19TH JUNE INHERITANCE, POLYMORPHISM AND ABSTRUCTION

1. **What is Inheritance in Java?**

Ans: Inheritance in Java is a mechanism that allows a class to inherit properties and methods from another class. It enables code reusability and the creation of a hierarchy of classes. The class that is being inherited from is called the superclass or parent class, and the class that inherits from it is called the subclass or child class.

2. **What is superclass and subclass??**

Ans: In Java, a superclass is a class that is being inherited from, also known as the parent class or base class. It provides common attributes and behaviors that can be inherited by its subclasses. A subclass, on the other hand, is a class that inherits properties and methods from a superclass. It can add its own unique attributes and behaviors while also having access to the inherited ones.

3. **How is Inheritance implemented/achieved in Java?**

Ans: Inheritance is implemented in Java using the keyword extends. To achieve inheritance, a subclass is created with the extends keyword followed by the name of the superclass. This allows the subclass to inherit all the non-private members (methods and variables) of the superclass. The subclass can then add additional members or override the inherited ones.

4. **What is polymorphism?**

Ans: Polymorphism is a concept in object-oriented programming that allows objects of different classes to be treated as objects of a common superclass. It enables methods to be defined in a generic way, applicable to multiple types of objects. Polymorphism can be achieved through method overriding and method overloading.

5. **Differentiate between method overloading and overriding.**

Ans: Method overloading refers to having multiple methods in the same class with the same name but different parameters. The methods must differ either in the number of parameters or the types of parameters. The compiler determines which method to invoke based on the arguments provided during the method call.

Method overriding, on the other hand, occurs when a subclass provides its own implementation of a method that is already defined in its superclass. The method in the subclass must have the same name, return type, and parameters as the method in the superclass. The method in the subclass overrides the behavior of the superclass method.

6. **What is an abstraction explained with an Example?**

Ans: Abstraction is a concept in Java that focuses on hiding unnecessary details and exposing only essential features of an object. It allows the creation of abstract classes and interfaces that define common behaviors and methods without specifying their implementation. An example of abstraction is a car. We don't need to know the intricate details of how the car's engine works internally. We can simply use the car's interface (accelerator, brake, steering wheel) to operate it.

7. **What is the difference between an abstract method and final method in Java?**

Ans: An abstract method in Java is a method declared without an implementation in an abstract class or an interface. It provides a signature for the method without specifying the implementation details. Subclasses or implementing classes are required to provide the implementation for the abstract method.

A final method in Java is a method that cannot be overridden by any subclass. When a method is marked as final in a class, it means that the method's implementation in the class is the final and cannot be modified by any subclass.

8. **Explain with an example What is the final class in Java?**

Ans: A final class in Java is a class that cannot be inherited or subclassed by other classes. When a class is marked as final, it means that it cannot have any subclasses. This is useful when a class is designed to have a specific implementation and should not be extended or modified. An example of a final class in Java is the String class. It cannot be subclassed, and its implementation cannot be changed.

9. **Differentiate between abstraction and encapsulation.**

Ans: Abstraction and encapsulation are both important concepts in object-oriented programming, but they serve different purposes.

Abstraction focuses on hiding unnecessary details and exposing only essential features of an object. It allows the creation of abstract classes and interfaces to define common behaviors and methods. Abstraction is achieved by defining abstract methods or interfaces that can be implemented by different classes. It helps in managing complexity and allows for code reusability.

Encapsulation, on the other hand, is about bundling data and methods together into a single unit called a class. It allows the data to be accessed and manipulated only through the methods defined in the class, providing control over data integrity and access. Encapsulation helps in achieving data hiding and improves the maintainability and security of the code.

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

Ans: Runtime polymorphism, also known as dynamic polymorphism, occurs when the appropriate method implementation is determined at runtime based on the actual type of the object. It is achieved through method overriding. An example of runtime polymorphism is when a superclass reference variable is used to refer to a subclass object, and the method of the subclass is called.

Compile-time polymorphism, also known as static polymorphism, occurs when the appropriate method implementation is determined at compile time based on the method signature. It is achieved through method overloading. An example of compile-time polymorphism is having multiple methods with the same name but different parameters, and the compiler determines which method to invoke based on the method call's arguments.