement;

- Initialization is normally for setting a loop control variable.
- Condition is for stopping the loop.
- Iteration is normally for updating the loop control variable.

EXAMPLE OF A FOR LOOP

```
class ForDemo {
        public static void main(String[] args) {
                int count;
                for(count = 0; count < 5; count = count+1)
                        System.out.println("This is count: " + count);
                System.out.println("Done!");
        }
}
```

OTHER LOOPS

- While
- Do While
- More advanced loops
  - Labelled for
  - Enhanced for
  - For each

JAVA IDENTIFIERS

- An identifier is a name given to a method, variable, or other userdefined item.
- Identifiers are one or more characters long.
- The dollar sign, the underscore, any letter of the alphabet, and any digit can be used in identifiers.
  - In fact, you can use Unicode characters.
- The first character in an identifier cannot be a digit.
- Upper case and lower case are different: myvar and MyVar are different identifiers.