

ASSIGNMENT...

1. What is Inheritance in Java?

Ans: The technique of creating a new class by using an existing class functionality is called inheritance in Java.

In other words, inheritance is a process where a child class acquires all the properties and behaviours of the

parent class.

2. What is superclass and subclass??

Ans: A class from where a subclass inherits features is called superclass. It is also called base class or parent

class.

A class that inherits all the members (fields, method, and nested classes) from another class is called a

subclass. It is also called a derived class, child class, or extended class.

3. How is Inheritance implemented/achieved in Java?

Ans: Inheritance can be implemented or achieved by using two keywords:

extends: extends is a keyword that is used for developing the inheritance between two classes and two

interfaces.

implements: implements keyword is used for developing the inheritance between a class and interface.

4. What is polymorphism?

Ans: Polymorphism in OOP is the ability of an entity to take several forms. In other words, it refers to the ability of

an object (or a reference to an object) to take different forms of objects. It allows a common data-gathering

message to be sent to each class. Polymorphism encourages what is called 'extendibility' which means an

object or a class can have its uses extended.

5. Differentiate between method overloading and method overriding.

Overriding	Overloading
Method overriding always needs inheritance.	Method overloading may or may not require inheritance.

In case of method overriding, <i>parameter must be same.</i>	In case of method overloading, <i>parameter must be different.</i>
Dynamic binding is being used for overriding methods.	Static binding is being used for overloaded methods.
Method overriding is a run-time polymorphism.	Method overloading is a compile-time polymorphism.
It is performed in two classes with inheritance relationships.	It occurs within the class.

6. What is an abstraction explained with an Example?

Ans: Abstraction is nothing but the quality of dealing with ideas rather than events. It basically deals with hiding

the internal details and showing the essential things to the user.

Abstract class Sports {
// abstract class sports

Abstract void jump ();
// abstract method

}

7. What is the difference between an abstract method and final method in Java? Explain with an example

Ans: The abstract method is incomplete while the final method is regarded as complete. The only way to use

an abstract method is by overriding it, but you cannot override a final method in Java.

8. What is the final class in Java?

Ans: A class declared with the final keyword is known as the final class. A final class can't be inherited by

subclasses. By using the final class, we can restrict the inheritance of the class. We can create a class as a final

class only if it is complete in nature, which means it must not be an abstract class. In java, all the wrapper

classes are final classes like String, Integer, etc.

If we try to inherit a final class, then the compiler throws an error at compilation time. We can't create a class as

immutable without the final class.

final class ParentClass

{

void showData()

{

```

System.out.println("This is a method of final Parent class");
}
}
//It will throw compilation error
class ChildClass extends ParentClass
{
void showData()
{
System.out.println("This is a method of Child class");
}
}
class MainClass
{
public static void main (String arg[])
{
ParentClass obj = new ChildClass();
obj.showData();
}
}

```

9. Differentiate between abstraction and encapsulation.

Abstraction	Encapsulation
Abstraction is the process or method of gaining the information.	While encapsulation is the process or method to contain the information.
In abstraction, problems are solved at the design or interface level.	While in encapsulation, problems are solved at the implementation level.
We can implement abstraction using abstract class and interfaces.	Whereas encapsulation can be implemented using by access modifier i.e. private, protected and public.
Abstraction provides access to specific part of data.	Encapsulation hides data and the user can not access same directly (data hiding).
Abstraction focus is on “what” should be done.	Encapsulation focus is on “How” it should be done.

10. Difference between Runtime and compile time polymorphism explain with an example

Runtime time polymorphism	compile time polymorphism
In Run time Polymorphism, the call is not resolved by the compiler.	In Compile time Polymorphism, the call is resolved by the compiler.
It is also known as Dynamic binding, Late binding and overriding as well.	It is also known as Static binding, Early binding and overloading as well
It is achieved by virtual functions and pointers.	It is achieved by function overloading and operator overloading.
Inheritance is involved.	Inheritance is not involved.
Compile time polymorphism is less flexible as all things execute at compile time.	Run time polymorphism is more flexible as all things execute at run time.