Q:-1

A)

1) IIB stands for ____

-> Instance initialization Block

2) IDE stands for ____

-> integrated development environment

3) Jvm stands for ____

-> Java virtual machine

4) JRE stands for ____

-> Java runtime environment

Q:-1

B)

1) List and explain JDk tools

-> The jDk contain a private java virtual machine (Jvm) and few other resources such as an interpreter / louder (Java), a compiler (Javac), an archiver (Jar), a documentation generator (Javadoc), etc. to complete the development of a Java Application.

2) List types of operator of Java explain any one in details.

-> operator in Java is a symbol that is used to perform operation on variable and value.

1) Arithmetic operator
2) relational operator
3) Boolean operator
4) Logical operator
5) Bitwise operator
6) Assignment operator
7) Unary operator

* Logical operator

operator	
&&	called Logical AND operator. if both the operand are non-zero, then the condition becomes true.

ex (A && B) is false

||	called Logical OR operator. if any of the two operands are non-zero, then the condition become true

ex (A || B) is true

!	called Logical Not operator. use to reverse the logical state of its operand.

ex ! (A && B) is true

Q-1

(C)

1) explain Jagged Array with example.

-> Jagged Array is a multidimensional array where member array are of diffrent size. for example, we can create a 2D array where first array is of 3 elements, and is of 4 elements following is the example demostrating the concept of Jagged array

Eg

```
class main{

    Public static void main (String [] args)
    {
        int arr[][] = new int [2][];
        arr[0] = new int [3];
        arr[1] = new int [2];
      Int count = 0;

    for (int i = 0; i < arr.length; i++)
    {
      for (int j=0; j < arr[i].length ; j++)
      {
          arr [i][j] = count++;
      }
    }
System.out.Println ("contents of 2D Jagged Array");
```

```
for (int i = 0; i < arr.length; i++)
    {
      for (int j=0; j<arr[i].length; j++)
      {
        System.out.Println(arr[i][j] + " ");
      }
        System.out.Println();
    }
  }
}
```

OutPut :-

Contents of 2D Jagged Array

0 1 2
3 4

2) Explain Varargs with example.

-> the Varargs allow the method to accept zero or multiple arguments.

-> Before Varargs either we use overlouded method or take an array as the method Parameter but it was not considered good because it leads to the meintenance Problem.

-> if we don't know How many argument we will have to Pass in the method, Varagrs is the better approach.

Page
Date

ex4

```
class Test1
{
    static void fun(int a)
    {
System.out.println("Number of arguments: "
                    + a.length);

    for(int i : a)
     System.out.println(i + " ");
     System.out.println();
    }
  Public static void main(String args[])
    {
        fun(100); // one

        fun(1, 2, 3, 4); // four

        fun(); // zero
    }
}
```

Output :- Number of arguments : 1
100
Number of arguments : 4
1, 2, 3, 4
Number of arguments: 0

Q-1(D)

1) what is constructor ? list types of constructor explain any one with example.

-> A constructor is a special type of method in java that is invoked whenever a new object of a class is created using the New keyword. its Primary Purpose is to initialize the object's state (i.e. the values of its instance variables) by assigning them appropriate value.

-> the constructor name must always match the class Name.

-> the constructor do not have an explicit return type.

-> Constructor are called automatically when an object is created using the new keyword.

* types of constructor

1) Default Constructor
2) Parameterized constructor
3) Copy constructor

* Parameterized constructor:

→ You can define constructor with Parameters to Provide initial values for the object's instance Variables during creation.

→ this allows for flexiblility in creating object with diffrent states.

Example

```
Class Person
  {
     Public Person (string name, int age)
     {
        this . name = name;
        this . age  = age ;
     }

  Public static void main (String [] args)
     {
       Person Person1 = new Person ("Alice", 30);
       Person Person 2 = new Person ("Bob", 25);

  System . out. Println (Person1 . name + "is" +
                          Person1. age + "year old.");
System. out. Println ( Person2. name + "is" + person2. age
                          + "year old.");
     }
}
```

2) what is inheritance? list types of inheritance explain any one with example.

-> inheritance in java is a fundamental concept in object oriented Programming that allows you to create new classes that inherit Properties and behavior from existing classes.

-> it Promotes code reusability, code organization and facilities the creation of hierarchical relationships between classes.

* types of inheritance

1) Single inheritance
2) multilevel inheritance
3) Hierarchical inheritance
4) Hybrid inheritance

1) Single inheritance :-

-> A subclass inherits from only one superclass. this is the most common and straightforward type of inheritance

-> By inheriting properties and behaviors, you avoid code duplication and promote code maintainbility.

-) A Subclass can only inherit from one direct superclass using the "extends" keyword.

-) the subclass inherits all non-Private members of the superclass.

exa
```
Class Vehicle {
        String color;
        int wheels;

    Public void move () {

        System.out.Println ("the Vehicle is
                                moving.");
    }
}
class Car extends Vehicle
{
    int doors;

    Public void oPenDoors ()
    {
        System.out.Println(" The car doors are open.")
    }
}
Public class Main {
Public static Void main (String [] args)
{
    Car mycar = new car ();
```

```
        myCar.color = "Red";
        myCar.wheel = 4;
        myCar.doors = 4;

        myCar.move();
        myCar.openDoors();
    }
}
```

-2(A)

1) ____ is the default Package imported in .java A

→ Java.lang Package

2) ____ is the superclass of all Java classes.

→ Object class (java.lang.object)

3) ____ is the Default access specifier for java variables?

→ default (a Package-Private) Accessible from the same Package.

4) Private data members are always inherit in sub class

→ false

Q-2(B)

1) Explain method overriding.

→ method overriding in java is a Powerful mechanism in OOP that allows a subclass to Provide a specific implementation for method already defined in its superclass. this enables flexiblity and Polymorphism in your code.

→ we can use @override annotation in the subclass method declaration to explicitly indicate that you are overriding a method.

- Subclass redefine inherited method
- Same name, Parameter, and return type
- Provide specialized behavior

2) Explain Normal import and Static import.

* Normal import :-

→ Used to import entire classes or interfaces.

→ Allows you to use the class or interface name directly without the Package qualification.

* Static import :-

→ Used to import specific members (fields and methods) from a class

→ Allows you to use the static members directly without the class name or object creation (for static methods)

Page
Date

2-2(c)

1) Explain abstract modifier

→ the abstract modifier in java is keyword used to define abstract classes and methods.

→ it plays a crucial role in achieving abstraction.

* Abstract classes :-

→ An abstract class is a blueprint for creating subclasses that inherit its properties and behaviors

→ it cannot be directly instantialed (meaning you cannot create objects of an abstract class).

→ it servers as a base class that defines a common structure for its subclasses.

→ An abstract class must contain at least one abstract method.

→ An abstract method has no implementation body (curly braces with code). it just declares the method signature (name, parameters, return type).

→ Promote code reusability by defining common functionalities for subclasses.

→ Provide a structure for creating diffrent types of objects that share some common characteristics.

Example

```
abstract class Animal {

Public abstract void makeSound ();
}


class Dog extends Animal {
    @ Override
    Public Void makeSound ()
  {
      System.out. Println ("woof!");
  }
}

Public class Main {

Public Static void main (String [] args)
  {
      Dog myDog = new Dog ();
      myDog. makeSound ();
  }
}
```

Output:-

woof!

2) Explain final modifier.

1) final Variable :-

-> Declares a variable whose value cannot be changed after initialization.
-> ensures data integrity and Prevents accidental modifications.
-> often used for constants that represent fixed values

eg

```
final double PI = 3.14;
```

2) Final method :-

-> Prevents subclasses from overriding a method inherited from a superclass.
-> Useful for methods that represent core Functionality and shouldn't be altered in subclass.

eg

```
class Shape {

    Public final double calculateArea()
    {

    }
}
```

3) final classes:-

-> Prevent inheritance from a class.
-> Useful for utility classes or classes that represent a fixed concept that shouldn't be extended.

exq

```
final class StringUtils {

    Public Static String toUpperCase (String str)
    {
        - - - -
    }
}
```

Q-2(D)

1) What is Package ? list Built-in Packages, explain any one in detail.