3/2/24

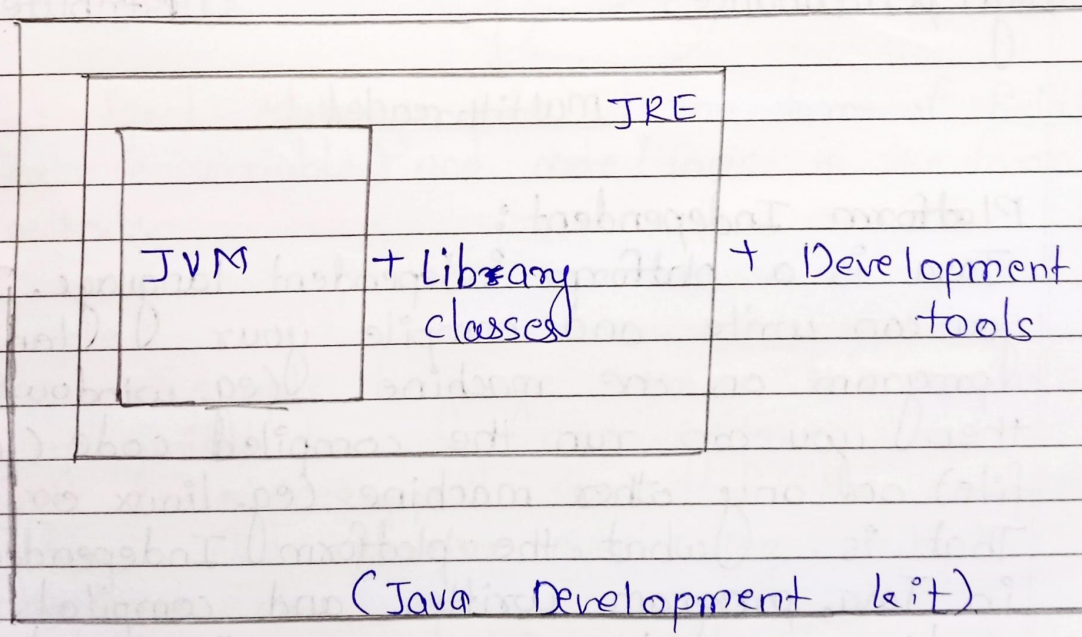
PAGE No.
DATE / /

ASSIGNMENT No:-1

Q.1] Write a short note an Java Development k

Ans

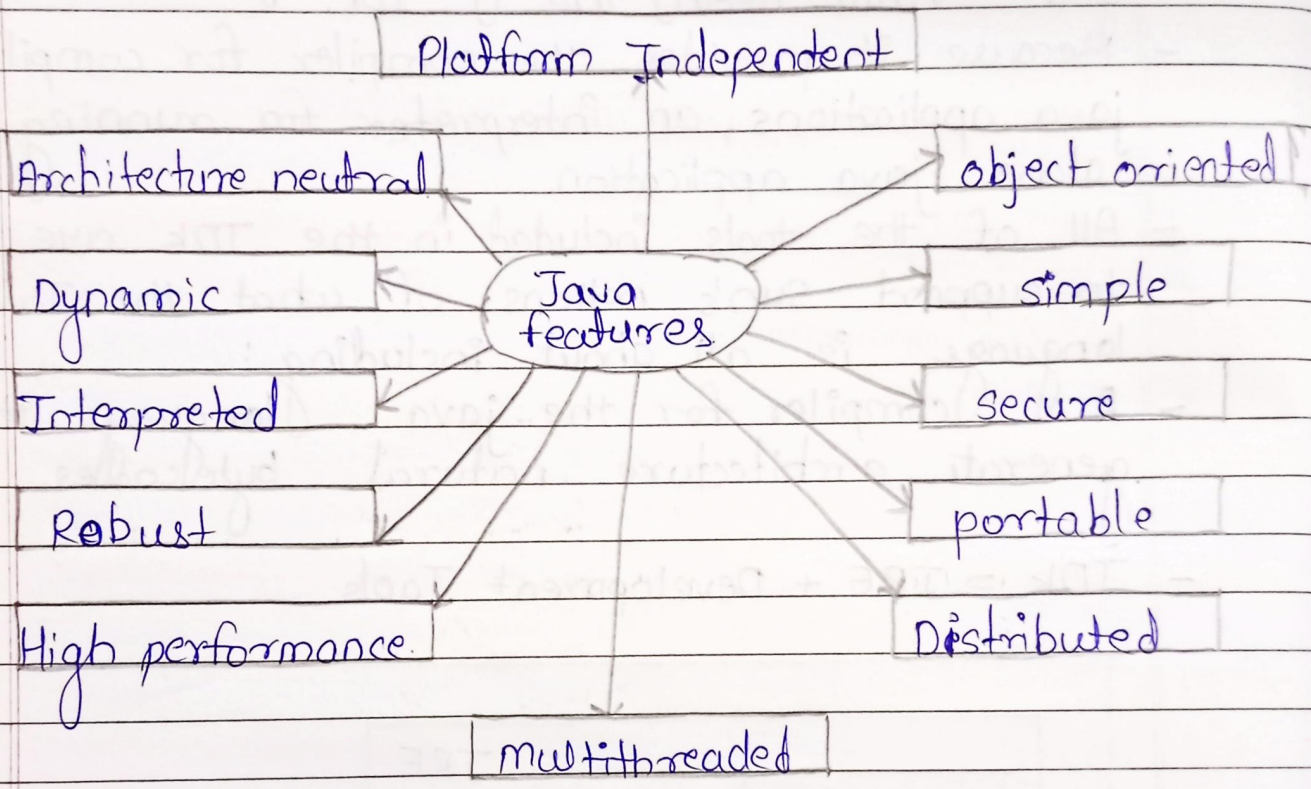
- Java Development kit (JDk):
  - For any java programming that you ~~at~~ wish to you will need the JDk.
  - Because it provides the compiler for compiling the java applications, an interpreter for running stan alone java application.
  - All of the tools included in the JDk are design to support Sun's notions of what the Java language is all about including:
  - A compiler for the java language that generate architecture natural bytecodes.
  - JDk = JRE + Development Tools.



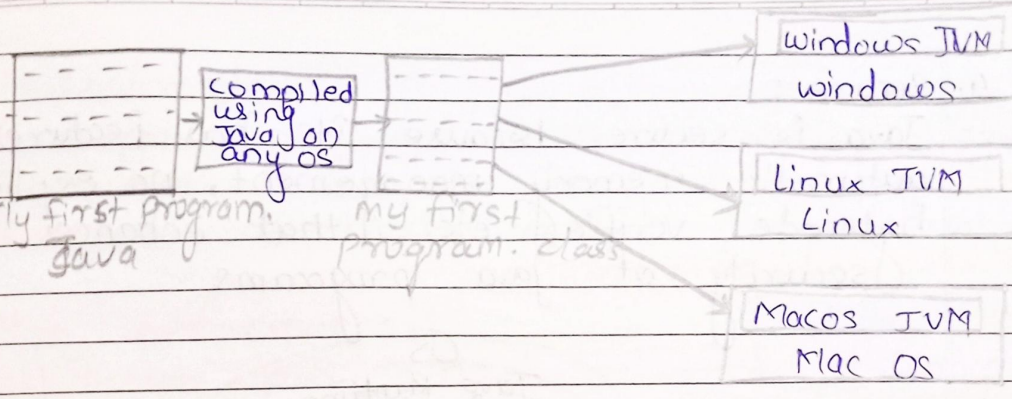
Q.2] List and explain the salient features of Java

Ans

Features of Java : Java has many important features that makes java one of the most useful programming language today.

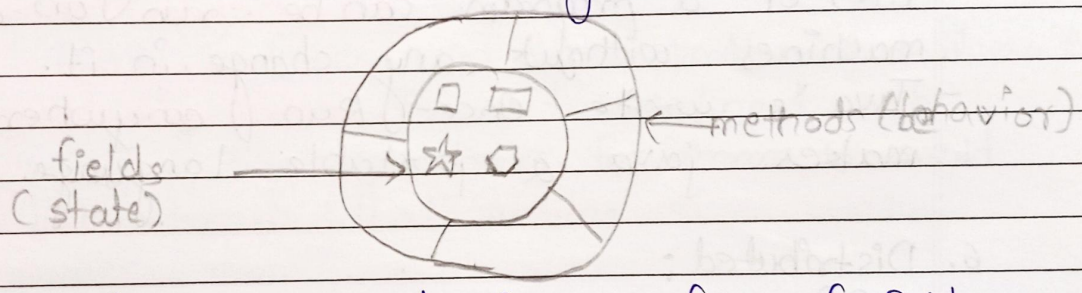


1. Platform Independent :
Java is a platform independent language. In java you can write and compile your ~~language~~ program on one machine (eg. windows) and then you can run the compiled code (class file) on any other machine (eg. linux or macos). That is what the platform Independent mean in Java, program written and compiled on one machine can be run on any other machine.



2]. Object oriented:

- Java is an object oriented programming language because java supports the principle of object oriented programming (OOPs) like Encapsulation, Inheritance Abstraction polymorphism etc.
-

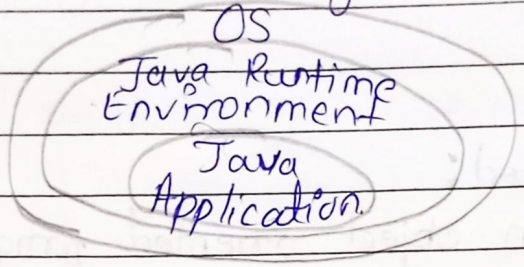


- An object contains data in the form of fields (instance variable) and code/logic in the form of methods.

3. Simple:

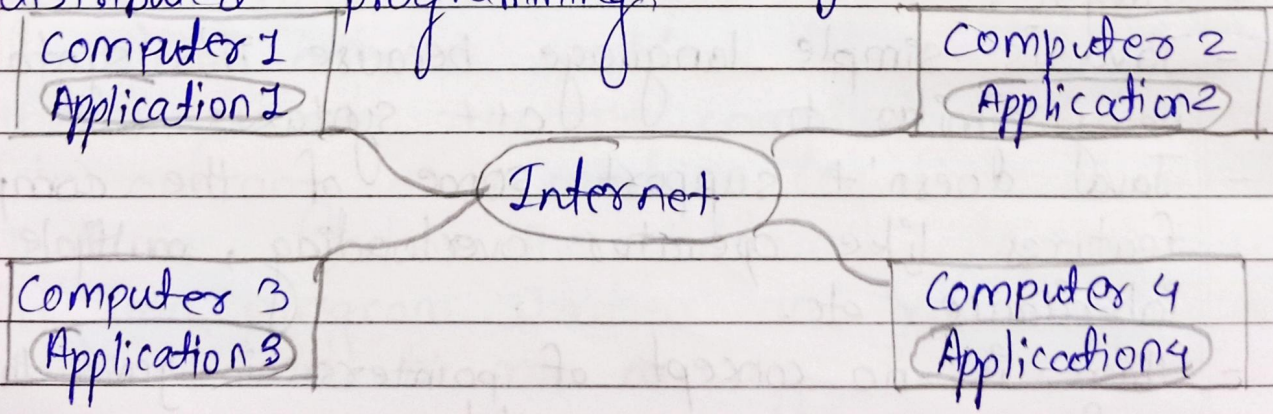
- Java is simple language because it's syntaxes are very similar to c++ syntaxes.
- Java doesn't support some of the complex features like operator overloading, multiple inheritance etc.
- There is no concept of pointers in java that confuses programmers a lot.

4. Secure :
Java is secure because it has features like automatic memory management, no explicit pointer, bytecode verifier etc. that enhances the security of java programs



5. Portable :
- Java is portable because the bytecode (.class file) of a program can be run on different machines without any change in it.
- Java's write Once-Run anywhere features makes java a portable language.

6. Distributed :
Different applications running on different JVM on different machines can interact with each other for sharing data over the internet. RMI and EJB are mostly used for distributed programming.



7. Multithreaded:
An application capable of processing more than one tasks simultaneously / parallely is known as multithreaded application.

8. High performance:
Though java is not as fast as compiled languages like C, C++ but java still has some of the good features like just-in-time compiler, multithreading, garbage collection etc. that increases the performance of java near to compiled languages.

9. Robust:
A robust application is an application that doesn't break easily with incorrect data/input such applications strongly checks for error to avoid application crash at runtime.

10. Dynamic:
The runtime features like memory management, dynamic binding etc makes java a dynamic language.

11. Interpreted:
Java is both compiled and interpreted language since java programs are compiled first using java compiler then at runtime the compiled code (bytecode) is interpreted by java interpreter in order to execute the program.

12. Architecture Neutral:
Java can be run on any computer architecture. As bytecode are platform independent, it also make java as an architecture neutral language. Something which is not dependent strongly on specific system configuration.

Q.3] List and explain the components of java virtual machine.

Ans

- Java Virtual Machine:
  - The Java virtual machine (JVM) is a crucial component of the Java platform.
  - It acts as a runtime engine to run java applications.

- Components of the JVM:

1. Class loader:
It is Responsible for loading classes into the memory of the JVM.

2. Class Area (Method Area):
Memory area where class-related data is stored, such as class bytecode, state variables, and method Data.

3. Heap:
Memory area used for dynamic memory allocation primarily for objects and their related data.

4. Stack:
Each thread running in the JVM has its own stack, storing local variables and method call information.

5. Program Counter (PC) Register:
keeps track of the currently executing instruction in a thread.

6. Native method Interface (JNI):
Enables java code to call and be called by native applications and libraries

7. Execution Engine:
Responsible for executing java bytecode line by line.

8. Java Native Interface (JNI):
Allows java code to interact with application and libraries written in other language.

9. Native method libraries:
Contains native libraries specific to the underlying hardware and operating system.

10. Garbage Collector:
Manages the automatic memory cleanup of unused objects.

#

11. Security manager:
Implements security policies to control the actions that can be performed by Java code.

Q.4] 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.

Ans

Different types of operators in Java.
- Arithmetic Operations
- Relational Operations
- Logical Operations

- Arithmetic Operations :

1. Addition (+):
- functionality : Adds two operands
- Operands : Numeric values (integers, decimal)
- return type : Same type as the operands
e.g. : "int result = 5+3 ;"

2. Subtraction (-):
- functionality : Subtracts the right operands from the left
- Operands : Numeric values.
- return type : Same type as the operands
e.g. : "double result = 10.5 - 3.2;"

3. Multiplication (*):
- functionality: multiplies two operands
- Operands: Numeric values.
- return type: Same type as the operands
  e.g. 'int result = 4 * 7;'

4. Division (/);
- functionality: Divides the left operand by the right
- Operands: Numeric values
- Return type: floating-point type.
  e.g. 'float result = 15/2;'

- Relational Operations:

1. Equality (==):
- functionality: checks if two operands are equal
- Operands: Any comparable values
- return type: Boolean
  e.g.: 'boolean is equal = (a==b);'

2. Inequality (!=):
- functionality: checks if two operands are not equal.
- Operands: Any comparable values.
- return type: Boolean
  e.g.: 'boolean is not equal = (x!=y);'

3. Greater Than (>):
- functionality: Checks if the left operand is less than the right
- Operands: Numeric values.
- return type: Boolean
  e.g.: 'boolean is less than = (m>n);'

4. Less than (<):
- Functionality: checks if the left operand is less than the right.
- Operands: Numeric values.
- return type: Boolean.
  e.g.: 'boolean is less than = (P<q);'

- Logical Operations:

1. Logical AND (&&):
- Functionality: Returns true if both operands are true
- Operands: Boolean values
- return type: Boolean
  e.g.: 'boolean both True = (P&&q);'

2. Logical OR (||):
- Functionality: Returns true if at least one operand is true
- Operands: Boolean values.
- return type: Boolean.
  e.g.: 'boolean either true = (x || y);'

3. Logical NOT (!):
- Functionality: Returns true if the operand is false, and vice versa
- Operands: Boolean values
- return type: Boolean.
  e.g.: 'boolean not True = !is True;'

These are some of the fundamental operations in Java, categorized by their functionality, operands, and return type.

Q.5] What are the primitive data types in java? Briefly explain their size, range and other details.

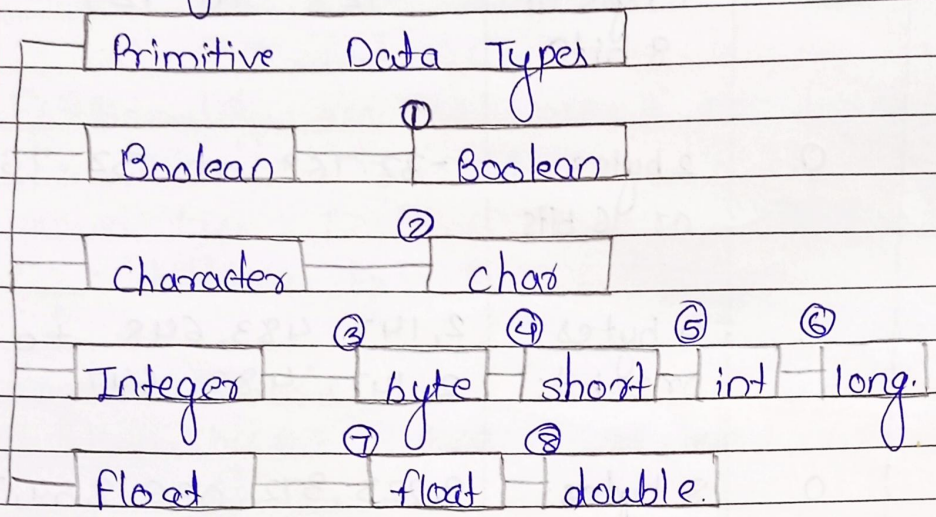
Ans

Primitive Data Types Table - Size, range and Default values

Data type	Default Value	Default size	Range
byte	0	1 byte or 8 bits	-128 to 127
short	0	2 bytes or 16 bits	-32.768 to 32.767
int	0	4 bytes or 32 bits	2,147,483,648 to 2,147,483,647.
long	0	8 bytes or 64 bits	9,223,372,036,854,775,808 to 9,2223,372,036,854,775,80
float	0.0f	4 bytes or 32 bits	1.4e-045 to 3.4e+038
double	0.0d	8 bytes or 64 bits	4.9e-324 to 1.8e+308
char	'\u0000'	2 bytes or 16 bits	0 to 65536

boolean	FALSE	1 byte or 2 bytes	0 or 1

- Primitive Data Types : Primitive data types specify the size and type of variable values. They are the building blocks of data manipulation and cannot be further divided into simpler data types.



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

Ans

- Memory management in java with reference to stack:

- Local variables : The stack holds primitive data types and references to objects.
- Method calls : Each method call creates a new frame on the stack, storing local variables and control flow information.

- LIFO structure : follows a last - In - first - Out structure, where the last method called is the first to finish.

- Memory management in java with reference to heap:

- Object storage : The heap stores objects and their associated data

- Dynamic Memory Allocation : Memory is allocated and deallocated dynamically, allowing for flexible memory management.

- Garbage collection : Java has an automatic garbage collector that identifies and removes unreferenced objects, freeing up memory.

- The stack is used for ~~storing~~ storing local variables and managing method calls, while the heap is used for dynamic memory allocated.

Q.7. Explain the terms : narrowing, widening.

Ans

- Narrowing : This involves converting a larger data type to a smaller data type. This conversion requires explicit casting because it may result in less of data for instance, converting a 'double' to an 'int' involves narrowing and requires explicit casting.

e.g.
```
double larger = 10.5;
int smaller = (int) larger;
```

- Widening: This occurs when you convert a smaller data type to a larger data type. Java performs widening automatically without explicit casting because it is considered a safe conversion.

e.g.
```
int smaller = 5;
long larger = smaller;
```

Q.8] Write in detail about static keyword.

Ans

- The 'static' keyword is used to create class-level variables and methods.
- When a member (variable or method) is declared as static, it means it belongs to the class rather than instances of the class.

- Detailed explanation:

1. Static variables.
2. Static methods
3. Static Block
4. Static Nested classes
5. Static Import.

1. static variables : Static variables are often used for constants or valued that should be consistent across all instances of the class

e.g. : class myclass {
sta' static int staticvariable = 10;
}

2. static methods : static methods cannot access non-static variables directly.

e.g. : class : myclass {
static void staticmethod () {
}
}

3. static Block : The block is useful for initializing static variables or performing other one-time tasks.

e.g. : class myclass {
static {
}
}

4. static Nested classes : static nested classes are associated with the outer class, but they can be instantiated without creating an instance of the outer class.

e.g. : class outerclass {
static class staticNestedclass {
}
}

5. Static Import : Introduced in Java 5, static import allows members (variables and methods) of a class to be used in another class without qualifying them with the class name
e.g. : import static java.lang.Math.PI;
class myclass {
double area (double radius) {
return PI * radius * radius;
}
}

Q.9] Write a short note on access specifiers in java.

Ans:

- Access specifiers are used to specify the access level of a class or its member (data and methods).

- There are four access specifiers in Java :
1. Public 2. Private 3. Default 4. protected

1. Public :
When we declare class members as public, they are accessible from outside the class.

2. Private.
When we declare class members as private, they are only accessible within the class and are not accesible from outside the class.

3. Default:
When we declare class members with no access specifier is considered as default, they are only accessible within the package and are not accessible from outside the package.

4. Protected:
When we declare class members as protected they are only accessible by any class within the same package or by any subclasses of the parent class in which the class members are declared as protected, regardless of whether the subclass is in the same package or a different package.