**JAVA WEEK 1 SUMMARY**

**Day 1 – Introduction to Java**

**Java Components:**

Java Virtual Machine (JVM) - Generally referred as JVM, it’s the primary function is to execute the bytecode produced by compiler.

Each operating system has different JVM, however the output they produce after execution of bytecode is same across all operating systems – hence Java is referred regarded as platform independent.

Program execution phases follows this general sequence write the program, then compile the program and run the program.

1) Writing of the program is of course done by the java programmer like you and me.

2) Compilation of program is done by javac compiler, javac is the primary java compiler included in java development kit (JDK). It takes java program as input and generates java bytecode as output.

3) In third phase, JVM executes the bytecode generated by compiler. This is called program run phase.

Java Development Kit (JDK) - This is a complete java development kit that includes JRE (Java Runtime Environment), compilers and various tools like JavaDoc, Java debugger etc. You would need JDK installed on your computer in order create, compile and run Java program(s).

Java Runtime Environment (JRE) - JRE allows you to run java programs, it includes JVM, browser plugins and applets support. When you only need to run a java program on your computer, you would only need JRE.

**Typical structure of a Java program –** A typical structure of a Java program contains the following elements:

* Package declaration
* Import statements
* Comments
* Class definition
* Attributes
* Methods/behaviours

Package declaration - A class in Java can be placed in different directories/packages based on the module they are used. For all the classes that belong to a single parent source directory, a path from source directory is considered as package declaration. Keyword package is used on package declaration statement.

Import statements - There can be classes written in other folders/packages of your working java project and also there are many classes written by individuals, companies, etc. which can be useful in your program. To use them in a class, you need to import the class that you intend to use. Many classes can be imported in a single program and hence multiple import statements can be written. Keyword import is used on import statement.

Comments - The comments in Java can be used to provide information about the variable, method, class or any other statement. There are two ways to write comments in Java:

* Single line comment - used to comment single line of code. It uses double forward slash (//)
* Multiline comment - used to comment multiple lines of code.

Class definition - A name should be given to a class in a java file. This name is used while creating an object of a class, in other classes/programs. Keyword class is used on Class Definition.

Main method - Execution of a Java application starts from the main method. In other words, it’s an entry point for the class or program that starts in Java Run-time.

Methods/Behaviours - A set of instructions which form a purposeful functionality that can be required to run multiple times during the execution of a program. To not repeat the same set of instructions when the same functionality is required, the instructions are enclosed in a method. A method’s behaviour can be exploited by passing variable values to a method.

**Day 2 – Java Basic Fundamentals**

**What is a variable?**

* A variable is a name given to a memory location. It is the basic unit of storage in a program.
* The value stored in a variable can be changed during program execution.
* A variable is only a name given to a memory location; all the operations done on the variable effects that memory location.
* In Java, all the variables must be declared before use.

**Data types in Java -** Data type defines the values that a variable can take, for example if a variable has int data type, it can only take integer values. Java is a statically typed language; the data type of a variable is known at compile time. This means that you must specify the type of the variable (Declare the variable) before you can use it. In java you have two categories of data type:

* Primitive data types - A primitive data type specifies the size and type of variable values, and it has no additional methods.
* Non-primitive data types - These are the datatypes which have instances like objects. Hence, they are called reference variables. They are primarily classes, arrays, strings or interfaces.

**Operators in Java –** An operator is a character that represents an action, for example – is an arithmetic operator that represents subtraction. Java operators can be classified into:

* Arithmetic operators
* Relational operators
* Bitwise operators
* Logical operators
* Assignment operators
* Misc operators

**Java Classes and objects -** Everything in Java is associated with classes and objects, along with its attributes and methods. For example: in real life, a car is an object. The car has attributes, such as weight and color, and methods, such as drive and brake. A class is like an object constructor, or a “blueprint” or a factory for creating objects. This means without a class no object can be created.

Class – A class can be defined as a template/blueprint that describes the behaviour/state that the object of its type support. To create a class, use the keyword “class”. A class is made up of:

* Constructor
* Methods
* Variables/properties

An object in Java is an instance of a class, i.e., they are created from classes. Objects have states/attributes and behaviours. Example: A dog has states – color, name, breed as well as behaviours – wagging the tail, barking, eating.

**Day 3 – Object Oriented Programming Concepts**

**What is OOP? -** Object-oriented programming (OOP) is a programming paradigm centered on "objects" containing both data and methods. The key goal is to enhance program flexibility and maintainability. OOP consolidates data and behavior in objects, simplifying program comprehension. Java is a well-established OOP language.

Objects in OOP have procedures accessing and modifying their own data fields, introducing the notion of "this" or "self." Programs in OOP are constructed using interacting objects. While OOP languages vary, class-based languages, including Java, C++, C#, Python, and others, are popular.

Notable OOP languages encompass Java, C++, C#, Python, R, PHP, JavaScript, Ruby, and more. Each language follows the object-oriented paradigm, contributing to diverse applications in software development.

**Features of OOP – core features –** The four main features of OOP are:

* Abstraction – a process of hiding the implementation details from the user, only the functionality will be provided to the user. In other words, the user will have the information on what the object does instead of how it does it. In Java, abstraction is achieved using Abstract classes and interfaces.
* Encapsulation - a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes and can be accessed only through the methods of their current class. Therefore, it is also known as data hiding.
* Inheritance - the process where one class acquires the properties (methods and fields) of another. With the use of inheritance, the information is made manageable in a hierarchical order. The class which inherits the properties of other is known as subclass (derived class, child class) and the class whose properties are inherited is known as superclass (base class, parent class).
* Polymorphism – means to process objects differently based on their data type. In other words, it means, one method with multiple implementations, for a certain class of action. And which implementation to be used is decided at runtime depending upon the situation (i.e., data type of the object). This can be implemented by designing a generic interface, which provides generic methods for a certain class of action and there can be multiple classes, which provides the implementation of these generic methods.

**Other features of OOP –** Other features of OOP include:

* Coupling - Coupling refers to the knowledge or information or dependency of another class. It arises when classes are aware of each other. If a class has the details information of another class, there is strong coupling. In Java, we use private, protected, and public modifiers to display the visibility level of a class, method, and field. You can use interfaces for the weaker coupling because there is no concrete implementation.
* Cohesion - Cohesion refers to the level of a component which performs a single well-defined task. A single well-defined task is done by a highly cohesive method. The weakly cohesive method will split the task into separate parts. The java.io package is a highly cohesive package because it has I/O related classes and interface. However, the java.util package is a weakly cohesive package because it has unrelated classes and interfaces.
* Association - Association represents the relationship between the objects. Here, one object can be associated with one object or many objects. There can be four types of association between the objects: One to One, One to Many, Many to One and Many to Many.
* Aggregation - Aggregation is a narrower kind of association. It occurs when there’s a one-way (HAS-A) relationship between the two classes you associate through their objects. For example, every Passenger has a Car, but a Car doesn’t necessarily have a Passenger. When you declare the Passenger class, you can create a field of the Car type that shows which car the passenger belongs to. Then, when you instantiate a new Passenger object, you can access the data stored in the related Car as well.
* Composition - Composition is a stricter form of aggregation. It occurs when the two classes you associate are mutually dependent on each other and can’t exist without each other. For example, take a Car and an Engine class. A Car cannot run without an Engine, while an Engine also can’t function without being built into a Car. This kind of relationship between objects is also called a PART-OF relationship.

**Day 4 – Decision Control**

**IF statement, IF ELSE statement, Nested IF statement:**

* If statement – An if statement consists of a Boolean expression followed by one or more statements.
* If else statement - An if statement can be followed by an optional else statement, which executes when the Boolean expression is false.
* Nested if statement - An if statement can be followed by an optional else if...else statement, which is very useful to test various conditions using single if...else if statement. You have as many else if as the number of your possible conditions.

**Switch statement and the ternary (?) operator:**

Switch statement - A switch statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case. Switch case statement is used when you have number of options (or choices) and you may need to perform a different task for each choice.

The ternary (?) operator - The ternary operator only caters for two outcomes. Like the IF ELSE statement evaluates for the true or false only. It can only be used in place of if else statement.

**Day 5 – Java Modifiers**

**Java identifiers -** Identifiers are the names of variables, methods, classes, packages and interfaces. Unlike literals they are not the things themselves, just ways of referring to them. Identifiers must be composed of letters, numbers, the underscore \_ and the dollar sign $. Identifiers may only begin with a letter, the underscore or a dollar sign. Java uses CamelCase as a practice for writing names of Identifiers (methods, variables, classes, packages and constants). Camel case consists of compound words or phrases such that each word or abbreviation begins with a capital letter or first word with a lowercase letter, rest all with capital.

**Java modifiers -** Modifiers are keywords that you add to those Identifiers to change their meanings. Java language has a wide variety of modifiers, including the following:

* Java Access Modifiers
* Non-Access Modifiers

To use a modifier, you include its keyword in the definition of a class, method, or variable. The modifier precedes the rest of the statement.