JAVA ANSWERS(2 MARKS)

1) What is meant by Binding, what are the binding types?

Answer:

Binding: Binding is a mechanism creating link between method call and method actual implementation. As per the polymorphism concept in Java , object can have many different forms. Object forms can be resolved at compile time and run time.

There are two types of Binding:

i) Static Binding ii) Dynamic Binding

i) Static Binding: If linking between method call and method implementation is resolved at compile time then we call it static binding.

ii) Dynamic Binding: If it is resolved at run time then it is dynamic binding. Dynamic binding uses object to resolve binding but static binding use type of the class and fields.

Example of Static and Dynamic Binding:

```
public class FastFood
{
        public void create()
        {
                System.out.println("Creating in FastFood class");
        }
}
public class Pizza extends FastFood
{
        public void create()
        {
                System.out.println("Creating in Pizza class");
        }
}
public class Main
{
        public static void main(String[] args)
        {
                FastFood fastFood= new FastFood();
                fastFood.create();
                FastFood pza= new Pizza();    //Dynamic binding
                pza.create();
        }
}
```

2) What's the difference between constructors and other methods?

Answer:

Following are the difference between constructor and method.

- Constructor is used to initialize an object whereas method is used to exhibits functionality of an object.
- Constructors are invoked implicitly whereas methods are invoked explicitly.
- Constructor does not return any value where the method may/may not return a value.
- In case constructor is not present, a default constructor is provided by java compiler. In the case of a method, no default method is provided.
- Constructor should be of the same name as that of class. Method name should not be of the same name as that of class.

3) What is the Final Keyword in Java? Give an example.

Answer:

In Java, the *final* keyword is used to denote constants. It can be used with variables, methods, and classes. Once any entity (variable, method or class) is declared *final*, it can be assigned only once. i.e.

- the final variable cannot be reinitialized with another value
- the final method cannot be overridden
- the final class cannot be extended

Example:

```
class Main
{
        public static void main(String[] args)
        {
                final int AGE = 32;             // create a final variable
                AGE = 45;                       // try to change the final variable
                System.out.println("Age: " + AGE);
        }
}
```

In the above program, we have created a final variable named *age*. And we have tried to change the value of the final variable.
When we run the program, we will get a compilation error with the following message.

```
cannot assign a value to final variable AGE
    AGE = 45;
    ^
```

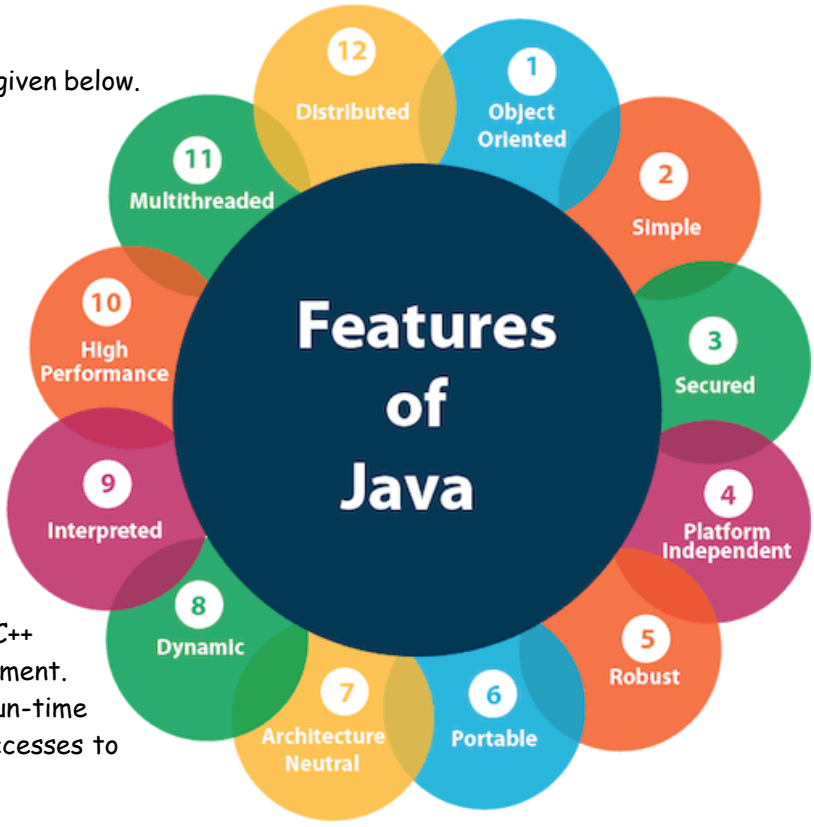
Similarly, final method and final class will also show compilation error.

4) List any four features of java

Answer:

A list of most important features of Java language is given below.

1. Simple
2. Object-Oriented
3. Portable
4. Platform independent
5. Secured
6. Robust
7. Architecture neutral
8. Interpreted
9. High Performance
10. Multithreaded
11. Distributed
12. Dynamic



5) Why is java considered dynamic?

Answer:

Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry an extensive amount of run-time information that can be used to verify and resolve accesses to objects at run-time.

6) Briefly explain any two java keywords?

Answer:

public: Java public keyword is an access modifier. It is used to indicate that an item is accessible anywhere. It has the widest scope among all other modifiers.

return: Java return keyword is used to return from a method when its execution is complete.

7) What do you mean by object?

Answer:

Object – Objects have states and behaviours. *Example:* A dog has states - colour, name, breed as well as behaviours – wagging the tail, barking, eating. An object is an instance of a class.

8) What kind of variables a class consists of? Explain briefly any two?

Answer:

There are three different types of variables a class can have in Java are **local variables**, **instance variables**, and **class/static** variables.

Local Variable:

A **local variable** in Java can be declared locally in **methods**, **code blocks**, and **constructors**. When the program control enters the **methods**, **code blocks**, and **constructors** then the local variables are **created** and when the program control leaves the methods, code blocks, and constructors then the local variables are **destroyed**. A local variable **must be initialized** with some value.

Example:

```
public class LocalVariableTest
{
	public void show()
	{
		int num = 100; // local variable
		System.out.println("The number is : " + num);
	}
	public static void main(String args[])
	{
		LocalVariableTest test = new LocalVariableTest();
		test.show();
	}
}
```

Output:

The number is : 100

Instance Variable:

An **instance variable** in Java can be declared **outside a block**, **method** or **constructor** but inside a class. These variables are **created** when the class **object is created** and **destroyed** when the class **object is destroyed**.

Example:

```
public class InstanceVariableTest
{
	int num; // instance variable
	InstanceVariableTest(int n)
	{
		num = n;
	}
	public void show()
	{
		System.out.println("The number is: " + num);
	}
	public static void main(String args[])
	{
		InstanceVariableTest test = new InstanceVariableTest(75);
		test.show();
	}
}
```

Output:

The number is : 75

Static/Class Variable:

A **static/class variable** can be defined using the **static** keyword. These variables are declared **inside a class** but **outside a method** and **code block**. A class/static variable can be **created** at the **start of the program** and **destroyed** at the **end of the program**.

Example:

```
public class StaticVaribleTest
{
	int num;
	static int count; // static variable
	StaticVaribleTest(int n)
	{
```

```
                num = n;
                count ++;
        }
        public void show()
        {
                System.out.println("The number is: " + num);
        }
        public static void main(String args[])
        {
                StaticVaribleTest test1 = new StaticVaribleTest(75);
                test1.show();
                StaticVaribleTest test2 = new StaticVaribleTest(90);
                test2.show();
                System.out.println("The total objects of a class created are: " + count);
        }
}
```

Output:

The number is: 75
The number is: 90
The total objects of a class created are: 2

9) What is a constructor? What is a destructor?

Answer:

Constructor: A class **constructor** is a special member function of a class that is executed whenever we create new objects of that class. A constructor has exactly the same name as the class and it does not have any return type at all, not even void. Constructors can be very useful for setting initial values for certain member variables.

Destructor: A **destructor** is a special member function of a class that is executed whenever an object of its class goes out of scope or whenever the delete expression is applied to a pointer to the object of that class. A destructor has exactly the same name as the class prefixed with a tilde (~). It can neither return a value nor can it take any parameters.

10)What are access modifiers, explain?

Answer:

The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class. We can change the access level of fields, constructors, methods, and class by applying the access modifier on it.

There are four types of Java access modifiers:

1. **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
2. **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
3. **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
4. **Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

11)What are the core OOP's concepts?

Answer:

The basic Object-oriented programming concepts are:

Inheritance

Inheritance can be defined as the process where one (parent/super) class acquires the properties (methods and fields) of another (child/sub). With the use of inheritance, the information is made manageable in a hierarchical order.

Polymorphism

Polymorphism is the ability of an object to perform different actions (or, exhibit different behaviours) based on the context.

Abstraction

Abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user. In other words, the user will have the information on what the object does instead of how it does it.

In Java, abstraction is achieved using Abstract classes and interfaces.

Encapsulation

Encapsulation in Java is a mechanism for wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes and can be accessed only through the methods of their current class. Therefore, it is also known as **data hiding**. To achieve encapsulation in Java:

- Declare the variables of a class as private.
- Provide public setter and getter methods to modify and view the variables values.

12)What is a Class? What is an Instance?

Answer:

Class: A class can be defined as a template/blueprint that describes the behaviour/state that the object of its type support.

An object is an **instance** of a class.

13)What is meant by Encapsulation?

Answer:

Encapsulation

Encapsulation in Java is a mechanism for wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes and can be accessed only through the methods of their current class. Therefore, it is also known as **data hiding**. To achieve encapsulation in Java:

- Declare the variables of a class as private.
- Provide public setter and getter methods to modify and view the variables values.

14)What is method overloading and method overriding?

Answer:

Method Overloading

- In method overloading a class have two or more methods in with the same name and different parameters.
- In overloading return type could vary in both methods.
- JVM calls the respective method based on the parameters passed to it, at the time of method call.

Example:

```
public class OverloadingExample
{
	public void display()
	{
		System.out.println("Display method");
	}
	public void display(int a)
	{
		System.out.println("Display method "+a);
	}
	public static void main(String args[])
	{
		OverloadingExample obj = new OverloadingExample();
		obj.display();
		obj.display(20);
	}
}
```

Output:

Display method
Display method 20

Method Overriding

- In method overriding Super class and subclass have methods with same name including parameters.
- In overriding return types should also be same.
- JVM calls the respective method based on the object used to call the method.

Example:

```
class SuperClass
{
	public static void sample()
	{
		System.out.println("Method of the super class");
	}
}
public class RuntimePolymorphism extends SuperClass
{
	public static void sample()
	{
		System.out.println("Method of the sub class");
	}
	public static void main(String args[])
	{
		SuperClass obj1 = new RuntimePolymorphism();
		RuntimePolymorphism obj2 = new RuntimePolymorphism();
		obj1.sample();
	obj2.sample();
	}
}
```

Output:

Method of the super class
Method of the sub class

15)What is a Static member class?

Answer:

In Java, static members are those which belongs to the class and you can access these members without instantiating the class.

The static keyword can be used with methods, fields, classes (inner/nested), blocks.

16)What is the difference between this () and super ()?

Answer:

Along with various other keywords Java also provides this and super as special keywords which primarily used to represent current instance of a class and its super class respectively. With this similarity both these keywords have significant differences between them which are listed as below -

Key	this	super
Represent and Reference	this keyword mainly represents the current instance of a class.	On other hand super keyword represents the current instance of a parent class.
Interaction with class constructor	this keyword used to call default constructor of the same class.	super keyword used to call default constructor of the parent class.
Method accessibility	this keyword used to access methods of the current class as it has reference of current class.	One can access the method of parent class with the help of super keyword.
Static context	this keyword can be referred from static context i.e. can be invoked from static instance. For instance, we can write System.out.println(this.x) which will print value of x without any compilation or runtime error.	On other hand super keyword can't be referred from static context i.e. can't be invoked from static instance. For instance, we cannot write System.out.println(super.x) this will lead to compile time error.

17)What is meant by Inheritance?

Answer:

Inheritance

Inheritance can be defined as the process where one (parent/super) class acquires the properties (methods and fields) of another (child/sub). With the use of inheritance, the information is made manageable in a hierarchical order.

18)What is the use of Inheritance and what are its advantages?

Answer:

Use of Inheritance:

- For Method Overriding (so runtime polymorphism can be achieved).
- For Code Reusability.

Advantages of Inheritance:

- The main **advantages of inheritance** are code reusability and readability.
- When child class **inherits** the properties and functionality of parent class, we need not to write the same code again in child class.
- This makes it easier to reuse the code, makes us write the less code and the code becomes much more readable.

19)Differentiate between a Class and an Object?

Answer:

Object - Objects have states and behaviours. Example: A dog has states - colour, name, breed as well as behaviours – wagging the tail, barking, eating. An object is an instance of a class.

Class - A class can be defined as a template/blueprint that describes the behavior/state that the object of its type support.

In the following example, Pummy is a class and myPuppy is an object.

```
public class Puppy
{
        public Puppy(String name)       // This constructor has one parameter, name.
        {
                System.out.println("Passed Name is :" + name );
        }
        public static void main(String []args)
        {
                Puppy myPuppy = new Puppy( "tommy" );          // This statement would create an object myPuppy
        }
}
```

If we compile and run the above program, then it will produce the following result.

Output:

```
Passed Name is :tommy
```

20)What is garbage collection?

Answer:

Java Garbage Collection: In java, garbage means unreferenced objects. Garbage Collection is process of reclaiming the runtime unused memory automatically. In other words, it is a way to destroy the unused objects. To do so, we were using free() function in C language and delete() in C++. But, in java it is performed automatically. So, java provides better memory management.

21)What is a variable? How to declare variable in java?

Answer:

Variable: A variable provides us with named storage that our programs can manipulate. Each variable in Java has a specific type, which determines the size and layout of the variable's memory; the range of values that can be stored within that memory; and the set of operations that can be applied to the variable. You must declare all variables before they can be used. Following is the basic form of a variable declaration:

data type variable [= value][, variable [= value] ...] ;

Here *data type* is one of Java's datatypes and *variable* is the name of the variable. To declare more than one variable of the specified type, you can use a comma-separated list.

Example:

```
int a, b, c=10;        // Declares three integers, a, b, and c.
double pi = 3.14159; // declares and assigns a value of PI.
```

22)What are the difference between static variable and instance variable?

Answer:

Following are the notable differences between Class (static) and instance variables.

Instance variables	Static (class) variables
Instance variables are declared in a class, but outside a method, constructor or any block.	Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block.
Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.	Static variables are created when the program starts and destroyed when the program stops.
Instance variables can be accessed directly by calling the variable name inside the class. However, within static methods (when instance variables are given accessibility), they should be called using the fully qualified name. *ObjectReference.VariableName*.	Static variables can be accessed by calling with the class name *ClassName.VariableName*.
Instance variables hold values that must be referenced by more than one method, constructor or block, or essential parts of an object's state that must be present throughout the class.	There would only be one copy of each class variable per class, regardless of how many objects are created from it.

23)What are the Data Types in Java?

Answer:

Data Types:

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in the memory.

Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals, or characters in these variables.

There are two data types available in Java –

- Primitive Data Types
- Reference/Object Data Types

24)Define Arrays

Answer:

Arrays: Java provides a data structure, the **array**, which stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

25)What is an Object and how do you allocate memory to it?

Answer:

Object - Objects have states and behaviours. Example: A dog has states - colour, name, breed as well as behaviours – wagging the tail, barking, eating. An object is an instance of a class.

In Java, all objects are dynamically allocated on Heap. In Java, when we only declare a variable of a class type, only a reference is created (memory is not allocated for the object). To allocate memory to an object, we must use new(). So, the object is always allocated memory on heap

26)What is the difference between constructor and method?

Answer:

Following are the difference between constructor and method.

- Constructor is used to initialize an object whereas method is used to exhibits functionality of an object.
- Constructors are invoked implicitly whereas methods are invoked explicitly.
- Constructor does not return any value where the method may/may not return a value.
- In case constructor is not present, a default constructor is provided by java compiler. In the case of a method, no default method is provided.
- Constructor should be of the same name as that of class. Method name should not be of the same name as that of class.

27)What is static variable and static method and static Block?

Answer:

Static Variable: The static variable is a class level variable and it is common to all the class objects i.e. a single copy of the static variable is shared among all the class objects.

Static Method: A static method manipulates the static variables in a class. It belongs to the class instead of the class objects and can be invoked without using a class object.

Static Block: The static initialization blocks can only initialize the static instance variables. These blocks are only executed once when the class is loaded.

A program that demonstrates this is given as follows:

Example:

```
public class Demo
{
        static int x = 10;
        static int y;
        static void func(int z)
        {
                System.out.println("x = " + x);
                System.out.println("y = " + y);
                System.out.println("z = " + z);
        }
        static
        {
                System.out.println("Running static initialization block.");
                y = x + 5;
        }
        public static void main(String args[])
        {
                func(8);
        }
}
```

Output:

Running static initialization block.
x = 10
y = 15
z = 8

28)How does the recursion concept work?

Answer:

Recursion:

The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called as recursive function. Using recursive algorithm, certain problems can be solved quite easily. Examples of such problems are Towers of Hanoi (TOH), Inorder/Preorder/Postorder Tree Traversals, DFS of Graph, etc.

In the recursive program, the solution to the base case is provided and the solution of the bigger problem is expressed in terms of smaller problems.

```
int fact(int n)
{
   if (n < = 1) // base case
      return 1;
   else
      return n*fact(n-1);
}
```

In the above example, base case for n < = 1 is defined and larger value of number can be solved by converting to smaller one till base case is reached. The idea is to represent a problem in terms of one or more smaller problems, and add one or more base conditions that stop the recursion. For example, we compute factorial n if we know factorial of (n-1). The base case for factorial would be n = 0. We return 1 when n = 0.