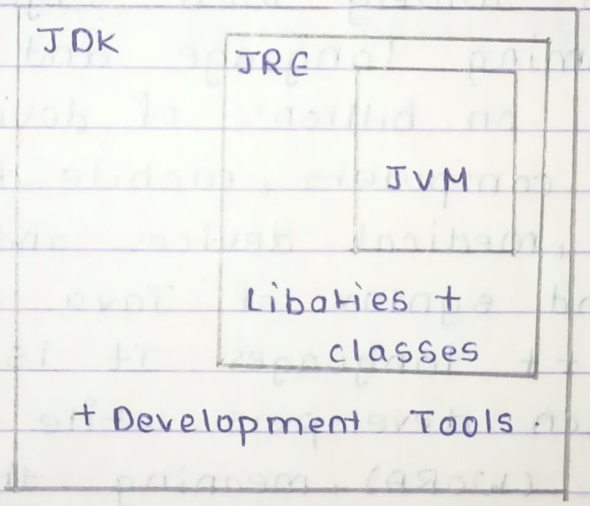
Java:

Assignment no. 1:

Q1.) Write a short note on Java Development kit.

Ans JDK is an acronym for Java Development kit. It is a cross platform software development environment that offers a collection of tools and libaries necessary for developing Java based software application and applets. The JDK is a superset of the JRE (Java Run time environment) and contains everything that is in the JRE, plus tools such as the compilers and debuggers necessary for developing.



(i) JVM

Java virtual Machine (JVM) is a virtual Machine that provides runtime environment to execute java byte code. The JVM doesn't understand Java code directly, that's why you need to compile your .java file to obtain .class files that contain the bytecode

understandable by the JVM.

ii) JRE

The Java Runtime Environment(JRE) provides the libaries, the java virtual Machine, and other components to run applets and applications written in the Java programming language. JRE does not contain tools and utilities such as compilers or debuggers for developing applets and applications.

List and explain the salient features of Java.

Java is widely used object-oriented programming language and software platform that runs on billions of devices, including notebook computers, mobile devices, gaming consoles, medical devices and many others. The rules and syntax of Java are based on the C and C++ languages. It is intended to let application developers write once, and run anywhere (WORA), meaning that compiled Java code can run all platforms that support Java without the need for recompilation. Java was first released in 1995 by James Gosling at Sun microsystems Inc and later acquired by Oracle Corporation.

i) Simple:-

Java is easy to learn and its syntax is quite simple and easy to understand. The confusing concepts of C++ are either left out in Java or they have been re-implemted in a easier way.

Ex:- Pointers and operators overloading are not there in java but were an important part of C++.

ii) Object oriented:-

In Java, everthing is in the form of an object. Java can be easily extended as it is base on object model. Following are some basic concept of OOP's

- Object
- Class
- Inheritance
- Abstraction
- Polymorphism
- Encapsulation.

iii) Robust:-

Java makes an effort to elimanate error prone codes by emphasizing mainly on complie time error checking and runtime checking. But the main areas which java improved were memory Mangement and mishandled Exceptions by ~~int~~ introducing automatic Garbage Collector & Exception Handling.

4) Platform Independent

- Unlike other programming languages as C, C++ etc which are compiled into platform specific machines Java is platform independent programming language means you can write program on any machine and it can run on any machine.
- On compilation Java program is complied into bytecode This bytecode is platform independent and can be run on any machine, plus this bytecode format also provide security. Any machine with Java Runtime Environment can run Java Programs.

5) Security (secure)

When it comes to security, Java is always the first choice. With java secure features it enable us to develop virus free, temper free system. Java program always runs in Java runtime environment with almost null interaction with system OS, hence it is more secure

6) Features of Java.

Java multithreading feature make it possible to write program that can do many tasks simultaneously (at the same time). Benefit of multithreading is that it utilizes same memory and other resources to execute multiple threads at the same time.

for example:- while typing, grammatical errors are checked along.

7) Architectural Neutral

→ Compiler generates bytecode, which have not to do with a particular computer architecture. a Java program is easy to interpret on machine.

8) Portable

→ Java Byte code can be carried to any pl No implementation dependent features. Everyth related to storage is predefined, example size primitive data types.

Q3.) List and explain the components of Java V Machine.

Ans:- JVM is an abstract machine. It is a sp tion that provides runtime environment in which java bytecode can be executed. Similar virtual machines, the JVM creates an isolated space on a host machine. This space can used to execute Java programs irrespective the platform or operating system of the m

The three distinct components of JVM.

1. class Loader
2. Runtime Memory / Data Area
3. Execution Engine.

-) Classloader:-

Classloader is a subsystem of JVM which is used to load class files. Whenever we run the java program, it is loaded first by the # classloader. There are three built-in classloa in java.

) Bootstrap classloader:-

This is the first classloader which is the super class of Extension classloader. It loads the rt.jar file which contains all class files of Java Standard Edition like java.lang package classes, java.net package classes, java.util packages classes, java.io packages classes, java.sql classes etc.

Extension classloader:-

This is the child classloader of Bootstrap and parent class loader of system classloader. It loades the jar files located inside $JAVA_Home/jre/lib/ext directory.

System/Application Classloader:-

This is the child classloader of Extension class loader. It loads the classfiles from classpa By default, classpath is set to current director You can change the class path using "-cp" or "-classpath" switch. It is also known as Applica classloader.

2) Data Area

Class Area stores pre-class structures such as the runtime constant pool, field and method data the code for methods.

i) Heap:-

It is the runtime data area in which objects are allocated.

ii) Stack:-

Java Stack stores frames. It hold locals variable and partial results, and plays a part in method invocation and return.

Each thread has a private JVM stack, created at the same time as thread.

A new frame is created each time a method is invoked A frame is destroyed when it's method invocation completes.

iv) Program Counter Register:-

PC (program counter) register contains the address of the java virtual machine instruction currently being executed.

v) Native method stack:-

It contains all the native methods used in the application.

] Execution Engine:-

i) A virtual processor

ii) Interpreter:-
Read bytecode stream then execute the instructions.

iii) Just-In-Time (JIT) compiler:-
It is used to improve the performance. JIT compiles part of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation. Here, the term "compiler" refers to a translator from the instruction set of a JVM to the instruction set of a specific CPU.

.) Write in detail about different types of operators, in java, category wise quoting their functionality, operands and return type. Give one example statement for each.

:- Different types of operators:-

i) Arithemetic operators
ii) Unary operators
iii) Assignment operators
iv) Relational operators
v) Logical operators
vi) Ternary operators
vii) Bitwise operators
viii) Shift operators
ix) instance of operator

i) Arithemetic operator:-
Perform simple arithemetic operations on primitive data types.

- + : Addition
- - : Subtraction
- % : Modulo
- / : Division
- * : Multiplication

Example:-

```
class Arithematic_Operators
{
    public static void main (String[] args)
    {
        int a = 10;
        int b = 5;

        System.out.println("a + b =" + (a+b));
        System.out.println("a - b =" + (a-b));
        System.out.println("a * b:" + (a*b));
        System.out.println("a / b =" + (a/b));
    }
}
```

- Modulus operator:-
The modulus operator % returns the remainder of a division operation. It can be

applied to floating point types as well as integer types.

example:-

```
class ModulusOperator
{
public static void main(String args[])
{
  int x = 42;
  int y = 10;
  int z = 42 % x % y;

  System.out.println("modulus =" + z);
}
}
```

ii) Unary operators:-

Unary operators need only one operand. They are used to increment, decrement a value.

- - :- Unary Minus.
Used for negating the values.
- + :- Unary plus.
Indicates the positive value(numbers are positive without this, however). It performs an automatic conversion to int when the types of its operand is the byte, char, or short. This is called unary numeric promotion.

- ++ :- Increment operator:-

Used for incrementing the value by 1. There are two varieties of increment operators.

- Post-Increment:-
Value is first used for computing the result and then incremented.
- Pre-Increment:-
Value is incremented first, and then the result is computed.

- -- :- Decremented operator:-
Used for decrementing the value by 1. There are two varities of decrement operators.
- Post-decrement:-Value is first used for computing the result and then decremented.
- Pre-decrement:- The value is decremented first, and then the result is computed.

- ! :- Logical not operator:-
Used for inverting a boolean value.

Example:-

```
Class UnaryOperators
{
  public static void main(String args[])
  {
    int a1 = 10, a2 = 5, a3 = 15, a4 = 12;
    a1++ = 5
    a2 - - = 5
    a3 = a++
    a4 = a--
```

```
System.out.println("Unary plus =" + a1 + "  " + "Unary
    minus =" + a2 + "  " + "Unary Increment =" + a3
    + "  " + "decrement =" + a4);
  }
}
```

iii) Relational Operators: —

These operators are used to check for relations like equality, greater than, and less than. They return boolean results after the comparison and are extensively used in looping statement as well as conditional if-else statements.

==, Equal to returns true if the left-hand side is equal

!=, Not Equal to returns if the left-hand side is not equal to the right-hand side.

<, less than: returns true if the left-hand size is less than the right-hand side.

<=, less than or equal to returns true if the left-hand side is less than or equal to the right-hand side.

> Greater than: returns true if the left-hand side is greater than the right-hand side.

>= Greater than or equal to returns true if the left-hand side is greater than or equal to the right-hand side.

Example:-

```
class RelationalOperators
{
  public static void main(String[] args)
  {
    int a = 10;
    int b = 3;
    int c = 5;
    System.out.println("a>b:" + (a>b));
    System.out.println("a<b:" + (a<b));
    System.out.println("a>=b:" + (a>=b));
    System.out.println("a<=b:" + (a<=b));
    System.out.println("a==b:" + (a==c));
    System.out.println("a!=c:" + (a!=a));
  }
}
```

(iv) Logical Operators

- &&, logical AND: —

Returns true when both conditions are true.

A	B	A&B
0	0	0
0	1	0
1	0	0
1	1	1

- || , logical OR:-
Returns true if at least one condition is true.

A	B	A || B
0	0	0
0	1	1
1	0	1
1	1	1

- ! , logical NOT :-
Returns true when a condition is false and vice-versa.

A	!A
0	1
0	1
1	0
1	0

Example:-

```
class LogicalOperators
{
public static void main(String args[])
{
boolean x = true;
boolean y = false;
System.out.
```

Java Ternary operator

Java Ternary operator is used as one line replacement for if then-else statement and u a lot in Java programming. It is the only cond al operator which takes three operands.

ex:-

```
public class OperatorExample
{
public static void main(String args[])
{
int a = 2;
int b = 5;
int min = (a<b)? a:b;
System.out.println(min);
}}
```

- floating Point

i) float data type:-

The float data type is a single-precision 32-bit floating point. It value range is unlimited. default val 0.0f

ex:- float f1 = 3.610f.

ii) Double data type:-

The double data type is a double-precision 64-bit floating point. It's default value is 0.0d.

ex:- double f2 = 18.03.

- Characters

The char data type is a single 16-bit unicode characters. It value-range lies between '\u0000' (or 0) to '\uffff' (or 65,535). The char data type is used to store characters.

ex:- char letterA = 'A'.

- Boolean Data Type

The boolean data type is used to store only two possible values: true & false. This data type is used for simple flags that track true/ false condition It's size is bit.

ex:-

boolean a = true;

Q6) Explain about memory management in Java with reference to stack and heap.

Ans:- Stack memory in Java is used for static memory allocation and the execution of a thread. It contains primitive values that are specific to a method and references to objects reffered from the method that are in a heap.

Access to this memory is in LIFO - In-first-Out (LIFO) order. Whenever we call a new method, a new block is created on top of the stack which contains values specific to that method like primitive variables and reference to objec when the method finishes execution, its corresponding stack frame is flushed, the flow goes back to the calling method, and space becomes available for next method.

Heap Mem Space in Java

Heap space is used for the dynamic memory allocation of Java objects and JRE classes at runtime. New objects are always created in heap space, and the references to these objects have global access and we can access them from anywhere in the application.

We can break this memory model down into smaller parts, called generation, which are

1. Young Generation - this where all new objects are allocated and age. A minor Garbage collection occurs when this fills up.

2.) Permenent Generation:- This consist of JVM metadata for the runtime classes and application.

Q7) Explain the terms: narrowing, widening

Ans:- i) Widening, also known as upcasting, is a conversion that implicitly takes place in the following situations.

- Widening takes place when a smaller primitive type value is automatically accommodated in a larger/wider primitive data type.

ex:-

```
class Widening
{
  public static void main (String args[])
  {
     byte b = 10;
     short s = b;
     int i = b;
     long l = b;
     float f = b;
     double d = b;
     System.out.println (" short value = " + s + " " +
       " int value = " + i + " " + " long value = " + l +
       " " + " float value = " + f + " " + " double valu
        + d);
  }
}
```

ii) Narrowing is explicitly performed when a wider/bigge primitive type value is assigned to a smaller primitive type value. This is also known as downcast a bigger value to a small primitive value.

ex:—

```
class Narrowing {
    public static void main(String args[])
    {
        double d = 10.5;
        byte b = (byte) d;
        short s = (short) d;
        System.out.println("double value to
                short:" + s + " "
                "double value to byte:" + b);
```

Q8-) Write in detail about static keyword.

Ans:— In Java, there are several reserved keywords for a specific use, we cannot use these pre-defined keywords as variable names or identifiers in a Java program.

The static keyword in Java is one most commonly used keywords. The most fundamental reason for the widespread usage of static keyword in Java is to efficiently manage our program memory.

In general, we first have to make an instance/object of a class in order to access variables or methods declared and defined inside the class. However, there may be times when you simply need to access few of a class's methods or variables and don't want to create a new instance of class merely to access these members. This is when Java's static keyword comes in handy.

static variables are fundamentally considered as global variables. When we declare a static variable in a class, a single copy of that variable is made at the class level is shared across all of it's objects.

Syntax:—

static <data-type> <var-name> = <value>

Q.) Write a short note on access specifiers in java.

Ans:- In Java, there are four types of access specifiers and the name of these access specifiers are given below:

1) public access specifiers
2) protected access specifiers
3) default access specifiers
4) private access specifiers

1) public access specifiers

- "public" is the keyword which is introduced in java.
- The access scope of the "public" is everywhere like in all classes and methods as well.
- If we prefixed "public" keyword with any class, variable or method then it can be accessed by any class or methods

ex:-

```
class A
{
  public void display()
  {
    System.out.println("Software Testing Help!!");
  }
}
class main
{
  public static void main(String args[])
  {
    A obj = new A();
    object = display();
  }
}
```


2) protected access specifiers

- "protected" is the keyword which is introduced in java.
- The access scope of the "protected" is not everywhere and it is accessible in the same class or its child class or in all those classes.
- If we prefixed "protected" keyword with any class, variable or method then it can be accessed by the same class or its child classes.

ex:-

```
class A
{
  protected void display()
  {
    System.out.println("Software Testing Help");
  }
}
class B extends A {}
class C extends B {}
class main
{
  public static void main(String args[])
  {
    B obj = new B();
    obj.display();
    C cobj = new C();
    cObj.display();
  }
}
```

3) default access specifiers

- "default" is the keyword which is introduced in java.
- The access scope of the "default" is not everywhere.
- It is not mandated to prefixed "default" keyword with any class, variable or method because by default class, variable or method is default public in java and it can be accessed by all those classes.

ex:-
```
class BaseClass {
    void display()
    {
        System.out.println("Base class :: Display with default Scope");
    }
}

Class Main {
    public static void main(String args[])
    {
        Base class obj = new BaseClass();
        obj.display();
    }
}
```

4) private access specifiers.

- "private" is the keyword which is introduced in java.
- The access scope of the "private" is not everywhere.
- If we prefixed "private" keyword with any variable or method then it can be accessed only in same class.

ex:-
```
class TestClass
{
    private int num = 100;
    private void printMessage()
    {
        System.out.println("Hello Java");
    }
}
public class main
{
    public static void main(String args[])
    {
        TestClass obj = new TestClass();
        System.out.println(obj.num);
        obj.printMessage();
    }
}
```