Week 1 - Lecture Notes

Codegain

March 6, 2013

1 Introduction to Programming

Computers only understand Machine code (very difficult for humans to read and write). Programming languages (like Java, C, C++, etc) are made so software engineers can interact with computers easily.

What do we mean by easy? Well the 3 programs below do exactly the same thing: outputting the words "Hello world". Which do you think is the easiest to understand and code?

Binary:

```
7f 45 4c 46 01 01 01 00  00 00 00 00 00 00 00 00
02 00 03 00 01 00 00 00  54 80 04 08 34 00 00 00
00 00 00 00 00 00 00 00  34 00 20 00 01 00 00 00
00 00 00 00 01 00 00 00  00 00 00 00 00 80 04 08
00 80 04 08 74 00 00 00  74 00 00 00 05 00 00 00
00 10 00 00 b0 04 31 db  43 b9 69 80 04 08 31 d2
b2 0b cd 80 31 c0 40 cd  80 48 65 6c 6c 6f 20 77
6f 72 6c 64
```

Assembly:

```
globl _start
_start:
        movb $4, %al
        xor %ebx, %ebx
        inc %ebx
        movl $hello, %ecx
        xor %edx, %edx
        movb $11, %dl
        int $0x80
        xor %eax, %eax
        inc %eax
```

```
        int 0x80
hello:
        .ascii "Hello world"
```

Java:

```
System.out.println("Hello world");
```

Origins of Java Java was created in 1991 by a team at Sun Microsystems (acquired by Oracle in 2009). Java is an Object Oriented programming language (we will go more in depth about this later!)

- Object Oriented Programming (OOP): Programming methodology that views programs as consisting of objects which perform actions and interact with each other.
- Actions that objects perform are called methods.

Compiler Compilers have 1 main purpose: Translating high-level code like Java into into low-level machine code

- Before translating, the compiler checks to make sure your high-level code is written correctly.
- When you make mistakes in your code, you will get compiler errors. You can only run your program when your code is error-free.

```
Programming Language -> Compiler -> Machine Language -> Executable
```

For Java:

```
Java Code (.java) --javac--> Java Bytecode (.class) --java--> Execute program
```

2 Program Structure

Classes/Methods A Java class name must match its filename. Every Java application program you run must have a main method. When a Java application is executed, its main method is automatically invoked (called).

HelloWorld.java

```
public class HelloWorld {
    public static void main ( String[] args ) {
        // All your code goes here!
        System.out.println ( "Hello World!" );
    }
}
```

General Method Structure

```
class CLASSNAME {
    public static void main(String[] arguments) {
        // WRITE STATEMENTS HERE
    }
}
```

3 Print Statements

Strings are simply words represented by surrounding double quotes ("") in Java.

Syntax:

```
System.out.println( SOME_STRING );
```

Examples of Printing Strings:

```
String name = "Jimmy";
String city = "Cupertino";
System.out.println("Jimmy");
System.out.println(name);
```

4 Variables/Types

Variables are named objects that can store values of different types
Syntax:

```
Type Variable_1, Variable_2,  ;
```

Examples:

```
int myAge, yourAge, hisAge;
boolean isEmpty;
double speed, distance;
```

Java has basic types for integers, floating-point numbers (numbers with decimals), characters, and values for true and false. These are called primitive types.

Type	Value	Possible Values	Size
boolean	truth value	(true or false)	1 byte
char	character	('a', 'b', 'c')	2 bytes
byte	integer	(-128 to 127)	1 byte
short	integer	(-32,768 to 32,767)	2 bytes
int	integer	(-2,147,483,648 to 2,147,483,647)	4 bytes
long	integer	(-9,223,372,036,854,775,808 to -9,223,372,036,854,775,807)	8 bytes
float	floating-point	$(1.40239846 * 10^{-45}$ to $3.40282347 * 10^{+38})$	4 bytes
double	floating-point	$(\pm 4.9406564584e-324d$ to $\pm 1.797693134e+308d)$	8 bytes

Table 1: Primitive Types

5 Assignment

Assign values to variables using the equals sign (=)

Syntax:

```
Optional_Type Variable_1 = Expression_1, Variable_2 = Expression_2,
```

Examples with variable declarations:

```
int score = 0, numberOfTeams = 2, totalScore = numberOfTeams * score;
boolean isGameOver = false;
char firstLetter = 'a';
double roundedPi = 3.14;
```

Examples with assignment after variable declaration:

```
int score;
score = 0;
```

6 Operations

Addition: +
Subtraction: −
Multiplication: ∗
Division: /
Modulo: %

Just like in standard mathematics, you have to follow the Order of Operations:

1. Parentheses

2. Modulo, Multiplication and division
3. Addition and subtraction

Example:

```
public class MathPractice1 {
    public static void main ( String[] arguments ) {
        int x = 4 + 5 * 2;
        System.out.println("x = " + x);

        x = (4 + 5) * 2;
        System.out.println("x= " + x);

        x = 10 % 2 + 2;
        System.out.println("x = " + x);

        x = 10 % 3 + 2;
        System.out.println("x = " + x);
    }
}
```

7 String Concatenation

Using the addition operator (+), we can concatenate Strings together.

Example:

```
int age = 5;
System.out.println("I am " + age + " years old");
```

8 Scanning Console Input

Sometimes, we want a program to interact with the user through the console.

- Programs can pick up lines of input sent by a user using the Scanner class

Example:

```
Scanner inputScanner = new Scanner(System.in);

System.out.println("I will repeat whatever you say! Try me:");

String inputString = inputScanner.nextLine();
```

```
System.out.println("You said: " + inputString);

inputScanner.close();
```

System.in is used to tell the scanner to scan any keyboard input from the user through the console.

- The Scanner.nextLine() function retrieves the line of input from the user and returns a String that you can store in a variable
- Scanner.nextFloat(), Scanner.nextInt() are examples of other functions that can be used when different types of input are expected.
- REMEMBER to always close your Scanner when youre done with it to prevent memory leaks!