1. Trace the output for the following program.

using System;

namespace ArrayManipulation

{

class Program

{

static void Main(string[] args)

{

// Step 1: Initializing a 2D array

int[,] array2D = new int[,] { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };

Console.WriteLine("2D array content:");

for (int i = 0; i < array2D.GetLength(0); i++)

{

for (int j = 0; j < array2D.GetLength(1); j++)

{

Console.Write(array2D[i, j] + " ");

}

Console.WriteLine();

}

Console.WriteLine($"Total rows: {array2D.GetLength(0)}");

Console.WriteLine($"Sum of elements in row 1: {SumArray(array2D, 1)}");

}

static int SumArray(int[,] array, int row)

{

int sum = 0;

for (int i = 0; i < array.GetLength(1); i++)

{

sum += array[row, i];

}

return sum;

}

}

}

|  |
| --- |
| 2D array content:  1 2 3  4 5 6  7 8 9  Total rows: 3  Sum of elements in row 1: 15 |

2. Trace the output for the following program.

using System;

namespace PolymorphismExample

{

public class Vehicle

{

protected int speed;

protected int fuel;

public Vehicle(int speed, int fuel)

{

this.speed = speed;

this.fuel = fuel;

}

public virtual void DisplayInfo()

{

Console.WriteLine($"Speed: {speed} Fuel: {fuel}");

}

public virtual void AdjustSpeed(ref int refParam, out string outParam, params int[] newSpeed)

{

if (newSpeed.Length > 0)

{

speed = newSpeed[0];

}

refParam = speed + fuel;

outParam = $"Speed: {speed} Fuel: {fuel}";

}

}

public class Car : Vehicle

{

public Car(int speed, int fuel) : base(speed, fuel) { }

public override void DisplayInfo()

{

Console.WriteLine($"Car - Speed: {speed} Fuel: {fuel}");

}

public override void AdjustSpeed(ref int refParam, out string outParam, params int[] newSpeed)

{

base.AdjustSpeed(ref refParam, out outParam, newSpeed);

refParam += 5; // Specific adjustment for Car

}

}

class Program

{

static void Main(string[] args)

{

int refParam = 0;

string outParam;

Vehicle vehicle = new Vehicle(60, 40);

Car car = new Car(80, 50);

vehicle.DisplayInfo();

car.DisplayInfo();

vehicle.AdjustSpeed(ref refParam, out outParam, 70);

Console.WriteLine($"Vehicle ref: {refParam} out: {outParam}");

vehicle.DisplayInfo();

car.AdjustSpeed(ref refParam, out outParam, 90);

Console.WriteLine($"Car ref: {refParam} out: {outParam}");

car.DisplayInfo();

}

}

}

|  |
| --- |
| Speed: 60 Fuel: 40  Car - Speed: 80 Fuel: 50  Vehicle ref: 110 out: Speed: 70 Fuel: 40  Speed: 70 Fuel: 40  Car ref: 145 out: Speed: 90 Fuel: 50  Car - Speed: 90 Fuel: 50 |

**Problem: Vehicle Management System**

You are required to create a simple vehicle management system that involves different types of vehicles. This system should demonstrate the concepts of inheritance and type casting in Java. Additionally, include static methods to manage and display vehicle information.

**Requirements:**

1. **Create a base class Vehicle**:
   * **Attributes**:
     + vehicleId (String)
     + brand (String)
     + price (double)
     + vehicleCount (int, static)
   * **Methods**:
     + Constructor to initialize all attributes
     + Getters and setters for all attributes
     + displayInfo() method to print vehicle details
     + Static method showVehicleDetails(Vehicle vehicle) that uses type casting to display details based on the specific type of vehicle
2. **Create a derived class Car that inherits from Vehicle**:
   * **Additional Attributes**:
     + numberOfDoors (int)
     + fuelType (String)
   * **Methods**:
     + Constructor to initialize all attributes (including those from the base class)
     + Getters and setters for the new attributes
     + displayCarInfo() method to print car details
3. **Create another derived class Motorcycle that inherits from Vehicle**:
   * **Additional Attributes**:
     + engineCapacity (int) // in cc
     + type (String) // e.g., "Sport", "Cruiser"
   * **Methods**:
     + Constructor to initialize all attributes (including those from the base class)
     + Getters and setters for the new attributes
     + displayMotorcycleInfo() method to print motorcycle details
4. **Test the inheritance structure and type casting**:
   * Create objects of Vehicle, Car, and Motorcycle
   * Use the static method showVehicleDetails(Vehicle vehicle) to display the appropriate information for each object using type casting
   * Also show the total number of vehicles