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 are 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 Overriding.
Ans:

Method Overloading	**Method Overriding**
Implements Compile time Polymorphism	Implements Runtime Polymorphism
The method call is determined at Compile Time	The method call is determined at Runtime
Occurs between methods inside the same class, i.e, no inheritance is required for this to happen	Occurs between a Superclass and its Subclass, i.e, inheritance is a must for this to happen
The methods will have same name but different parameters	The method signatures will be same(both name and parameters)
On error, the effect will ve visible at Compile Time	On error, the effect will be visible at Runtime

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 a 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.
Ans:

Abstraction	Encapsulation
It is a feature of OOPs that hides unnecessary details and shows only the essential information.	It is also a feature of OOPs that hides/binds the code and data into a single entity/unit so that data can be protected from the outside world.

It solves an issue at the design level.	It solves an issue at the implementation level.
It focuses on external lookout.	It focuses on internal working.
Implemented using abstract classes and interfaces	Implemented using access modifiers(public, private, protected)
It is the process of gaining the information.	It is the process of containing information.
We use abstract classes and interfaces to hide code complexities.	We use getter and setter methods to hide the data.
The objects are encapsulated which helps perform abstraction.	The object need not be abstracted to result in encapsulation.

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

Compile Time Polymorphism	Runtime Polymorphism
Less flexible as all things execute at Compile Time.	More flexible as all things execute at Run Time.
Method Call is resolved at Compile Time.	Method Call is resolved at Run time.
No inheritance is involved.	Inheritance is involved.
AKA Static Binding, Early Binding, Overloading.	AKA Dynamic Binding, Late Binding, Overriding
Fast Execution because the method that needs to be executed is known early at Compile Time.	Slow Execution as the method that needs to be executed is only known at Runtime.
Implemented by Method Overloading	Implemented by Method Overriding