1. Considera estás desarrollando un programa donde necesitas trabajar con objetos de tipo Persona. Define una clase Persona, pero en este caso considerando los siguientes atributos de clase: nombre (String), apellidos (String), edad (int), casado (boolean), numeroDocumentoIdentidad(String) y 3 metodos como acciones diferentes por persona de acuerdo a una profesión. Define un constructor y los métodos para poder establecer y obtener los valores de los atributos. Mínimo 7 personas diferentes con acciones diferentes.

```
2. public class Person
3. {
4.     private string name;
5.     private string lastName;
6.     private int age;
7.     private bool married;
8.     private string identityDocument;
9.     private string profession;
10.    public Person(string name, string lastName, int Age, bool married,
   string indetityDocument, string profession)
11.    {
12.        this.name = name;
13.        this.lastName = lastName;
14.        this.age = age;
15.        this.married = married;
16.        this.identityDocument = indetityDocument;
17.        this.profession = profession;
18.        this.Age = age;
19.        IdentityDocument = identityDocument;
20.        Profession = profession;
21.    }
22.
23.    public string Name
24.    {
25.        get { return name; }
26.        set { this.name = name; }
27.    }
28.    public string LastName
29.    {
30.        get { return lastName; }
31.        set { this.lastName = lastName; }
32.    }
33.    public int Age
34.    {
35.        get { return age; }
36.        set { this.age = age; }
37.    }
38.    public bool Married
39.    {
40.        get { return married; }
41.        set { this.married = married; }
42.    }
43.    public string IdentityDocument
```

```
    {
        get { return identityDocument; }
        set { this.identityDocument = identityDocument; }
    }
    public string Profession
    {
        get { return profession; }
        set { this.profession = profession; }
    }

    public void activities()
    {
        Console.WriteLine($"\n{name} {lastName} is a {profession} and activies are:\n");

        switch (profession.ToLower())
        {
            case "software engineer":
                design();
                program();
                solveProblems();
                break;
            case "doctor":
                diagnose();
                operate();
                attendPatients();
                break;
            case "teacher":
                teach();
                evaluate();
                prepareClasses();
                break;
            case "chef":
                cook();
                decoratePlates();
                superviseKitchen();
                break;
            case "firefighter":
                fightFires();
                rescuePeople();
                train();
                break;
            case "artist":
                paint();
                exhibitWorks();
                createSculptures
                ();
                break;
            case "police":
                patrol();
                investigate();
                maintainOrder
                ();
                break;
        }
    }
    private void design() => Console.WriteLine("- Designing structures and systems.");
```

```
100.          private void program() => Console.WriteLine("- Writing code in
   different languages.");
101.          private void solveProblems() => Console.WriteLine("- Analyzing
   and solving technical problems.");
102.
103.          private void diagnose() => Console.WriteLine("- Conducting
   medical diagnoses.");
104.          private void operate() => Console.WriteLine("- Performing
   complex surgeries.");
105.          private void attendPatients() => Console.WriteLine("-
   Providing care to patients.");
106.
107.          private void teach() => Console.WriteLine("- Explaining new
   concepts to students.");
108.          private void evaluate() => Console.WriteLine("- Grading exams
   and assignments.");
109.          private void prepareClasses() => Console.WriteLine("- Creating
   materials for classes.");
110.
111.          private void cook() => Console.WriteLine("- Preparing gourmet
   dishes.");
112.          private void decoratePlates() => Console.WriteLine("-
   Enhancing dishes for customers.");
113.          private void superviseKitchen() => Console.WriteLine("-
   Ensuring quality in the kitchen.");
114.
115.          private void fightFires() => Console.WriteLine("- Combating
   fires in the city.");
116.          private void rescuePeople() => Console.WriteLine("- Saving
   lives in emergencies.");
117.          private void train() => Console.WriteLine("- Training for
   future emergencies.");
118.
119.          private void paint() => Console.WriteLine("- Painting works of
   art.");
120.          private void exhibitWorks() => Console.WriteLine("- Displaying
   creations in galleries.");
121.          private void createSculptures() => Console.WriteLine("-
   Sculpting figures in different materials.");
122.
123.          private void patrol() => Console.WriteLine("- Monitoring
   streets to ensure security.");
124.          private void investigate() => Console.WriteLine("- Analyzing
   cases to solve crimes.");
125.          private void maintainOrder() => Console.WriteLine("- Enforcing
   the law and ensuring public order.");
126.      }
127.          class Program
128.      {
129.          static void Main()
130.          {
131.
132.              Person person1 = new Person("Juan", "Pérez", 35, true,
   "12345678", "software engineer");
133.              Person person2 = new Person("Ana", "López", 28, false,
   "87654321", "doctor");
134.              Person person3 = new Person("Carlos", "Ramírez", 42, true,
   "13579246", "teacher");
```

```
135.             Person person4 = new Person("Elena", "Martínez", 30,
   false, "98765432", "Chef");
136.             Person person5 = new Person("Luis", "González", 50, true,
   "24681357", "firefighter");
137.             Person person6 = new Person("Sofía", "Torres", 22, false,
   "15975368", "artist");
138.             Person person7 = new Person("Miguel", "Hernández", 45,
   true, "36925814", "police");
139.
140.             person1.activities();
141.             person2.activities();
142.             person3.activities();
143.             person4.activities();
144.             person5.activities();
145.             person6.activities();
146.             person7.activities();
147.         }
148.     }
```

Crea una clase Cuenta con los métodos ingreso, reintegro y transferencia. La clase contendrá un constructor por defecto, un constructor con parámetros y los métodos getters y setters para mostrar e ingresar.

```
public class Account
{
   string accountId;
   string nameAccountOwner;
   double balance;

   public Account(string accountId, string nameAccountOwner, double balance)
   {
      this.accountId = accountId;
      this.nameAccountOwner = nameAccountOwner;
      this.balance = balance;
   }

   public String getaccountId()
   {
      return this.accountId;
   }
   public string setAccountId(string accountId)
   {
      return this.accountId = accountId;
```

```
}

    public String getNameAccount() {
   return this.nameAccountOwner;
}

public string setNameAccount(string nameAccountOwner)
{
   return this.nameAccountOwner = nameAccountOwner;
}
   public double getBalance()
{
   return this.balance;
}
public double setBalance(double balance)
{
   return this.balance = balance;
}

public bool income(double amount)
{
   if (amount > 0)
   {
      this.balance += amount;
      return true;
   }
   else
   {
      Console.WriteLine("error, the values is negative");
      return false;
   }
}
public bool deposit(double amount)
{
   if (amount <= 0)
   {
      Console.WriteLine("error, the values no is possitive");
      return false;
   }
   if (balance < amount)
   {
```

```
            Console.WriteLine("Error: insufficient balance");
            return false;
        }

        this.balance = balance - amount;
        return true;
    }
    public bool transfer(Account accountToTransfer,double amount)
    {
        if (amount <= 0)
        {
            Console.WriteLine("error, the values no is possitive");
            return false;
        }
        else if (balance < amount) {
            Console.WriteLine("error: insufficient fund");
            return false;
        }
        else if(accountToTransfer == null )
        {
            Console.WriteLine("error account incorrect");
            return false;
        }

            this.balance -= amount;
        accountToTransfer.income(amount);
        return true;
            }
}
```

Crea una clase Contador con los métodos para incrementar y decrementar el contador. La clase contendrá un constructor por defecto, un constructor con parámetros, y los métodos getters y setters.

```
public class Counter
{
   int count;
   public Counter(int count)
   {
      this.count = count;
   }
   public Counter()
   {
      this.count = 0;
   }
   public int counter
   {
      get { return count; }
      set { count = value; }
   }
   public void Increment()
   {
      count++;

   }
   public void Decrement()
   {
      count--;

   }
```

}

Crea una clase Libro con los métodos préstamo, devolución y ToString. La clase contendrá un constructor por defecto, un constructor con parámetros y los métodos getters y setters.

```
public class Book
{
   private string name;
   private string autor;
   private int year;
   private int page;
   private bool lending;
   public Book()
   {
      this.name = "";
      this.autor = "";
      this.year = 0;
      this.page = 0;
      this.lending = false ;
   }
   public Book(string name, string autor, int year, int page)
   {
      this.name = name;
      this.autor = autor;
      this.year = year;
      this.page = page;
      this.lending = lending;
   }
```

```
public string Name
{
   get { return this.name; }
   set { this.name = value; }
}
public string Autor
{
   get { return this.autor; }
   set { this.autor = value; }
}
public int Year
{
   get { return this.year; }
   set { this.year = value; }
}
public int Pages
{
   get { return this.page; }
   set { this.page = value; }
}
public bool Lending
{
   get { return lending; }
   set { lending = value; }
}
public bool loanBook()
{
```

```
        if (lending)
        {
            return false;
        }
        lending = true;
        return true;

    }
    public bool returnBook() {



        if (!lending)
        {
            return false;
            }
        lending = false;
        return true;
    }
    public string BookStatuTostring()
    {
        string status = lending ? "book not available" : "Book available";
        return $"book: {name}, Autor: {autor}, statu: {status}";
    }
}
```

Crea una clase Fracción con métodos para sumar, restar, multiplicar y dividir fracciones.

```
class Fraction
{
    private int numerator;
    private int denominator;

    public Fraction(int numerator, int denominator)
    {
        if (denominator == 0)
        {
            throw new ArgumentException("Denominator cannot be zero.");
        }

        this.numerator = numerator;
        this.denominator = denominator;
        Simplify();


    }
    private int GreatestCommonDivisor(int a, int b)
    {
        while (b != 0)
        {
            int temp = b;
            b = a % b;
            a = temp;
```

```
        }
        return Math.Abs(a);
    }

    private void Simplify()
    {
        int gcd = GreatestCommonDivisor(numerator, denominator);
        numerator /= gcd;
        denominator /= gcd;

        if (denominator < 0)
        {
            numerator = -numerator;
            denominator = -denominator;
        }
    }
    public Fraction Add(Fraction other)
    {
        int newNumerator = (this.numerator * other.denominator) + (other.numerator *
this.denominator);
        int newDenominator = this.denominator * other.denominator;
        return new Fraction(newNumerator, newDenominator);
    }

    public Fraction Subtract(Fraction other)
    {
        int newNumerator = (this.numerator * other.denominator) - (other.numerator *
this.denominator);
        int newDenominator = this.denominator * other.denominator;
        return new Fraction(newNumerator, newDenominator);
```

```
    }

    public Fraction Multiply(Fraction other)
    {
        int newNumerator = this.numerator * other.numerator;
        int newDenominator = this.denominator * other.denominator;
        return new Fraction(newNumerator, newDenominator);
    }

    public Fraction Divide(Fraction other)
    {
        if (other.numerator == 0)
        {
            throw new ArgumentException("Cannot divide by a fraction with numerator zero.");
        }

        int newNumerator = this.numerator * other.denominator;
        int newDenominator = this.denominator * other.numerator;
        return new Fraction(newNumerator, newDenominator);
    }

    public override string ToString()
    {
        return $"{numerator}/{denominator}";
    }
}
```