CSE1321 J06 Week 1

You will find the some of the example code in my github repository for this section:

> https://github.com/eunsikim/Fall23-CSE1321-J06

Integrated Development Environment (IDE)

- We can technically write code on simple text editors such as Notepad or TextEdit.
- This can be cumbersome on the long run since it involves a lot of manual work other than coding, specially on large and complex programs.
- Most IDEs have lots of dedicated features to help you to focus mostly on writing code.
- IntelliJ and/or Replit

> 💡 More info on how to install IntelliJ in the Lab 1 instruction manual.

Main class and method

- No need to start talking about Object Oriented Programming yet, but classes and methods are a fundamental element of it.

- Whenever we run our code, we need to specify the computer from which point the computer should start executing our program.
- We specify this through the Main method.

```
public class main{
  public static void main(String[] args){
    ...
  }
}
```

💡 The “name” of the should always match the name of the file you are writing your code. For example, if you are writing code in a file called “calculator.java”, then you should call the class “calculator”.

💡 The name of the `main` method is always main. This is because your computer will always look for a method called `main` to start.

- Most of the lines of code you will write throughout the semester will be done “inside” the `main` class and “inside” the `main` method

  💡 By “inside”, we refer that the lines of code will go inside the curly braces ‘{ }’

Command Line Interface

- To make learning and coding easier to learn, we will run our program directly into the terminal or Command Line Interface.
- Whenever you hear the word *Print*, we refer to the action of printing a String of text into the terminal.
- And whenever you hear the word *Read* or *Scan*, we refer to the action of asking the end-user of the program for some input.
- Here is an example:

```
import java.util.Scanner;

public class main{
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
```

```
    int age;

    System.out.println("How old are you?: ");
    age = sc.nextInt();

    System.out.println("Your are " + age + " years old.");
  }
}
```

ExampleA.java

In this program, we print the prompt `"How old are you?: "`, then we wait for the user to input some value, then read the inputted value, and finally store the value into a variable called `age`. Finally, we do another print where we output the age of the user.

⚠️ **From this point forward, assume that every code snippet goes inside the `main` method.**

Introductory concepts

Comments

- It is important to be able to write some information regarding your code in plain readable human language instead of code.
- But in Java you cannot write in English or any other human language, since the computer will not be able to differentiate between Java or English.
- To add comments into a Java file we use `//` for single line comments, and `/* ... */` for multi line comments

```
//Like this

/* Or like
   this
*/
```

- Another example of commenting is what you need to add at the top of your code:

```
/*
  Class: CSE 1321L
  Section: J06
  Term: Fall 2023
  Instructor: Dmitri Nunes
  Name: yourName
```

```
  Lab#: 1
*/

public class Lab1A{
  ...
}
```

Data

- One of the fundamental elements in programming and computing in general is data.
- We represent this data with binary values, 0s or 1s.
- A single value is called a bit.
- 8 bits = 1 byte
- Using binary numbers, we can represent values. For example:
  - 01000001 can be the representation of the decimal number 65.
  - 01000001 can also be the ASCII representation of the uppercase character A.
  - 00111110000000000000000000000000 can represent the decimal value of 0.125 using single-precision floating-point.

Primitive Data types

- We call the following Data Types primitive because they are simple data that can represent a single basic values.
- These Data Types have fixed sizes, these are the following

Data type	Range	Size
byte	-128 to 127	1 byte
short	-32,768 to 32,767	2 bytes
int	-2,147,483,648 to 2,147,483,647	4 bytes
long	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	8 bytes
float	~3.4 x 10^38 to 3.4 x 10^38 (~6-7 decimal digits)	4 bytes
double	~1.7 x 10^308 to 1.7 x 10^308 (~15 decimal digits)	8 bytes
char	Unicode characters (0 to 65,535 or '\u0000' to '\uffff')	2 bytes
boolean	`true` or `false` values	1 bit

Complex Data types

- Complex Data types are Objects. Shortly explained, an Object can have multiple data types (either primitive and/or complex)

Variables

- Whenever we are running a program, there are certain data that we need the computer to temporarily hold.
- To do this, we create variables. Variables can represent a value (or multiple values as the program goes on).
- We can have multiple variables at once or throughout the program, therefore each variable should have an unique identifier or name.
- We also specify what type of value each variable should hold (a single variable can hold only one data type)
- A variable can only be initialized once. Like this:

```
short example = 1;
short example = 4; // This is not valid, we already have initialized a variable called example
```

ExampleB.java

- To Read or Update a variable we can call it using its name

```
example = 4; // Updating the value of example to 4
System.out.print(example); // Printing the value of example to the terminal
```

ExampleB_2.java

- More example of variables

```
int integer = 1;
short shortNumber; // We can initialize variables without an initial value
float floatingPoint = 1.2;
double doubleFloat = 1.4;

char character = 'a';
String string = "Hello World";
```

Concatenation

```
System.out.println("Hello" + " " + "world");
System.out.println("Hello " + someStringVariable);
System.out.println("The sum is " + (5 + someIntegerVariable));
```

ExampleC.java

⚠️ Assume that we already have a String variable called `someStringVariable` and that it has the value of `"world"`.
Same thing for `someIntegerVariable`, which has some integer value.

Assignment vs equality

```
x == y // Is x equal to y?, either true or false
x = y // Assign x with the value of y
```

Quick I/O operations

- Print Statements

```
System.out.print("This prints a string, but does not leave a new line after it has been printed. ");
System.out.println("While this print statement will leave a new line after it has been printed");
System.out.print("See?");
```

ExampleD.java

- Scanner/Read

```
Scanner sc = new Scanner(System.in);
integer = sc.nextInt();
```

Basic Arithmetic

Operators	Symbol	Precedence
Parenthesis	()	First (Highest)
Multiplication, Division, Modulo	*, /, %	Second
Addition, Subtraction	+, -	Third

If there are any conflicts, the order goes from left to right.

```
int sum = 4 + 7;
int sum2 = sum / 2;
System.out.println(sum2);
```

ExampleE.java

Errors

- Syntax and Semantic Errors
  - Syntax: Something in your code is not valid or not recognized as Java programming syntax.
  - Semantic: Your code is valid, but it may not be what you intended to do.

Examples of some errors

- Syntax error.

```
system.out.Println('Hello World'):
```

- Incorrect Logic or Arithmetic.

```
int blueMarbles = 10;
int redMarbles = 4;

System.out.println("There are " + (blueMarbles - redMarbles) + " marbles");
System.out.println("There are " + (blueMarbles + redMarbles) + " more blue marbles than red marbles");
```

ExampleF.java

- Incorrect out format.

```
int numberOfCats = 4;
int numberOfDogs = 6;

System.out.println("There are " + numberOfCats + " dogs and " + numberOfDogs + " cats.");
```

ExampleG.java