public class OuterClass {

private static int staticValue = 100;

private int instanceValue = 200;

public static class StaticNestedClass {

static {

System.out.println("Static block in StaticNestedClass executed");

}

{

System.out.println("Instance block in StaticNestedClass executed");

}

public StaticNestedClass() {

System.out.println("Constructor of StaticNestedClass called");

}

public void displayValues() {

System.out.println("Accessing static variable from OuterClass: " + OuterClass.staticValue);

}

public void showOuterClassInstanceValue(OuterClass outer) {

System.out.println("Accessing instance variable from OuterClass using OuterClass.this: " + outer.instanceValue);

}

}

public final class FinalInnerClass {

private final int finalValue = 300;

public FinalInnerClass() {

System.out.println("FinalInnerClass created with finalValue = " + finalValue);

}

public final void showFinalValue() {

System.out.println("Final value is immutable: " + finalValue);

}

}

@Override

public String toString() {

return "OuterClass instance: staticValue = " + staticValue + ", instanceValue = " + instanceValue;

}

public void synchronizedMethod() {

synchronized (OuterClass.class) {

staticValue++;

System.out.println("Synchronized block executed, staticValue = " + staticValue);

}

}

public static void staticMethod() {

System.out.println("Static method called. Static variable value: " + staticValue);

}

public static void main(String[] args) {

OuterClass.StaticNestedClass nested = new OuterClass.StaticNestedClass();

nested.displayValues();

OuterClass outer = new OuterClass();

nested.showOuterClassInstanceValue(outer);

OuterClass.FinalInnerClass finalInner = outer.new FinalInnerClass();

finalInner.showFinalValue();

System.out.println(outer);

outer.synchronizedMethod();

OuterClass.staticMethod();

}

}

final class UtilityClass {

public static void utilityMethod1() {

System.out.println("Utility Method 1 called");

}

public static void utilityMethod2() {

System.out.println("Utility Method 2 called");

}

private UtilityClass() {

}

}

/\*output

Static block in StaticNestedClass executed

Instance block in StaticNestedClass executed

Constructor of StaticNestedClass called

Accessing static variable from OuterClass: 100

Accessing instance variable from OuterClass using OuterClass.this: 200

FinalInnerClass created with finalValue = 300

Final value is immutable: 300

OuterClass instance: staticValue = 100, instanceValue = 200

Synchronized block executed, staticValue = 101

Static method called. Static variable value: 101\*/

**Explanation of the Code:**

1. **Static Nested Class**:
   * **StaticNestedClass** contains a static block, an instance block, and a constructor. It demonstrates accessing the static variable **staticVariable** from the outer class but cannot directly access the instance variable **instanceVariable**.
2. **Final Inner Class**:
   * **FinalInnerClass** is defined as a final class, preventing subclassing. It has a final variable **immutableVariable**, which demonstrates immutability since it cannot be changed after initialization.
3. **this and super Keywords**:
   * The **toString** method in **OuterClass** is overridden to provide a custom representation of the object.
   * In the **StaticNestedClass**, **OuterClass.this** is used to refer to the outer class's instance when calling the **toString** method.
4. **Static Method**:
   * The static method **staticMethod** demonstrates access to the static variable while restricting access to the instance variable.
5. **Synchronized Method**:
   * The **synchronizedMethod** demonstrates thread-safe access to the static variable **staticVariable**.
6. **Utility Class**:
   * **UtilityClass** is marked as final, and all its methods are static, preventing subclassing and instantiation.

