1. What is Inheritance in Java?

Ans:- Inheritance in java is a mechanism in which one object acquire all the properties and behaviors of a parent object. The idea behind inheritance in Java is that you can create new classes that are built upon existing classes.

1. What is superclass and subclass?

Ans:- A class from where all the features are inherited by sub class that class is known as super class.it is also called as parent class or base class.

A class which inherits all the member like field, inner classes, methods etc is known as sub class. It is also called as derived class or child class.

1. How is Inheritance is implemented / achieved in Java?

Ans:- Inheritance is implemented in java using two ways:-

extends keyword use to develop inheritance between two classes and two interfaces

Implements**:** keyword is used to develop inheritance between a class and interface.

1. What is polymorphism?

Ans:- a) in general word:- one thing exists in more than one form then it is called polymorphism.

b) There are two types of polymorphism:- 1) static polymorphism(compile time)

2) Dynamic polymorphism(runt time).

c) polymorphism can be achieved using method overloading ( compile time ) and method overriding ( run time ).

d) Achieving run time polymorphism using method overriding:-

when we make an object of subclass with reference of the child type class only

and also we make one method which take sub class type reference as a parameter and calling overridden method as per object creation.

1. Differentiate between method overloading and method overriding?

Ans:-

| Method overloading enables several method to use the same name but have distinct signature. | Method overriding enables child class method to replace existing functionality with the same name and different parameters. |
| --- | --- |
| Implement compile time polymorphism | Implement run time polymorphism |
| Occur between the method in the same class | Occur between super class and subclass |
| The method call is determined at the compile time | The method call is determined at runt time based on the object type. |
| On error the effect will be visible at run time | On error effect will be visible at compile time. |

1. What is an abstraction explain with an example

Ans:- Abstraction in Java is a process of achieve security and abstraction(hide certain details and show only necessary details to the users). It can be achieve using abstract class.

Example:- ATM machine, All are performing operations like cash withdraw, money transfer, retrieve statement but they don't know about internal details.

1. What is the difference between an abstract method final method in java? Explain with an example?

Ans:- abtstract method is just declared without body or implementation and define in sub class using inheritance but final method cannot modify onced defined and cannot overridden.

1. What is the final class in java?

Ans:- class declared as final cannot participate in inheritance. Restricting inheritance!.

Final class is complete in nature it means it is not abstract class. If we will inherit final class that compiler will give an error.

Example:-

final class ParentClass

{

Void disp()

{

System.out.println(“parent class method declared as final”);

}

}

Class ChildClass extends ParentClass

{

Void disp()

{

System.out.println(“child class method”);

}

}

Public class Demo

{

ParentClass ref = new ChildClass();

ref.disp();

}

1. Differentiate between abstraction and encapsulation?

Ans:-

| Abstraction is a feature of Oops which hide internal details from user and show only functionality that is essential details. | Encapsulation is also a feature of Oops and it also hide a code and data into a single unit or entity that the data can be protected and safe from outside world. |
| --- | --- |
| It focuses on external lookout.  Ex:- ATM machine :- all user use operation of cash withdraw and many more but don't know internal details | It focuses on internal working  Ex:- it hide implementation details and everything from user using private keyword. |
| It can be implemented using abstract classes and interface | It can be implemented using access modifiers such as public, private, protected, default. |
| In abstraction we use abstract classes and interface to hide the code complexities. | In encapsulation we use the getters and setters method to hide the data. |

1. Differentiate between compile time polymorphism and run time polymorphism?

Ans:-

| compile time polymorphism is less flexible and all things execute at compile time | Runtime polymorphism is more flexible and all thing executed at run time. |
| --- | --- |
| The call is resolved by the compiler | Call is not resolved by the compiler. |
| Inheritance is not involved | Inheritance is involved |
| It is also known as static binding, early binding and overloading as well | It is also known as dnamic binding, late binding and overriding as well. |
| It provides fast execution because which method will be invoke on the basis of parameters is decided at compile time. | It provides slow execution because the method that need to executed decided at run time. |
| Method overloading is the compile time polymorphism where more than one method shares the same name with different parameters and return type prototype in same classes | Method overriding is the run time polymorphism where method with same name and sam signature get inherited from parent class in child class and replace functionality as per child class requirement. |