**Day\_1\_7\_June**

**Java Basics**

**Define the scope of variables:**

* Scope of a variable is the part of the program where the variable is accessible.
* Java scope rules can be covered under the following categories.
  + There are two types of variables:

1. Member Variables (Class Level Scope) [MemberVar.java]
   * 1. These variables must be declared inside class (w/o any function).
     2. They can be directly accessed anywhere in class.
2. Local Variables (Method Level Scope) [LocalVar.java]
   * 1. Variables declared inside a method have method level scope and can’t be accessed outside the method.
     2. Local variables don’t exist after method’s execution is over.

**Define the structure of a Java class:**

The following pseudocode shows the structure of a java class:

[Modifiers] class <ClassName>{

//fields

[Modifiers] >dataType> <variablename>;

//constructors

[Modifiers] ClassName([ParameterList]) [ExceptionList] {…}

// methods

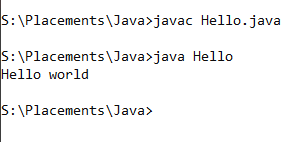
[Modifiers] returnType methodName([ParamList]) [ExceptionList] {…}

// nested types

[Modifiers] (class|interface|enum) NestedTypeName {…}

}

**Create executable Java applications with a main method; run a Java program from the command line; produce console output [Hello.java]**

****

**Import other Java packages to make them accessible in your code**

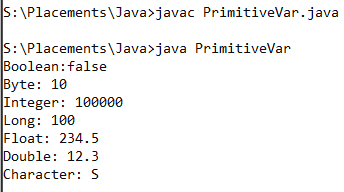
**[firstpackage.java] [secondpackage.java]**

**Compare and contrast the features and components of Java such as: platform independence, object orientation, encapsulation, etc.**

* **Features of Java are:**
  + **Simple:**
    - Java is very easy and the syntax is simple and easy to learn and understand.
  + **Object Oriented:**
    - Java is an object-oriented programming language.
    - Object oriented programming has the following basic concepts:
      * Object
      * Class
      * Inheritance
      * Polymorphism
      * Abstraction
      * Encapsulation
  + **Platform independent:**
    - Java can run on any OS.
    - It is a write once and run anywhere language.
  + **Secured:**
    - No concept of pointers
    - It is run inside a virtual machine also known as JVM.
  + **Robust:**
    - It used strong memory management.
    - It has automatic garbage collection in java which runs on JVM.
    - There is exception handling and type checking mechanism in Java.
  + **Architecture neutral:**
    - In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. However, it occupies 4 bytes of memory for both 32 and 64-bit architectures in Java.
  + **High Performace**
  + **Multi-threaded:**
    - A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads.
  + **Distributed**
  + **Dynamic:**
    - It supports dynamic loading of classes. It means classes are loaded on demand.

**1- Working With Java Data Types**

**Declare and initialize variables (including casting of primitive data types): [PrimitiveVar.java]**

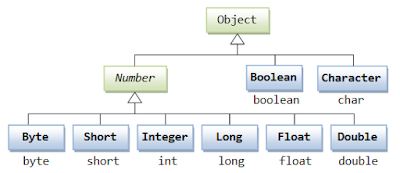
* There are 8 different types of primitive data types:
  + Boolean
  + Byte
  + Char
  + Short
  + Int
  + Long
  + Float
  + Double
* 

**Differentiate between object reference variables and primitive variables**

* There are two types of variables in Java, primitive and reference types.
* Reference variables are not pointers but a handle to the object which is created in heap memory.
* The main difference b/w primitive and reference type is that primitive type always has a value – **it can never be null** but reference type– **can be null which denotes the absence of value.**
* **Default value of primitive type e.g int is 0 while reference variable by default has a null value, which means no reference is assigned to it.**
* **As a developer look out for null pointer exception.**

**Know how to read or write to object fields**

**Develop code that uses wrapper classes such as Boolean, Double, and Integer: [Wrapper.java]**

* 
* Each java primitive type has a corresponding wrapper class. Constructing a wrapper class creates an object.
* **Autoboxing:** Autoboxing is the automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes.
  + **For example:** converting int to an Integer, double to Double.
  + **If the conversion goes the other way, this is called unboxing.**
  + Example code: Character ch = ‘a’;

**3- Operator**

**Using Operators and Decision Constructs**

**Use Java operators; use parentheses to override operator precedence:**

**Test equality between Strings and other objects using == and equals ()**

**Create if and if/else and ternary constructs**

**Use a switch statement**