Q1.

public class VariableScopeExample {

private static int x = 1; // Class variable accessible in the whole class

public static void main(String[] args) {

int y = 5; // Local variable to main method

System.out.println("Class variable x: " + x);

System.out.println("Local variable y: " + y);

someMethod();

}

public static void someMethod() {

System.out.println("Class variable x from someMethod: " + x);

// Error: y cannot be accessed here

// System.out.println("Local variable y from someMethod: " + y);

}

}

out put

java -cp /tmp/K35ys1Y7fF/VariableScopeExample

Class variable x: 1

Local variable y: 5

Class variable x from someMethod: 1

=== Code Execution Successful ===

Q2.

public class Car {

// Fields (attributes)

private String color;

private String model;

// Constructor

public Car(String color, String model) {

this.color = color;

this.model = model;

}

// Method to display information about the car

public void displayInfo() {

System.out.println("Car model: " + model + ", Color: " + color);

}

// Main method to create and use objects of Car class

public static void main(String[] args) {

Car myCar = new Car("Red", "Toyota Corolla");

myCar.displayInfo();

}

}

out put

java -cp /tmp/ss8Fpt6SBG/Car

Car model: Toyota Corolla, Color: Red

=== Code Execution Successful ===

Q3.

public class TypeCasting{

public static void main(String[] arg)

{

int myInt=7;

double myDouble=myInt;

System.out.println("Int value="+myInt);

System.out.println("Converted to double="+myDouble);

double anotherDouble= (8.29);

int anotherInt =(int) anotherDouble;

System.out.println("Double value="+anotherDouble);

System.out.println("converted to int="+anotherInt);

}

}

output

java -cp /tmp/6gtkaKfeaB/TypeCasting

Int value=7

Converted to double=7.0

Double value=8.29

converted to int=8

=== Code Execution Successful ===

Q4.

public class ExceptionHandlingExample {

public static void main(String[] args) {

try {

int[] numbers = {1, 2, 3};

System.out.println(numbers[5]); // This will throw an ArrayIndexOutOfBoundsException

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("An exception occurred: " + e.getMessage());

} finally {

System.out.println("The 'try catch' is finished");

}

}

}

output

java -cp /tmp/iYwHZ9k5GH/ExceptionHandlingExample

An exception occurred: Index 5 out of bounds for length 3

The 'try catch' is finished

=== Code Execution Successful ===

Q5.

class Vehicle { // Base class (parent)

protected String brand = "Ford"; // Vehicle attribute

public void honk() { // Vehicle method

System.out.println("Tut tut!");

}

}

class Car extends Vehicle { // Derived class (child)

private String modelName = "Mustang"; // Car attribute

public void displayModel() {

System.out.println("Brand: " + brand + ", Model: " + modelName);

}

}

public class Main {

public static void main(String[] args) {

Car myCar = new Car();

myCar.honk();

myCar.displayModel();

}

}

output

java -cp /tmp/VfVzOeVvV3/Main

Tut tut!

Brand: Ford, Model: Mustang

=== Code Execution Successful ===

Q6.

class Animal {

public void sound() {

System.out.println("Some sound");

}

}

class Dog extends Animal {

@Override

public void sound() {

System.out.println("Woof");

}

}

class Cat extends Animal {

@Override

public void sound() {

System.out.println("Meow");

}

}

public class Main {

public static void main(String[] args) {

Animal myAnimal = new Animal();

Animal myDog = new Dog(); // Corrected syntax: '=' instead of '-'

Animal myCat = new Cat();

myAnimal.sound(); // Outputs Some sound

myDog.sound(); // Outputs Woof

myCat.sound(); // Outputs Meow

}

}

output

java -cp /tmp/WNsIusEcmO/Main

Some sound

Woof

Meow

=== Code Execution Successful ===

Q7.

class AccessSpecifierDemo {

public int publicVar = 100; // Accessible from any other class

private int privateVar = 200; // Accessible only within the class

protected int protectedVar = 300; // Accessible within the class and by derived classes

public void display() {

System.out.println("Public: " + publicVar);

System.out.println("Private: " + privateVar);

System.out.println("Protected: " + protectedVar);

}

}

public class Main {

public static void main(String[] args) {

AccessSpecifierDemo demo = new AccessSpecifierDemo();

demo.display();

System.out.println("PublicVar: " + demo.publicVar);

}

}

output

java -cp /tmp/UPkHV9ZBT1/Main

Public: 100

Private: 200

Protected: 300

PublicVar: 100

=== Code Execution Successful ===

Q8.

class Car {

private String model;

private int year;

// Constructor

public Car(String model, int year) {

this.model = model;

this.year = year;

}

public void displayInfo() {

System.out.println("Model: " + model + " Year: " + year);

}

}

public class Main {

public static void main(String[] args) {

Car car1 = new Car("Toyota Corolla", 2020);

car1.displayInfo();

}

}

output

java -cp /tmp/JMv5o2QEtB/Main

Model: Toyota Corolla Year: 2020

=== Code Execution Successful ===

Q9.

interface Animal {

void sound(); // Interface method (does not have a body)

}

class Dog implements Animal {

public void sound() {

// The body of sound() is provided here

System.out.println("Woof");

}

}

class Cat implements Animal {

public void sound() {

System.out.println("Meow");

}

}

public class Main {

public static void main(String[] args) {

Dog myDog = new Dog();

Cat myCat = new Cat(); // Corrected syntax: added parentheses

myDog.sound();

myCat.sound();

}

}

output

java -cp /tmp/xtWoqGtB1X/Main

Woof

Meow

=== Code Execution Successful ===

Q10.

class ThreadDemo extends Thread {

public void run() {

try {

// Moving thread to Timed Waiting state

Thread.sleep(150);

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("State after completion: " + Thread.currentThread().getState());

}

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

ThreadDemo t1 = new ThreadDemo();

System.out.println("State when created: " + t1.getState());

t1.start();

System.out.println("State when started: " + t1.getState());

// waiting for thread to die

t1.join();

System.out.println("State after thread ended its task: " + t1.getState());

}

}

output

java -cp /tmp/iq1HPsAu8i/ThreadDemo

State when created: NEW

State when started: RUNNABLE

State after completion: RUNNABLE

State after thread ended its task: TERMINATED

=== Code Execution Successful ===