1.

class CarDetails {

String brand;

String model;

double price;

CarDetails(String brand, String model, double price) {

this.brand = brand;

this.model = model;

this.price = price;

}

@Override

public String toString() {

return "Car: " + brand + " " + model + ", Price: $" + price;

}

}

public class Car {

public static void main(String[] args) {

CarDetails car1 = new CarDetails("Toyota", "Camry", 25000);

CarDetails car2 = new CarDetails("Honda", "Civic", 22000);

System.out.println(car1);

System.out.println("Class name: " + car1.getClass().getName());

System.out.println();

System.out.println(car2);

System.out.println("Class name: " + car2.getClass().getName());

}

}

2.

class Address {

String city;

String state;

Address(String city, String state) {

this.city = city;

this.state = state;

}

@Override

public String toString() {

return city + ", " + state;

}

}

class Person implements Cloneable {

String name;

Address address;

Person(String name, Address address) {

this.name = name;

this.address = address;

}

@Override

public Person clone() throws CloneNotSupportedException {

return (Person) super.clone();

}

public Person deepClone() {

Address newAddress = new Address(this.address.city, this.address.state);

return new Person(this.name, newAddress);

}

@Override

public String toString() {

return "Person[name=" + name + ", address=" + address + "]";

}

}

public class Cloning {

public static void main(String[] args) throws CloneNotSupportedException {

Address addr = new Address("Mumbai", "Maharashtra");

Person p1 = new Person("Raj", addr);

Person p2 = p1.clone();

Person p3 = p1.deepClone();

System.out.println("Original: " + p1);

System.out.println("Shallow Copy: " + p2);

System.out.println("Deep Copy: " + p3);

System.out.println();

p1.address.city = "Delhi";

p1.address.state = "Delhi";

System.out.println("After changing original address:");

System.out.println("Original: " + p1);

System.out.println("Shallow Copy: " + p2);

System.out.println("Deep Copy: " + p3);

}

}

3.

interface MathOperation {

int operate(int a, int b);

}

class Calculator {

static class Operation {

int add(int a, int b) {

return a + b;

}

}

void performSubtraction(int a, int b) {

class Subtraction {

int subtract() {

return a - b;

}

}

Subtraction sub = new Subtraction();

System.out.println("Subtraction: " + sub.subtract());

}

MathOperation getMultiplication() {

return new MathOperation() {

@Override

public int operate(int a, int b) {

return a \* b;

}

};

}

}

public class InnerClassVariants {

public static void main(String[] args) {

Calculator.Operation op = new Calculator.Operation();

System.out.println("Addition: " + op.add(10, 5));

Calculator calc = new Calculator();

calc.performSubtraction(10, 5);

MathOperation multiply = calc.getMultiplication();

System.out.println("Multiplication: " + multiply.operate(10, 5));

}

}

4.

class BookDetails {

String title;

String author;

BookDetails(String title, String author) {

this.title = title;

this.author = author;

}

@Override

public boolean equals(Object obj) {

if (obj == null) return false;

if (obj == this) return true;

if (!(obj instanceof BookDetails)) return false;

BookDetails b = (BookDetails) obj;

return (title == null ? b.title == null : title.equals(b.title))

&& (author == null ? b.author == null : author.equals(b.author));

}

@Override

public int hashCode() {

int result = 17;

result = 31 \* result + (title == null ? 0 : title.hashCode());

result = 31 \* result + (author == null ? 0 : author.hashCode());

return result;

}

@Override

public String toString() {

return "BookDetails{title='" + title + "', author='" + author + "'}";

}

}

public class Main {

public static void main(String[] args) {

BookDetails b1 = new BookDetails("Java", "John");

BookDetails b2 = new BookDetails("Java", "John");

BookDetails b3 = b1;

BookDetails b4 = new BookDetails ("Python", "Jane");

System.out.println("b1 == b2: " + (b1 == b2));

System.out.println("b1.equals(b2): " + b1.equals(b2));

System.out.println();

System.out.println("b1 == b3: " + (b1 == b3));

System.out.println("b1.equals(b3): " + b1.equals(b3));

System.out.println();

System.out.println("b1 == b4: " + (b1 == b4));

System.out.println("b1.equals(b4): " + b1.equals(b4));

System.out.println();

System.out.println("b1 == null: " + (b1 == null));

System.out.println("b1.equals(null): " + b1.equals(null));

}

}

5.

class Outer {

private String message;

Outer(String message) {

this.message = message;

}

class Inner {

void display() {

System.out.println("Message from inner class: " + message);

}

}

}

public class MemberInnerClass {

public static void main(String[] args) {

Outer outer = new Outer("Hello from Outer class");

Outer.Inner inner = outer.new Inner();

inner.display();

}

}

6.

import java.util.HashSet;

import java.util.Objects;

class Student {

int id;

String name;

Student(int id, String name) {

this.id = id;

this.name = name;

}

@Override

public boolean equals(Object obj) {

if (obj == null) return false;

if (obj == this) return true;

Student s = (Student) obj;

return id == s.id;

}

@Override

public int hashCode() {

return Objects.hash(id);

}

@Override

public String toString() {

return "Student[id=" + id + ", name=" + name + "]";

}

}

public class StudentDetails {

public static void main(String[] args) {

HashSet<Student> students = new HashSet<>();

Student s1 = new Student(101, "Alice");

Student s2 = new Student(102, "Bob");

Student s3 = new Student(101, "Alice Again");

students.add(s1);

students.add(s2);

students.add(s3);

System.out.println("HashSet size: " + students.size());

System.out.println("Students in HashSet:");

for (Student s : students) {

System.out.println(s);

}

System.out.println();

System.out.println("s1.equals(s3): " + s1.equals(s3));

System.out.println("s1.hashCode(): " + s1.hashCode());

System.out.println("s3.hashCode(): " + s3.hashCode());

}

}