Inheritance and Polymorphism

I learned about refactoring, which involves improving the structure and readability of code without changing its functionality. Refactoring helped me organize code using meaningful method names, constants, and separate classes making the program easier to maintain.I also learned method overloading, where multiple methods share the same name but differ in parameters. This allows the same method to handle different types or numbers of inputs improving code flexibility and readability.I studied inheritance which allows a class to acquire properties and methods from another class using the extends keyword. This helps in code reuse and establishing a parent-child relationship between classes.I learned about the Object class which is the root class in Java. Every class implicitly extends the Object class and inherits methods such as toString(), equals(), and hashCode().I understood constructor inheritance where constructors are not inherited but the parent class constructor is called using the super() keyword during object creation in a child class.Also studied access modifiers such as public, private, protected, and default, which control the visibility and accessibility of classes, methods, and variables.Learned method overriding, where a child class provides its own implementation of a method defined in the parent class. Overriding supports runtime polymorphism and allows behavior customization.Understood upcasting and downcasting, where upcasting converts a child object reference to a parent type and is done automatically, while downcasting converts a parent reference back to a child type and requires explicit casting.I learned about polymorphism which allows the same method call to behave differently based on the object type at runtime. Polymorphism improves flexibility and scalability of programs.also studied abstract classes, which cannot be instantiated and are used to define common behavior that subclasses must implement.Learned about final classes, which cannot be inherited, ensuring that the class behavior remains unchanged and secure.I understood the concept of deep inheritance, where multiple levels of inheritance exist, and learned that multiple inheritance is not supported in Java using classes to avoid ambiguity, but it can be achieved using interfaces.