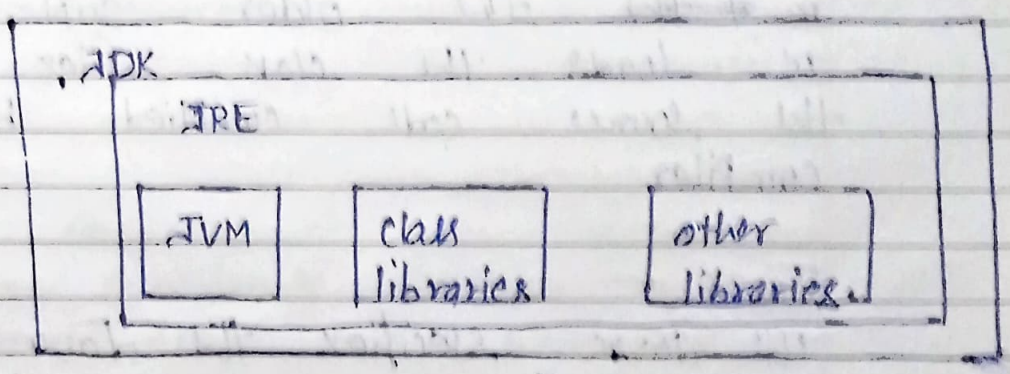
ROLL No= 10070
VISHAL PAL

DATE 6/2/24

Assignment No.1.

Q1. Write a short Note on Java Development kit

Ans Java development kit is a bundle of software development tools and supporting libraries combined with the Java Runtime Environment (JRE) and Java virtual machine (JVM)



Java virtual machine.
JVM is a software tools responsible for creating a run-time environment for the Java source code to run.
The very Powerful feature of Java. "write once and run anywhere" is made possible by JVM. The JVM sits right on top of the host operating system and converts the Java sourced code into Bytecode (machine language). and executes the Program.
Java Run-time Environment -
JRE is a software platform where all the Java source code are executed.

JRE is responsible for integrating the software plugins, jar files, and support libraries necessary for the source code to run.

Components of JDK in Java :-

Java :-
It acts as the deployment launcher in the older SUN, Java.
It loads the class files and interprets the source code complied by the Java compiler.

Javac :-
The javac specifies the Java compiler to convert the source code into bytecode.

Javadoc :-
The javadoc generates documentation for the comments added in the source code.

Jar :-
The jar helps the archivers to manage the jar files in the Packages library.

Q⇒2 List and explain the salient features of Java.

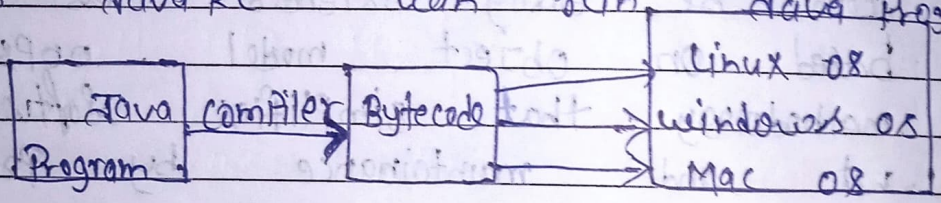
Ans. ① Simple —
Java is designed to be easy to learn. Its syntax is quite simple, clean and easy to understand. Java is inspired by c and c++. There is No need to remove unreferenced objects because there is an Automatic Garbage collection in Java.

② object oriented —
In Java, everything is an object which has some data and behaviour. Java can be easily extended as it is based on object model. oops is a methodology that simplifies software development and maintainance by providing some rules.
Everything in Java is written in terms of classes and objects. Basic concept of oops:

① object
② class
③ Inheritance
④ Polymorphism
⑤ Abstraction
⑥ Encapsulation

③ Platform Independent —

Java is a write once run anywhere language. A Platform is the ~~hardware~~ or hardware or software environment in which a Program runs. Java Provides a software - based Platform. Java code can be executed on multiple Platforms like windows, linux Sun Solaris, macos, etc.
on compilation Java Program is compiled 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 RE can run Java Programs.



④ Architecture - neutral —
Java compiler generates an architecture-neutral object file format which makes the compiled code executable on many processors, with the Presence of Java runtime System.

⑤ Portable —
Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Everything related to storage is Predefined —

Q.3 List and explain the componant of Java virtual machine.

A Components of Java virtual machine:

0 class loader —
class loader is a subsystem of JVM which is used to load class file. whenever a Java source file is compiled it is converted into byte code as a class file. when this class is used the class loader load it into the main memory. The first class that contains the main() method. There are three phases.

* Loader.
This phase loads file from secondary memory into the main memory for execution. loading involves taking the binary representation of a class or interface with a particular name, and generating the original class or interface from that.

* Linking —
Linking is a class or interface involves combining the different elements and the dependencies of Program together. linking include.

* verification —
This phase checks the structural correctness of the "class" file by checking it against a set of constraints or rules.

→ Preparation — The JVM allocates memory for the static fields of a class or interface and initializes them with default values.

→ Resolution — Symbolic references are replaced with direct references in the runtime constant Pool.

→ Initialization —

Initialization involves executing the initialization method of the class or interface. This can include calling the class constructors executing the static block and assigning values to all the static variables.

2 Runtime Data Area —

* Method Area —

Load all the class information used to keep type information about the classes. The method area is created in the virtual machine start-up and there is only one method area per JVM.

* Heap Area —

This is the run-time data from which memory for all class instances and arrays is allocated. There is only one Heap area per JVM.

— Stack Area —

whenever a new thread is created in the JVM, a separate runtime stack is also created of the same time. All local variables method calls and Partial result

are stored in the stack area.

* Program counter Registers—
The JVM supports multiple threads at the same time. Each thread has its own PC register to hold the address of the instruction currently executing JVM instruction. Once the instruction is executed, the PC register is updated with the next instruction.

* Native Method stacks—
The JVM contains stacks that support native methods. These method are written in a language other than Java such as C and C++. For every new threads a seperate native method stack is also allocated.

* Execution Engine—
once the byte code has been loaded into the main memory and details are available in the runtime data area, the next step is to run the Program. The Execution Engine handles this by executing the code present in each class.

→ Interpreter:- The interpreter reads and executes the bytecode instructions line by line.

→ JIT Compiler: The JIT compiler overcomes the disadvantage of the interpreter. when the interpreter finds some repeated code, it uses the JIT compiler.

→ Garbage collector —
collector and remove unreferenced objects from the Heap area. It is the process of reclaiming the runtime unused memory automatically by destroying them.

* Mark: GC identifies the unused objects
* Sweep: remove the object identified.

4. Java native Interface —
It is necessary to use native code. This can be in cases where we need to interact with hardware or to overcome the memory management and performance constraints in Java. Java supports the execution of native code via the Java native Interface.

5. Native Method libraries —
Native method in other libraries libraries that are written in other programming language such as C, C++ and assembly. These libraries are usually present in the form of dll or "so" files. These native libraries can be loaded through JNI

Q.4 write in detail about different type of operators in Java. category wise quoting there functionality, operands and return type Give one example statement for each.

A Arithmetic operation —
The basic Arithmetic operation : addition, subtraction multiplication and division - all behave as expected for all numeric type. The unary minus operator negates its single operands. The unary plus operator simply returns the value of its operend.
operends : int, float, double etc (numeric)
Return type : arithmetic.

```
class BasicMath {
public static void main (String Args[]) {
System.out.println (" Arithmetic operators");

    int a = 1+1;
    int b = b19; a% 3';
    int c = b19
    int d = (-9';
    int e = -d;

System.out.println ("a = "+a);
System.out.println ("b = "+b);
System.out.println ("c = "+c);
System.out.println ("d = "+d);
System.out.println ("e = "+e);
}
}
```

o/p Arithmetic operations

a = 2
b = 6
c = 1
d = -1
e = 1

It also includes

%	Modulus	// a%b
++	Increenment	// a++
--	Decrement	// a--
+=	Addition	assignment
-=	Subtraction	assignment
/=	Division	assignment
%=	Modulus	assignment
*=	Multiplication	assignment.

Bitwise operator —
These operator act upon the individual bits of their operands. They are

~	Bitwise unary NOT
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
>>	Sift right
>>>	shift right zero fill.
<<	shift left.

&= Bitwise AND assignment.
|= Bitwise OR assignment
^= Bitwise exclusive OR assignment
>>= Shift right assignment
>>>= shift right zero fill assignment
<<= shift left assignment.

. operands : Integer type.

```
class Bitlogic {
Public static void main (String args[]){
String binary [] = {"0000", "0001", "0010",
"0011", "0100", "0101", "0110", "0111", "1000",
"1001", "1010", "1011", "1100", "1101", "1110",
"1111", };
int a= 3;
int b= 6;
int c = a|b;
int d = a&b;
int e = a^b;
int f= (~a&b)|(a&~b);
int g = ~a &0x0f;

System.out.println (" a= " +binary [a]);
System.out.println (" b= " +binary [b]);
System.out.println (" a|b= " + binary [c]);
System.out.println (" a&b = " + binary [d]);
System.out.println (" a^b = " + binary [e]);
System.out.println (" ~a&b|a&~b = " + binary [f]);
System.out.println (" ~a = " + binary [g]);
}
}
```

o/p a = 0011
b = 0110
a|b = 0111
a^b = 0010
a&b = 0101
a&~b|~a&b = 0101
~a = 1100

Relational operators—
Determine the relationship that one operand has to the other operands. Any type
Return : boolean value

==	equal to
!=	Not equal to
>	Greater than
<	less than
>=	Greater than or equal to
<=	less than or equal to

Ex- class Relate {
Public static void main(String args[]) {
int a = 4;
int b = 1;
boolean c = a<b;
System.out.println(c);
}
}

o/p false.

Logical operator —
Perform logical operations on operands
operands : only Boolean type
Return : Boolean

OP	result
&	logical AND
|	logical OR
^	logical XOR
||	Short-circuit OR
&&	Short-circuit AND
!	logical unary NOT
&=	AND assignment
|=	OR assignment
^=	XOR assignment
==	Equal to
!=	Not equal to
?:	Ternary if-then-else.

Ex:-
```
class Boollogic {
Public static void main (String args[]) {
boolean a= true;
boolean b= false;
boolean c= a|b;
boolean d= a&b;
boolean e= a^b;
boolean f= (!a&b)|(a&!b);
boolean g= !a;
System.out.println (" a= " +a );
System.out.println (" b= "+b);
System.out.println (" a|b= "+c);
System.out.println (" a&b= "+d);
System.out.println (" a^b : "+e);
System.out.println (" !a&b|a&!b= "+f);
```

System.out.println("!a= "+g);
}
}

O/P
a= true
b= false
a|b = true
a&b = true
!a&b|a&!b = true
!a = false

Ternary operator —

Replace certain type of if-then-else statements.
operands : Boolean expressions
Return : Based on operands (any)
expression1 ? expression2 : expression3

Ex -

```
int x = 10, y = 15;
int result = (x>y) ? x : y;
System.out.println(result);
```

O/P 15

Q⇒5 What are the Primitive data types in Java? Briefly explain their size range and other details.

A. Primitive data types

① Integers -
This group includes byte, short, int and long which are for whole-valued signed numbers. All of these are signed, positive and negative values

long -
size : 64 bit
range : -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
uses: useful for those occasions where an int type is not large enough to hold the desired value.

Int
size : 32 bits
range : -2,147,483,648 to 2,147,483,647
uses: when byte and short value are used in an expression. they are Promoted to int when the expression is evaluated.
So int is the most commonly used integer type. It is the best choice when an integer is needed.

Short
size : 16 bits
range : -32,768 to 32,767

uses = least used Java type longer than byte

Byte :-
size : 8bits
range : -128 to 127
uses = used while working with stream of file or network and with raw binary data.

2. Floating point Types-
- Also known as real Number are used when evaluating expression that require fractional precision

double-
size = 64 bits
range : 4.9e-324 to 1.8e308
uses : It is actually faster than single Precission on some modern Processor that have been optimized for high-speed mathematical calculations.

float -
size = 32 bits
range = 1.4e-045 to 3.4e+038
uses = It specifies the single-precision value that uses 32 bit of storage single precision is faster on some processor and takes half as much space as double Precision.

3. Characters-
char
size : 16 bits

range : 0 to 65,536 (no negative chars)
uses : Represents a single character or it stores the characters. char in Java is designed to hold unicode characters, it can also be used as an integer type.

3. Booleans -
range : true or false
uses : It can have only one of two possible values, true or false. This is the type returned by all relational operations, as in the case of a<b. boolean is also the type required by the conditional expression that govern the control statement such as if and for.

Q.6. Explain about memory management in Java with reference to stack and heap.

A The process of allocation and de-allocation of object is called memory management. The JVM devides the memory into two parts stack memory and heap memory.

Stack —

Stack memory is a physical space allocated to each thread at run time. It is created when a thread creates. memory management in the stack follows LIFO order because it is accessible globally. It stores the variables, reference to objects and partial result.

features.

- variables inside the stack exist only as long as the method that created them is running
- It is automatically allocated and deallocated when the method finishes execution.
- Access to this memory is fast when compared to heap memory.
- If this memory is full. Java throws Java.long.StackOverFlow Error.
- This memory is threadsafe as each thread operates in its own stack

Heap :-
It is created when the JVM starts up and used by the application as long as the application runs. It stores objects and JRE classes. whenever are created object it occupies space in the heap memory while the reference of that object creates in the stack.
It dynamically handles the memory blocks. It means, we need not to handle the memory manually. Java provides the garbage collector that deletes the objects which are no longer being used

Feature :-
If heap space is full, Java throws Java. lang. out of memory Error.

- Access to this memory is comparatively slower than stack memory.
- This memory, in contrast to stack, isn't automatically ~~dealocated~~ deallocated. It needs Garbage collector to free up unused objects so as to keep the efficiently of the memory usage.

Q27 Explain the terms. narrowing, widening.

A Narrowing —
Converting a larger type to a smaller size type
double → float → long → int → char → short → byte
Narrowing must be done manually.
by placing the type in parenthesis in front of the value.

```
class main {
Public static void main(String" args[])
{
 double d= 9.78d
 int i = (int)d.
System.out.println (d);
System.out.println (i);
}
}
```

O/P 9.78
9

- Also known as explisit conversion
- converting higher data type to lower data type.

widening —
Converting a smaller type to a larger type size
byte → short → char → int → long → float → double

Ex- class main {
Public static void main (String args [])
{
int i = 9;
double d = i;
System.out.Println (i);
System.out.Println (d);
}
}

O/P 9
9.0

→ It is also known as implicit conversion
- It is done automatically
- It is safe because there is No chance to lose data. It takes Place when
* Both data ~~types~~ types must be compatible with each other.
- The target type must be larger than the source type.

Q⇒ write in detail about static keyword.

A⇒ The static keyword in Java is mainly used for memory management. The static keyword ~~management~~ in Java is used to share the same variable or method of a Given class. The static keyword belongs to the class than an instance of the class. The static keyword is used for every instance of a class it refer to the common Property of all objects.

The static keyword can be applied to variable.

- When a variable is declared as static, it is known as static variable

- The static ~~keyword~~ variable gets memory only once in the class area at the time of class loading.

- It makes program memory efficient
- static variable are essentially Global variable.
- static variable are created to class level only

- Method —
- When static keyword is applied with any method it is known as static method.
- A static method can be invoked without the need for creating an instance of a class.
- A static method belongs to the class rather than the object of a class.
- A static method can access static data member and can ~~access~~ ~~static data~~ change the value of it
- The most common example of a static method is the main() method
- They cannot refer to this or super in any way.

classes —

- A class can be made static only if it is a nested class.
- Nested static class doesn't need a reference of outer class. In this case, a static class cannot access non-static member of the outer class.

Block —

- It is used to ~~paint~~ initialize the static data member
- It is executed before the main method at the time of class loading.

Q.2) write a short note on access specifiers in Java.

A Access specifiers help to restrict the scope of the class constructor, variable method or data member it provides security, accessibility, etc to the user depending upon the access specifier used with the element.

① Default —
When no access specified is used for a class, method or data member, It is said to be having the default access specifier by default. The data members, classes, or method that are not declared using any access modifiers i.e. having default access specifiers are

declared as Public are accessible from everywhere in the Program. There is No restriction on the scope of Public data members.

③ Private —
- It is specified using the keyword Private.
- The method or data members declared as Private are accessible only within the class in which they are declared.
- Any other class of the same package will not be able to access these members
- Top-level classes or interface can not be declared as Private.

④ Protected.
- It is specified using the keyword Protected.
- The method or data members declared as Protected are accessible within the same Package or subclasses in different Package.
- It Provides more accessibility than the default modifier.