01)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace lab_3
{
    using System;

    class Course
    {
        private string courseName;
        private string instructorName;
        private double grade;

        public Course(string courseName, string instructorName, double grade)
        {
            CourseName = courseName;
            SetInstructorName(instructorName);
            Grade = grade;
        }

        public string CourseName
        {
            get { return courseName; }
            private set { courseName = value; }
        }
```

```
public double Grade
{
   get { return grade; }
   private set
   {
      if (value < 0 || value > 100)
      {
         throw new ArgumentException("Grade must be between 0 and 100.");
      }
      grade = value;
   }
}

public void SetInstructorName(string name)
{
   if (string.IsNullOrEmpty(name))
   {
      throw new ArgumentException("Instructor name cannot be empty.");
   }
   instructorName = name;
}

private string CalculateLetterGrade()
{
   if (grade >= 90)
   {
      return "A";
   }
```

```
        else if (grade >= 80)
        {
            return "B";
        }
        else if (grade >= 70)
        {
            return "C";
        }
        else if (grade >= 60)
        {
            return "D";
        }
        else
        {
            return "F";
        }
    }

    public void PrintCourseInfo()
    {
        string letterGrade = CalculateLetterGrade();
        Console.WriteLine($"Course: {CourseName}");
        Console.WriteLine($"Instructor: {instructorName}");
        Console.WriteLine($"Letter Grade: {letterGrade}");
    }
}

class Program
{
```

```
    static void Main(string[] args)
    {
        try
        {
            Course course = new Course("Math", "Mr.Anton", 85);
            course.PrintCourseInfo();
        }
        catch (ArgumentException ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }
    }
  }

}
```

02)

```
using System;

public class Employee
{
  // Properties
  public int EmployeeID { get; }
  public string FullName { get; set; }
  public double Salary { get; private set; }

  // Constructor
  public Employee(int employeeID, string fullName, double salary)
  {
    EmployeeID = employeeID;
```

```
        FullName = fullName;
        Salary = salary;
    }

    // Method to display employee information
    public void DisplayEmployeeInfo()
    {
        Console.WriteLine($"Employee ID: {EmployeeID}");
        Console.WriteLine($"Full Name: {FullName}");
        Console.WriteLine($"Salary: ${Salary}");
    }
}

class Program
{
    static void Main(string[] args)
    {
        // Create an instance of Employee
        Employee emp = new Employee(101, "John Doe", 50000);

        // Display employee ID using the read-only property
        Console.WriteLine($"Employee ID: {emp.EmployeeID}");

        // Update full name using the read-write property
        emp.FullName = "Jane Smith";
        Console.WriteLine($"Updated Full Name: {emp.FullName}");

        // Attempt to modify the salary directly from external code
        // This will result in a compilation error because Salary has a private set
```

```
        // emp.Salary = 60000; // Uncommenting this line will result in a compilation error

        // Display employee information
        emp.DisplayEmployeeInfo();
    }
}
```

03)

```
using System;

public class Product
{
    // Fields
    private int productId;
    private string productName;
    private double price;
    private int quantityInStock;

    // Constructor
    public Product(int productId, string productName, double price, int quantityInStock)
    {
        this.productId = productId;
        this.productName = productName;
        this.price = price;
        this.quantityInStock = quantityInStock;
    }

    // Method to add product to the inventory
    public void AddProduct(int quantity)
    {
```

```
        quantityInStock += quantity;
        Console.WriteLine($"{quantity} {productName}(s) added to inventory. New quantity in stock: {quantityInStock}");
    }

    // Method to buy product from the inventory
    public void BuyProduct(int quantity)
    {
        if (quantity > quantityInStock)
        {
            Console.WriteLine("Insufficient quantity in stock.");
            return;
        }

        quantityInStock -= quantity;
        Console.WriteLine($"{quantity} {productName}(s) bought. Remaining quantity in stock: {quantityInStock}");
    }

    // Method to display product details
    public void DisplayProductDetails()
    {
        Console.WriteLine($"Product ID: {productId}");
        Console.WriteLine($"Product Name: {productName}");
        Console.WriteLine($"Price: ${price}");
        Console.WriteLine($"Quantity in Stock: {quantityInStock}");
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        // Create an instance of Product
        Product laptop = new Product(101, "Laptop", 800, 10);

        // Access and display product details
        laptop.DisplayProductDetails();

        // Attempt to modify the product's ID from external code
        // This will result in a compilation error because productId is private
        // laptop.productId = 102; // Uncommenting this line will result in a compilation error
    }
}
```

04)

```
using System;

public class Shape
{
    // Fields
    protected string shapeType;
    protected double area;

    // Constructor
    public Shape(string shapeType)
    {
        this.shapeType = shapeType;
    }
```

```
    // Method to calculate area (to be overridden by subclasses)
    public virtual void CalculateArea(double val1, double val2 = 0)
    {
        // To be implemented by subclasses
    }

    // Method to display shape information
    public virtual void DisplayShapeInfo()
    {
        Console.WriteLine($"Shape Type: {shapeType}");
        Console.WriteLine($"Area: {area}");
    }
}

public class Rectangle : Shape
{
    // Fields
    private double length;
    private double width;

    // Constructor
    public Rectangle(double length, double width) : base("Rectangle")
    {
        this.length = length;
        this.width = width;
        CalculateArea(length, width);
    }
```

```
    // Override CalculateArea method for rectangle
    public override void CalculateArea(double val1, double val2 = 0)
    {
        area = val1 * val2;
    }
}

public class Circle : Shape
{
    // Fields
    private double radius;

    // Constructor
    public Circle(double radius) : base("Circle")
    {
        this.radius = radius;
        CalculateArea(radius);
    }

    // Override CalculateArea method for circle
    public override void CalculateArea(double val1, double val2 = 0)
    {
        area = Math.PI * val1 * val1;
    }
}

class Program
{
    static void Main(string[] args)
```

```
  {
    // Create an instance of Rectangle
    Rectangle rectangle = new Rectangle(5, 3);
    // Display shape information for the rectangle
    rectangle.DisplayShapeInfo();

    Console.WriteLine();

    // Create an instance of Circle
    Circle circle = new Circle(4);
    // Display shape information for the circle
    circle.DisplayShapeInfo();
  }
}
```

05)