**Final Keyword in Java**

1. **Declare a final variable in Java and demonstrate how trying to change its value results in a compile-time error.**
   * *Write a class with a final variable and attempt to reassign it.*
2. **Create a final method in a superclass and try to override it in a subclass. What happens?**
   * *Write a superclass with a final method and a subclass that attempts to override it.*
3. **Write a Java class with a final class and explain what happens if you try to extend this class in another class.**
   * *Define a final class and attempt to create a subclass.*
4. **Initialize a final variable in a constructor. Show how it can be done.**
   * *Create a class with a final variable and initialize it in the constructor.*
5. **Write a Java program where you use final keyword in the context of method parameters and explain its usage.**
   * *Create a method with a final parameter and demonstrate that it cannot be modified within the method.*

**Polymorphism in Java**

1. **Implement method overloading by creating a class with multiple methods that have the same name but different parameters.**
   * *Write a class that has overloaded methods for performing addition with different types of arguments.*
2. **Create a superclass with a method, and then override this method in a subclass. Show how the method call resolves at runtime.**
   * *Write a superclass with a method and a subclass that overrides this method.*
3. **Write a Java program to demonstrate runtime polymorphism using a base class reference to a derived class object.**
   * *Create a base class and a derived class, and show how runtime polymorphism is achieved.*
4. **Define a class with method overloading and demonstrate how Java resolves which method to call based on the parameters passed.**
   * *Create a class with multiple overloaded methods and show the method resolution based on different arguments.*
5. **Write a Java program to illustrate that Java does not support operator overloading by attempting to overload an operator.**
   * *Create a class and attempt to overload operators, explaining why this is not possible in Java.*

**Final Keyword in Java**

1. **Declare a final variable and demonstrate a compile-time error when trying to change its value:**

java

Copy code

public class FinalVariableExample {

public static void main(String[] args) {

final int x = 10;

// x = 20; // This line would cause a compile-time error

System.out.println("Value of x: " + x);

}

}

1. **Create a final method in a superclass and attempt to override it in a subclass:**

java

Copy code

class Superclass {

final void finalMethod() {

System.out.println("This is a final method.");

}

}

class Subclass extends Superclass {

// Uncommenting the following method would cause a compile-time error

// void finalMethod() {

// System.out.println("Attempting to override final method.");

// }

}

public class FinalMethodExample {

public static void main(String[] args) {

Subclass obj = new Subclass();

obj.finalMethod();

}

}

1. **Write a Java class with a final class and attempt to extend it:**

java

Copy code

final class FinalClass {

void display() {

System.out.println("This is a final class.");

}

}

// Uncommenting the following class would cause a compile-time error

// class Subclass extends FinalClass {

// }

public class FinalClassExample {

public static void main(String[] args) {

FinalClass obj = new FinalClass();

obj.display();

}

}

1. **Initialize a final variable in a constructor:**

java

Copy code

public class FinalVariableInitialization {

final int value;

// Constructor initializing the final variable

FinalVariableInitialization(int val) {

value = val;

}

public static void main(String[] args) {

FinalVariableInitialization obj = new FinalVariableInitialization(100);

System.out.println("Final variable value: " + obj.value);

}

}

1. **Use the final keyword in the context of method parameters:**

java

Copy code

public class FinalParameterExample {

void modifyParameter(final int num) {

// num = 10; // This line would cause a compile-time error

System.out.println("Parameter value: " + num);

}

public static void main(String[] args) {

FinalParameterExample obj = new FinalParameterExample();

obj.modifyParameter(5);

}

}

**Polymorphism in Java**

1. **Implement method overloading:**

java

Copy code

public class MethodOverloadingExample {

// Overloaded methods for addition

int add(int a, int b) {

return a + b;

}

double add(double a, double b) {

return a + b;

}

public static void main(String[] args) {

MethodOverloadingExample obj = new MethodOverloadingExample();

System.out.println("Addition of integers: " + obj.add(5, 10));

System.out.println("Addition of doubles: " + obj.add(5.5, 10.5));

}

}

1. **Create a superclass with a method and override it in a subclass:**

java

Copy code

class Animal {

void makeSound() {

System.out.println("Animal makes a sound");

}

}

class Dog extends Animal {

@Override

void makeSound() {

System.out.println("Dog barks");

}

}

public class MethodOverridingExample {

public static void main(String[] args) {

Animal myAnimal = new Dog();

myAnimal.makeSound(); // This will call Dog's makeSound method

}

}

1. **Demonstrate runtime polymorphism with a base class reference to a derived class object:**

java

Copy code

class Shape {

void draw() {

System.out.println("Drawing a shape");

}

}

class Circle extends Shape {

@Override

void draw() {

System.out.println("Drawing a circle");

}

}

public class RuntimePolymorphismExample {

public static void main(String[] args) {

Shape shape = new Circle(); // Base class reference to derived class object

shape.draw(); // This will call Circle's draw method

}

}

1. **Define a class with method overloading and demonstrate method resolution based on parameters:**

java

Copy code

public class MethodResolutionExample {

void display(int a) {

System.out.println("Integer: " + a);

}

void display(double a) {

System.out.println("Double: " + a);

}

void display(String a) {

System.out.println("String: " + a);

}

public static void main(String[] args) {

MethodResolutionExample obj = new MethodResolutionExample();

obj.display(10); // Calls display(int)

obj.display(10.5); // Calls display(double)

obj.display("Hello"); // Calls display(String)

}

}

1. **Illustrate that Java does not support operator overloading:**

java

Copy code

// Java does not support operator overloading. The following code is just for illustration

public class OperatorOverloadingExample {

// Attempting to overload operators is not possible in Java

// The following code will not compile if uncommented:

// OperatorOverloadingExample operatorOverload(OperatorOverloadingExample obj) {

// return obj;

// }

public static void main(String[] args) {

// Example to show that operator overloading is not supported

System.out.println("Java does not support operator overloading.");

}

}