

Computer Science - V (Programming in Java)

Unit I Short answers

Q) Write about Java :

- Java is a general-purpose, object-oriented programming language that produces software for multiple platforms.
- Java consists of JVM which is the heart of Java platform. When a programmer writes a Java application, the compiled code (known as bytecode) runs on most operating systems (OS), including Windows, Linux and Mac OS.
- Java derives much of its syntax from the C and C++.
- Java was developed in the mid-1990s by James A. Gosling, a former computer scientist with Sun Microsystems.

Q) What are Java Essentials :

Java is a platform-independent, object-oriented programming language. Java has the following essential features:

Platform-independent : Java programs have WORA(Write Once Run Anywhere) feature. Java programs once written and compiled can be run on any operating system without being rewritten or recompiled.

High-level Language : Java looks very similar to C and C++ but offers many unique features of its own.

Java Bytecode : Bytecode is an intermediate code generated by Java Compiler. The bytecode is executed by the JVM.

Java Virtual Machine (JVM) : JVM is an interpreter for the bytecode. It takes the bytecode as input and executes it.

Q) Explain about Java Virtual Machine(JVM) :

- A Java virtual machine (**JVM**) is a virtual machine that enables a computer to run Java programs on any platform.
- Java was designed to allow programs to be run on any platform without having to be rewritten or recompiled for each separate platform. A Java virtual machine makes this possible.

- When we compile a *.java* file, *.class* file(i.e. byte-code) is generated by the Java compiler. This *.class* file is executed by the JVM to generate the output.
- JVM is a specification that provides runtime environment in which java bytecode can be executed.
- JVMs are available for many hardware and software platforms (i.e. JVM is platform dependent).

Q) Write the steps to create & run Java program :

We need the following 2 softwares to create and run Java programs:

1. Text Editor (Eg. Notepad , Notepad++)
2. JDK (Java Development Kit)

Step 1: Open Notepad

Step 2: Type the Java program. For eg.

```
class Hello
{
    public static void main(String args[])
    {
        System.out.println("Hello World");
    }
}
```

Step 3: Save the file as **Hello.java**. Select file type as “All files” in the working folder **d:\demo**

Step 4: Open the command prompt. Go to Directory **d:\demo**

Step 5: Compile the program using javac command:

```
d:\demo>javac Hello.java
```

Step 6: Run the program using java command:

```
d:\demo>java Hello
```

Q) Explain about Java datatypes

Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java:

1. **Primitive data types:** The primitive data types include boolean, char, byte, short, int, long, float and double.
2. **Non-primitive data types:** The non-primitive data types include Classes, Interfaces, Strings, and Arrays.

Java Primitive Data Types

In Java language, primitive data types are the building blocks of data manipulation. There are 8 basic data types in Java.

Data Type	Size	Default Value	Description
byte	1 byte	0	Stores whole numbers from -128 to 127
short	2 bytes	0	Stores whole numbers from -32,768 to 32,767
int	4 bytes	0	Stores bigger range of whole numbers
long	8 bytes	0L	Stores very big whole numbers
float	4 bytes	0.0f	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits

double	8 bytes	0.0d	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	false	Stores true or false values
char	2 bytes	'\u0000'	Stores a single character/letter or ASCII values

Q) Explain about Type Conversion & Type Casting

Assigning a value of one type to a variable of another type is known as **Type Conversion or Type Casting**.


Example :

```
int x = 10;
byte y = (byte)x;
```

In Java, there are two types of datatype conversion :

- Widening Conversion
- Narrowing Casting

Widening or Automatic or Implicit type conversion:-

byte → short → int → long → float → double

widening

Automatic Type conversion takes place when,

- the two types are compatible
- the destination type is larger than the source type

Example:

```
public class Test
{
    public static void main(String[] args)
    {
        int i = 100;

        long l = i;        //automatic type conversion

        float f = l;        //automatic type conversion

        System.out.println("Int value "+i);

        System.out.println("Long value "+l);

        System.out.println("Float value "+f);

    }
}
```

Output:-

Int value 100

Long value 100

Float value 100.0

Narrowing or Explicit type casting:-

double → float → long → int → short → byte


Narrowing

- When we are assigning a larger type value to a variable of smaller type, then we need to perform explicit type casting.
- Narrowing casting must be done manually by placing the type in parentheses in front of the value.

Example :

```
public class Test
{
    public static void main(String[] args)
    {
        double d = 100.04;

        long l = (long)d; //explicit type casting required

        int i = (int)l;    //explicit type casting required

        System.out.println("Double value "+d);

        System.out.println("Long value "+l);

        System.out.println("Int value "+i);

    }
}
```

Output:-

Double value 100.04

Long value 100

Int value 100

Qn) Difference between do..while and while loop :

do-while	while
It is exit controlled loop	It is entry controlled loop
The loop executes the statement at least once	loop executes the statement only after testing condition
The condition is tested before execution.	The loop terminates if the condition becomes false.

Q) Explain about method overloading .

Method Overloading means a class can have two or more methods with the same name, but each with a different parameter list.

There are two ways to overload the method in java

1. By changing number of arguments
2. By changing the data type

When a method is called, first the method name is matched and then, the number and type of arguments passed to the method are checked.

Overloading allows us to perform the same action on different types of inputs.

Method Overloading is one way of achieving polymorphism in Java.

Example :-

```
class Sum {
    int sum(int x, int y)
```

```

    {
        return (x + y);
    }

    int sum(int x, int y, int z)
    {
        return (x + y + z);
    }

    double sum(double x, double y)
    {
        return (x + y);
    }

    public static void main(String args[])
    {
        Sum s = new Sum();
        System.out.println(s.sum(10, 20));
        System.out.println(s.sum(10, 20, 30));
        System.out.println(s.sum(10.5, 20.5));
    }
}

```

Output :

```

30
60
31.0

```

Q) What is the difference between constructors & methods.

Differences

Constructors	Methods
Do not have any return type not even void	Will have a return type
Have the same name as the class name	May have any name (class name should not be used)
Invoked as soon as object is created and not thereafter	Invoked after the object is created and any no. of times thereafter

Invoked only once in an object's life time	Invoked any number of times in an object's life time
Invoked automatically	Invoked explicitly
Can not be inherited	Can be inherited
Used to initialize objects	Used to perform operations
Cannot be abstract, final, static, synchronized	Can be abstract, final, static, synchronized

Similarities

Constructors	Methods
Can take parameters	Can take parameters
Can be overloaded	Can be overloaded
Can be private, protected, public, default	Can be private, protected, public, default

Q) Explain about “this” keyword.

this is a keyword in Java.

this is a reference variable to the current Object, whose method or constructor is being invoked.

It can be used inside the *Method* or *constructor* of Class.

Uses of this keyword :

- Eliminate confusion between class variables & parameters
- Invoke a constructor from another constructor of the same class (Constructor Chaining)
- Invoke current class method
- Return the current class object
- Pass an argument in the method call

- Pass an argument in the constructor call

Example :-

The most common use of the `this` keyword is to eliminate the confusion between class attributes and parameters with the same name.

```
class Demo

{
    int x;

    Demo(int x)

    {
        this.x = x;
    }

    public static void main(String[] args)

    {
        Demo obj = new Demo(5);
        System.out.println("x = " + obj.x);
    }
}
```

Output:-

x = 5

Q) Explain about Command line arguments.

The command line arguments are the arguments passed to a program at the time when we run it.

They are stored as strings in String array passed to the “args” parameter of `main()` method.

- They can be used to specify configuration information while launching the application.
- There is no restriction on the number of command line arguments. We can specify any number of arguments
- Information is passed as Strings.
- They are captured into the `String args[]` of the main method

Example:

```
class CmdLineDemo
{
    public static void main(String[ ] args)
    {
        for(int i=0;i< args.length;i++)
        {
            System.out.println(args[ i ]);
        }
    }
}
```

Output:-

D:/> java CmdLineDemo One Two Three

One
Two
Three

Q) Explain about Nested Classes.

A class defined within another class is known as Nested class.

Syntax:

```
class Outer
{
    //class Outer members

    class Inner
    {
        //class Inner members
    }
}
```

Advantages of Nested Class

1. It is a way of logically grouping classes that are only used in one place.
2. It increases encapsulation.
3. It can lead to more readable and maintainable code.

Types of Nested Classes :

- Non-static nested classes / Inner classes
- Static nested classes
- Local inner classes
- Anonymous classes

Q) Explain about the “super” Keyword.

In Java, **super** keyword is used to refer to immediate parent class of a child class.

In other words **super** keyword is used by a subclass whenever it needs to refer to its immediate super class.

```
class Parent
{
    String name;
}
class Child extends Parent {
    String name;
    void detail()
    {
        super.name = "Parent";
        name = "Child";
    }
}
```

Uses of “super” keyword:-

1. Child class to access Parent class variables
2. Child class to call Parent class methods
3. Child class to call Parent class constructor

Q) Explain about Method Overriding.

Declaring a method in **sub class** which is already present in **parent class** is known as method overriding.

In method overriding, the child class has the same method as declared in the parent class.

Uses of Method Overriding

- Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.
- Method overriding is used for runtime polymorphism

Rules for Java Method Overriding

1. The method must have the same name as in the parent class
2. The method must have the same parameter as in the parent class.
3. There must be an IS-A relationship (inheritance).

Example :

```
class A
{
    public void show()
    {
        System.out.println("show in class A");
    }
}
class B extends A
{
    public void show()
    {
        System.out.println("show in class B");
    }
}
```

```

class Demo
{
    public static void main(String args[])
    {
        B obj = new B();
        obj.show();
    }
}

```

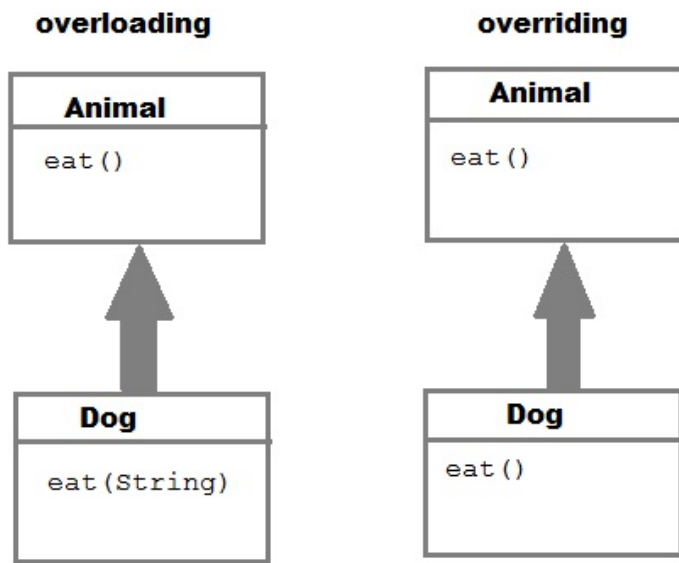
Output:-

show in class B

Q) What is the difference between method overloading and method overriding?

Method overloading and Method overriding seems to be similar concepts but they are not. Let's see some differences between both of them:

Method Overloading	Method Overriding
Parameters must be different and name must be same.	Both name and parameter must be same.
Compile time polymorphism.	Runtime polymorphism.
Increase readability of code.	Increase reusability of code.
Access specifier can be changed.	Access specifier cannot be more restrictive than original method(can be less restrictive).
Inheritance is optional	Inheritance is compulsory



Q) Explain about final Keyword.

The **final keyword** is used to restrict the user.

The final keyword can be applied with the following:

1. variable
2. method
3. class

1. final Variable:-

If we make any variable as final, we cannot change the value of final variable(It will be constant).

2. final Method:-

If we make any method as final, we cannot override it.

3. final Class:-

If we make any class as final, we cannot extend it.

Example :

```
class Demo
```

```
{
```

```

final int x=10; //final variable

void setx()
{
    x=4;    // error : we cannot change final variable
}

public static void main(String args[])
{
    Demo obj=new Demo();
    obj.setx();
}
}

```

Output:-

Compile Time Error

Q) Explain about abstract class and abstract method.

A class which is declared as abstract is known as an **abstract class**.

Abstract class contains one or more abstract methods.

It needs to be extended and its abstract methods need to be implemented.

Example of abstract class

```
abstract class Animal
```

```
{
    .....
}
```

Some points about abstract class:

- An abstract class must be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods .

Abstract Method :

A method which is declared as abstract and does not have implementation is known as an abstract method.

Example of abstract method

```
abstract void printStatus(); //no method body and abstract
```

Q) Explain about abstract class and abstract method.

A class which is declared as abstract is known as an **abstract class**.

Abstract class contains one or more abstract methods.

It needs to be extended and its abstract methods need to be implemented.

Abstract Method :

A method which is declared as abstract and does not have implementation is known as an abstract method.

Example :

```
abstract class Animal
{
    abstract void sound();
}

class Dog extends Animal
{
    void sound()
    {
        System.out.println("Woof");
    }
}
```

```

public static void main(String args[])
{
    Animal obj = new Dog();
    obj.sound();
}
}

```

Output:

Woof

Q) What is the difference between Class and Interface ?

Class	Interface
In class, you can instantiate variable and create an object.	In an interface, you can't instantiate variable and create an object.
Class can contain concrete(with implementation) methods	The interface cannot contain concrete(with implementation) methods
The access specifiers used with classes are private, protected and public.	In Interface only one specifier is used- Public.