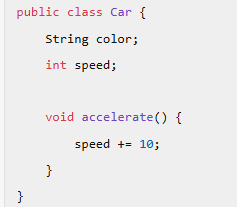
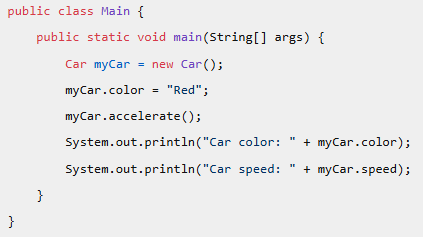
**OOP Concepts**

**Object-Oriented Programming (OOP) is a programming paradigm that uses “objects” to design applications and programs. It simplifies software development and maintenance by providing some key concepts:**

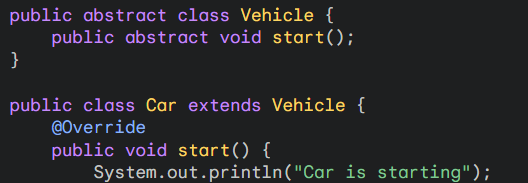
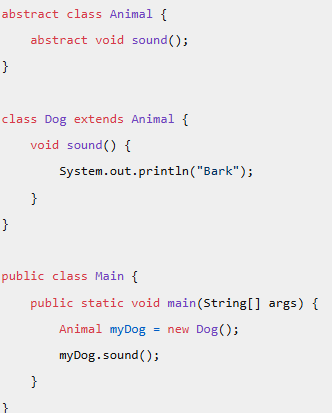
**1. Class**

**A class is a blueprint for creating objects. It defines a datatype by bundling data and methods that work on the data into one single unit.  
**

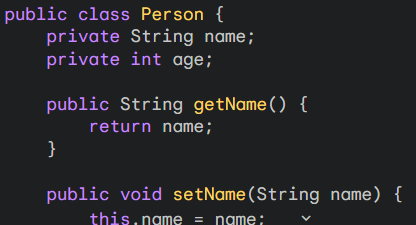
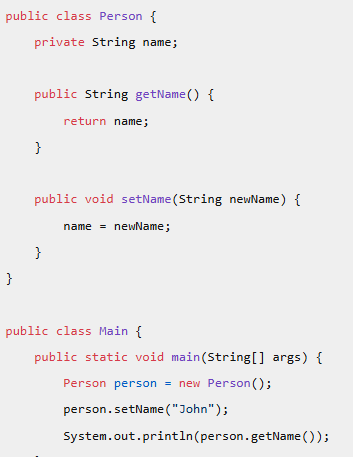
**2. Object**

**An object is an instance of a class. It contains state and behaviour. For example, a Car class can have objects like Audi, BMW, etc.  
**

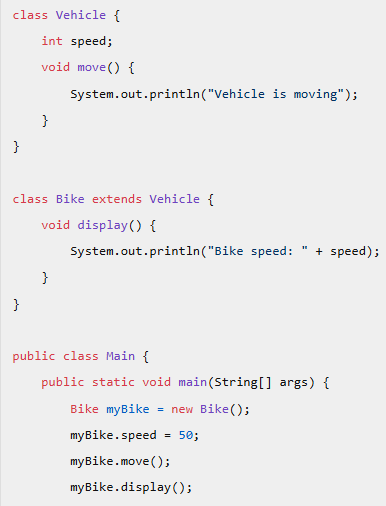
**3. Abstraction**

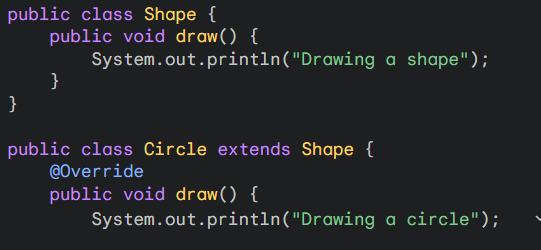
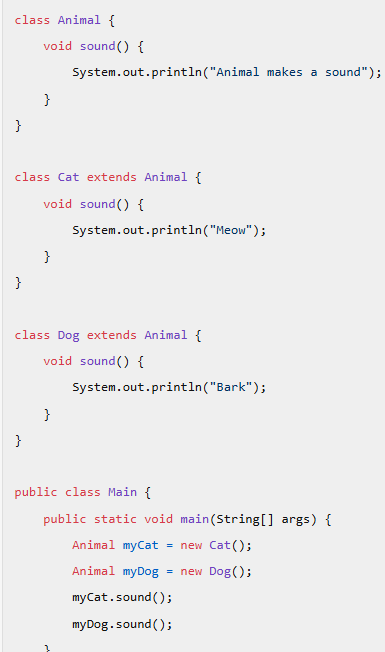
**Abstraction is the concept of hiding the complex implementation details and showing only the necessary features of an object. In Java, abstraction is achieved using abstract classes and   
Interfaces  
  
**

**4. Encapsulation**

**Encapsulation is the technique of wrapping the data (variables) and code (methods) together as a single unit. It restricts direct access to some of an object’s components, which can prevent the accidental modification of data.  
**

**5. Inheritance**

**Inheritance is a mechanism where one class acquires the properties (fields) and behaviors (methods) of another class. It helps in code reusability and method overriding.  
**

**6. Polymorphism  
Polymorphism allows methods to do different things based on the object it is acting upon. It can  
be achieved through method overloading and method overriding.  
  
  
**

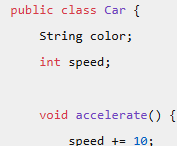
**Interview Questions**

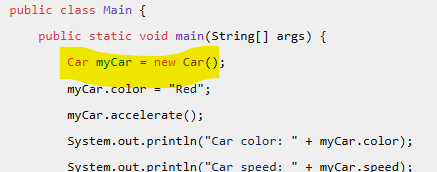
**Class and Object**

**Q: What is a class in Java? How is it different from an object?**

**A:** A class in Java is a blueprint or template for creating objects. It defines a datatype by bundling data and methods that work on the data into one single unit. An object, on the other hand, is an instance of a class. While a class is a logical entity, an object is a physical entity that occupies memory  
  
**Can you explain the concept of an object with an example?**

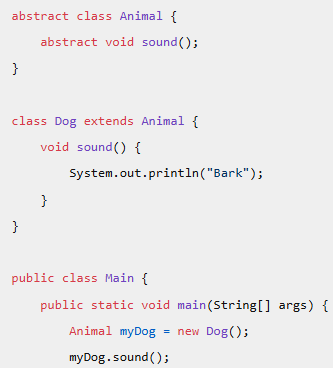
**A:** Sure! Consider a Car class:

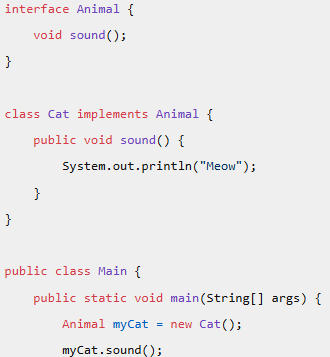
 Here, Car is a class. You can create objects of this class like this:

  
  
**Abstraction**

**Q: What is abstraction in Java? How do you achieve it?**

**A:** Abstraction is the concept of hiding the complex implementation details and showing only the necessary features of an object. In Java, abstraction is achieved using abstract classes and interfaces.

**Q: Can you provide an example of an abstract class and an interface?**

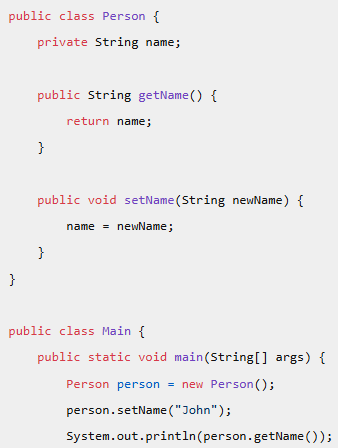
  
  
**Encapsulation**

**Q: What is encapsulation and why is it important?**

**A:** Encapsulation is the technique of wrapping the data (variables) and code (methods) together as a single unit. It restricts direct access to some of an object’s components, which can prevent the accidental modification of data. Encapsulation is important because it helps to protect the internal state of an object and promotes modularity and maintainability.

**Q: How do you implement encapsulation in Java?**

**A:** Encapsulation is implemented using access modifiers (private, protected, public) to restrict access to the fields of a class. Public getter and setter methods are provided to access and update the values of the private fields.

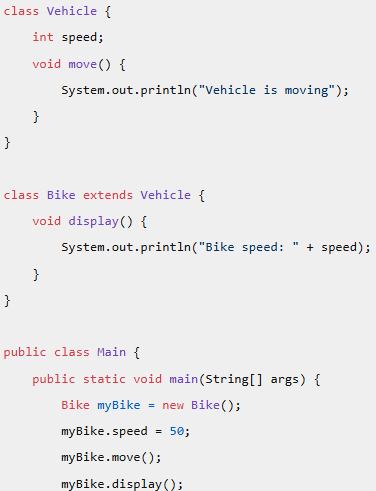


**Inheritance**

**Q: What is inheritance in Java? How does it promote code reuse?**

**A:** Inheritance is a mechanism where one class acquires the properties (fields) and behaviors (methods) of another class. It promotes code reuse by allowing a new class to inherit the fields and methods of an existing class, reducing redundancy.

**Q: Can you explain the difference between single inheritance and multiple inheritance? How does Java handle multiple inheritance?**

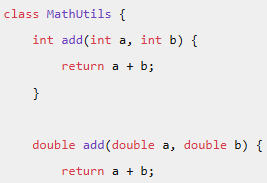
**A:** Single inheritance means a class inherits from one superclass, while multiple inheritance means a class can inherit from more than one superclass. Java does not support multiple inheritance with classes to avoid complexity and ambiguity. However, Java supports multiple inheritance through interfaces.  


**Polymorphism**

**Q: What is polymorphism in Java? Can you explain with an example?**

**A:** Polymorphism allows methods to do different things based on the object it is acting upon. It can be achieved through method overloading and method overriding.

**Q: What is the difference between method overloading and method overriding?**

**A:** Method overloading occurs when multiple methods in the same class have the same name but different parameters. Method overriding occurs when a subclass provides a specific implementation for a method that is already defined in its superclass  
**Example of Method Overloading:** **Example of Method Overriding:**