Om Sai Ram

Assignment 1

1. Java is the high-level, object-oriented, robust, secure programming language, platform-independent, high performance, Multithreaded, and portable programming language.
2. What is the difference between JDK, JRE, and JVM?

A: **JDK:** JDK stands for 'Java Development Kit' and which is collection of JavaCompiler, JavaLibrary and JVM.

**JavaCompiler** - is used to compile the Source Code and generate Bytecode.

**JavaLib** - will provide built-in 'classes and Interfaces' used for construction.

**JVM** - is used to execute ByteCode.

Define JavaLib?

**JavaLib** is represented with the word 'java'. =>JavaLib is collection of 'packages'. =>packages are the collection of 'classes and Interfaces'. =>'Classes and Interfaces' are collection of 'Variables and methods'.

**JRE:** JRE stands for 'Java Runtime Environment' and which is collection of JavaLib and JVM.(JRE not included with Java Compiler).

In Realtime JRE is used part of Servers(Web Servers and Application Servers)

1. What is JIT compiler?

A: Just-In-Time(JIT) compiler: It is used to improve the performance. JIT compiles parts of the bytecode that have similar functionality at the same time, and hence reduces the amount of time needed for compilation.

1. Java Modifiers(Access Modifiers, Non Access Modifiers)

A: In Java, access specifiers are the keywords which are used to define the access scope of the method, class, or a variable. In Java, there are four access specifiers given below.

**Public:** The classes, methods, or variables which are defined as public, can be accessed by any class or method.

**Protected:** Protected can be accessed by the class of the same package, or by the sub-class of this class, or within the same class.

**Default:** Default are accessible within the package only. By default, all the classes, methods, and variables are of default scope.

**Private**: The private class, methods, or variables defined as private can be accessed within the class only.

**Non-Access Modifiers**

Java provides a number of non-access modifiers to achieve many other functionality.

The static modifier for creating class methods and variables.

The final modifier for finalizing the implementations of classes, methods, and variables.

The abstract modifier for creating abstract classes and methods.

The synchronized and volatile modifiers, which are used for threads.

1. The methods or variables defined as static are shared among all the objects of the class. The static is the part of the class and not of the object. The static variables are stored in the class area, and we do not need to create the object to access such variables.
2. **Inheritance:**

In Java, classes can be derived from classes. Basically, if you need to create a new class and here is already a class that has some of the code you require, then it is possible to derive your new class from the already existing code.

This concept allows you to reuse the fields and methods of the existing class without having to rewrite the code in a new class. In this scenario, the existing class is called the superclass and the derived class is called the subclass.

1. **Interfaces:**

In Java language, an interface can be defined as a contract between objects on how to communicate with each other. Interfaces play a vital role when it comes to the concept of inheritance.

An interface defines the methods, a deriving class (subclass) should use. But the implementation of the methods is totally up to the subclass.

1. Java is an Object-Oriented Language. As a language that has the Object-Oriented feature, Java supports the following fundamental concepts −

* Polymorphism
* Inheritance
* Encapsulation
* Abstraction
* Classes
* Objects
* Instance
* Method
* Message Passing

1. define 'Class' in Java?

A: 'Class' is a 'Structured Layout' in Java, and which generates objects.

'Class' in java is a collection of Variables, Methods and main().

we use 'class' keyword to declare classes in java.

Structure of class: class Class\_name

{

//Variables

//methods

//main()

}

1. define 'Object' in Java?

A: Object is a memory or storage related to a class holding the components of Class.

we use 'new' keyword in Java to create objects.

syntax of Object creation using 'new' keyword: Class\_name obj\_name = new Class\_name();

1. What is the constructor?

A: The constructor can be defined as the special type of method that is used to initialize the state of an object. It is invoked when the class is instantiated, and the memory is allocated for the object. Every time, an object is created using the new keyword, the default constructor of the class is called. The name of the constructor must be similar to the class name.

Following is the syntax of a constructor:

class ClassName {

ClassName() {

}

}

Java allows two types of constructors namely:

* No argument Constructors
* Parameterized Constructors

**No argument Constructors:**

As the name specifies the no argument constructors of Java does not accept any parameters instead, using these constructors the instance variables of a method will be initialized with fixed values for all objects.

**Parameterized Constructors**

Most often, you will need a constructor that accepts one or more parameters. Parameters are added to a constructor in the same way that they are added to a method, just declare them inside the parentheses after the constructor's name.

1. Variables:

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in the memory.

There are two data types available in Java −

* Primitive Data Types
* Reference/Object Data Types

Primitive Data Types

There are eight primitive datatypes supported by Java. Primitive datatypes are predefined by the language and named by a keyword.

1. Byte-8bit
2. Short-16 bit
3. Int-32 bit
4. Long-64 bit
5. Float-32bit
6. Double-64bit
7. Boolean-  
   boolean data type represents one bit of information

There are only two possible values: true and false

1. char-16bit

**Reference Datatypes:**

Reference variables are created using defined constructors of the classes. They are used to access objects. These variables are declared to be of a specific type that cannot be changed. For example, Employee, Puppy, etc.

Class objects and various type of array variables come under reference datatype.

Default value of any reference variable is null.

A reference variable can be used to refer any object of the declared type or any compatible type.

Example: Animal animal = new Animal("giraffe");

1. Loops: A **loop** statement allows us to execute a statement or group of statements multiple times
2. **while loop:**

A while loop statement in Java programming language repeatedly executes a target statement as long as a given condition is true.

Syntax

while(Boolean\_expression) {

// Statements

}

Here, statement(s) may be a single statement or a block of statements. The condition may be any expression, and true is any non zero value.

When executing, if the boolean\_expression result is true, then the actions inside the loop will be executed. This will continue as long as the expression result is true.

1. **for loop:**

A for loop is a repetition control structure that allows you to efficiently write a loop that needs to be executed a specific number of times.

A for loop is useful when you know how many times a task is to be repeated.

Syntax

for(initialization; Boolean\_expression; update) {

// Statements

}

Here is the flow of control in a for loop −

The initialization step is executed first, and only once. This step allows you to declare and initialize any loop control variables and this step ends with a semi colon (;).

Next, the Boolean expression is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop will not be executed, and control jumps to the next statement past the for loop.

1. **do…while loop:**

A **do...while** loop is similar to a while loop, except that a do...while loop is guaranteed to execute at least one time.

Syntax

Following is the syntax of a do...while loop −

do {

// Statements

}while(Boolean\_expression);

Notice that the Boolean expression appears at the end of the loop, so the statements in the loop execute once before the Boolean is tested.

If the Boolean expression is true, the control jumps back up to do statement, and the statements in the loop execute again. This process repeats until the Boolean expression is false.

1. Decision making structures have one or more conditions to be evaluated or tested by the program, along with a statement or statements that are to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.

Java programming language provides following types of decision-making statements. Click the following links to check their detail.

Statement & Description

**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:**

You can use one if or else if statement inside another if or else if statement(s).

**switch statement:**

A switch statement allows a variable to be tested for equality against a list of values.

1. The ? : Operator

We have covered **conditional operator ? :** in the previous chapter which can be used to replace **if...else** statements. It has the following general form −

Exp1 ? Exp2 : Exp3;

Where Exp1, Exp2, and Exp3 are expressions. Notice the use and placement of the colon.

To determine the value of the whole expression, initially exp1 is evaluated.

If the value of exp1 is true, then the value of Exp2 will be the value of the whole expression.

If the value of exp1 is false, then Exp3 is evaluated, and its value becomes the value of the entire expression.

1. **Strings:** which are widely used in Java programming, are a sequence of characters. In Java programming language, strings are treated as objects.

The Java platform provides the String class to create and manipulate strings.

The most direct way to create a string is to write:

String greeting = "Hello world!";

Whenever it encounters a string literal in your code, the compiler creates a String object with its value in this case, "Hello world!'.

1. Java Arrays:

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

To declare an array, define the variable type with square brackets:

String[] cars;

We have now declared a variable that holds an array of strings. To insert values to it, we can use an array literal - place the values in a comma-separated list, inside curly braces:

String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

To create an array of integers, you could write:

int[] myNum = {10, 20, 30, 40};

**Creating Arrays**

You can create an array by using the new operator with the following syntax −

**Syntax**

arrayRefVar = new dataType[arraySize];

1. Method Overloading

When a class has two or more methods by the same name but different parameters, it is known as method overloading. It is different from overriding. In overriding, a method has the same method name, type, number of parameters, etc.

1. Exceptions:

In Java, there are two types of exceptions:

1. **Checked:** are the exceptions that are checked at compile time. If some code within a method throws a checked exception, then the method must either handle the exception or it must specify the exception using *throws*keyword.
2. **Unchecked** are the exceptions that are not checked at compiled time. In C++, all exceptions are unchecked, so it is not forced by the compiler to either handle or specify the exception. It is up to the programmers to be civilized, and specify or catch the exceptions.  
   In Java exceptions under Error and RuntimeException classes are unchecked exceptions, everything else under throwable is checked.