Universidad Nacional de San Agustín
Escuela Profesional de Ingeniería de Sistemas
Fundamentos de Programación II
Tema N° 10:
Programación Orientada a Objetos IV

Nombre: Jhonatan Benjamin Mamani Céspedes **CUI: 20232188**

Link de repositorio GitHub: https://github.com/JBenjamin01/fp2-24b

Ejercicio 1:

- Crear la clase Car para que funcione con el siguiente código:

```
public static void main(String[] args){
  Car johnCar = new Car();
  Car stacyCar;

  johnCar.setMake("Honda");
  johnCar.setYear(2003);
  johnCar.setColor("silver");
  stacyCar = johnCar.makeCopy();
  stacyCar.setColor("peach");
}
```

Clase Car.java:

```java
public class Car {
    private String make;
    private int year;
    private String color;

    public void setMake(String make) {
        this.make = make;
    }

    public void setYear(int year) {
        this.year = year;
    }

    public void setColor(String color) {
        this.color = color;
    }

    public Car makeCopy() {
        Car copy = new Car();
        copy.setMake(this.make);
        copy.setYear(this.year);
        copy.setColor(this.color);
        return copy;
    }
}
```

Clase Main.java:

```java
public class Main {
    public static void main(String[] args){
        Car johnCar = new Car();
        Car stacyCar;

        johnCar.setMake("Honda");
        johnCar.setYear(2003);
        johnCar.setColor("silver");
        stacyCar = johnCar.makeCopy();
        stacyCar.setColor("peach");
    }
}
```

Ejercicio 2:

- Escribir el método `equals` para la clase Car y probar con la siguiente Aplicación:

```
public class AplicacionAuto{
  public static void main(String[] args){
    ...
    if(johnCar.equals(stacyCar))
          System.out.println("iguales");
    else
          System.out.println("diferentes");
  }
}
```

Clase Car.java:

```java
public class Car {
    private String make;
    private int year;
    private String color;

    public void setMake(String make) {
        this.make = make;
    }

    public void setYear(int year) {
        this.year = year;
    }

    public void setColor(String color) {
        this.color = color;
    }

    public Car makeCopy() {
        Car copy = new Car();
        copy.setMake(this.make);
        copy.setYear(this.year);
        copy.setColor(this.color);
        return copy;
    }

    public boolean equals(Car c) {
        return this.make.equals(c.make) && this.year == c.year && this.color.equals(c.color);
    }
}
```

Clase AplicacionAuto.java:

```java
public class AplicacionAuto {
    public static void main(String[] args){

        Car johnCar = new Car();
        Car stacyCar;

        johnCar.setMake("Honda");
        johnCar.setYear(2003);
        johnCar.setColor("silver");
        stacyCar = johnCar.makeCopy();

        if(johnCar.equals(stacyCar))
            System.out.println("iguales");
        else
            System.out.println("diferentes");

    }
}
```

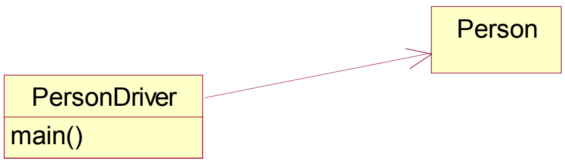
(Para probar el `equals` retornando true, eliminé el `setColor("Peach")` del anterior Main, y asi, solo lo dejé como la copia del otro objeto Car)

Consola:

```
PS C:\Users\jhona\OneDrive\Doc
neDrive\Documentos\University\
e' '-XX:+ShowCodeDetailsInExce
' 'AplicacionAuto'
iguales
```

Ejercicio 3:

- Dado el siguiente diagrama de clases, implementar la clase `Person`, de tal forma que funcione con la siguiente aplicación:




```java
public class PersonDriver{
  public static void main(String[] args){
    Person person1 = new Person();
    Person person2 = new Person();

    person1.setName("Jonathan");
    person2.setName("Benji");
    System.out.println(person1.getName()
      + ", " + person2.getName());

    person1.swapPerson(person2);
    System.out.println(person1.getName()
      + ", " + person2.getName());
  }
}
```

Clase Person.java:

```java
public class Person {
    private String name;

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void swapPerson(Person p) {
        String tempName = this.name;
        this.name = p.name;
        p.name = tempName;
    }
}
```

Clase PersonDriver.java:

```java
public class PersonDriver {
    public static void main(String[] args){
        Person person1 = new Person();
        Person person2 = new Person();

        person1.setName("Jonathan");
        person2.setName("Benji");
        System.out.println(person1.getName()
        + ", " + person2.getName());

        person1.swapPerson(person2);
        System.out.println(person1.getName()
        + ", " + person2.getName());
    }
}
```

Consola:

```
PS C:\Users\jhona\OneDrive\Documentos\
m Files\Java\jdk-21\bin\java.exe' '-XX
ava\jdt_ws\Ejercicio 3_3a082970\bin' '
Jonathan, Benji
Benji, Jonathan
```

Ejercicio 4:

- Crear la clase Fraccion con constructores sobrecargados, que se verifique la consistencia de sus datos y le dé valores por defecto en caso sean erróneos. sets, gets. También que permita mostrar en diferentes formatos: a/b, x.yzw. Y que permita retornar la suma, resta, multiplicación y división de la fracción con otro objeto fracción (el objeto llamado queda intacto). Además que permita retornar la fracción simplificada (el objeto llamado queda intacto). Evitar la duplicación de código.

```
suma = f1.sumar(f2);
```

Clase Fraccion.java:

```java
public class Fraccion {
    private int numerador;
    private int denominador;

    public Fraccion() {
        this.numerador = 0;
        this.denominador = 1;
    }

    public Fraccion(Fraccion f) {
        this.numerador = f.numerador;
        this.denominador = f.denominador;
    }

    // Aquí se maneja la consistencia de los atributos de la fracción
    // También se le da los valores por defecto en caso de estar erróneos
    public Fraccion(int numerador, int denominador) {
        if (denominador == 0) {
            this.numerador = 0;
            this.denominador = 1;
        } else {
            this.numerador = numerador;
            this.denominador = denominador;
        }
    }

    public int getNumerador() {
        return numerador;
    }

    public int getDenominador() {
        return denominador;
    }

    public void setNumerador(int numerador) {
        this.numerador = numerador;
    }

    public void setDenominador(int denominador) {
        if (denominador != 0) {
            this.denominador = denominador;
        }
    }
```

```java

    // Los formatos para mostrar la fracción
    public String mostrarFraccion() {
        return numerador + "/" + denominador;
    }

    public String mostrarDecimal() {
        return String.format("%.3f", (double) numerador / denominador);
    }

    // Operaciones
    public Fraccion sumar(Fraccion f) {
        int nuevoNumerador = this.numerador * f.denominador + f.numerador * this.denominador;
        int nuevoDenominador = this.denominador * f.denominador;
        return new Fraccion(nuevoNumerador, nuevoDenominador);
    }

    public Fraccion restar(Fraccion f) {
        int nuevoNumerador = this.numerador * f.denominador - f.numerador * this.denominador;
        int nuevoDenominador = this.denominador * f.denominador;
        return new Fraccion(nuevoNumerador, nuevoDenominador);
    }

    public Fraccion multiplicar(Fraccion f) {
        int nuevoNumerador = this.numerador * f.numerador;
        int nuevoDenominador = this.denominador * f.denominador;
        return new Fraccion(nuevoNumerador, nuevoDenominador);
    }

    public Fraccion dividir(Fraccion f) {
        int nuevoNumerador = this.numerador * f.denominador;
        int nuevoDenominador = this.denominador * f.numerador;
        return new Fraccion(nuevoNumerador, nuevoDenominador);
    }

    public Fraccion simplificar() {
        int mcd = mcd(numerador, denominador);
        return new Fraccion(numerador / mcd, denominador / mcd);
    }

    // Maximo comun divisor privado
    private int mcd(int a, int b) {
        while (b != 0) {
            int temp = b;
            b = a % b;
            a = temp;
        }
        return a;
    }
}
```

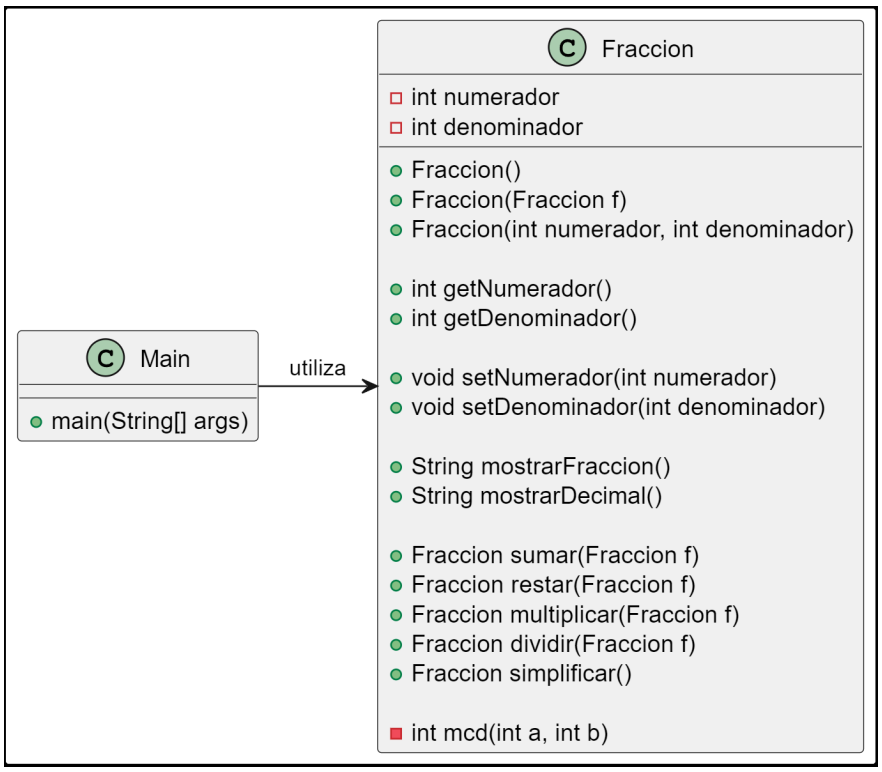
Clase Main.java:

```java
public class Main {
    public static void main(String[] args) {
        Fraccion f1 = new Fraccion(3, 4);
        Fraccion f2 = new Fraccion(5, 6);

        System.out.println("Fracción 1: " + f1.mostrarFraccion());
        System.out.println("Fracción 1 en decimal: " + f1.mostrarDecimal());

        System.out.println("Fracción 2: " + f2.mostrarFraccion());
        System.out.println("Fracción 2 en decimal: " + f2.mostrarDecimal());

        Fraccion suma = f1.sumar(f2);
        System.out.println("Suma: " + suma.mostrarFraccion() + " o " + suma.mostrarDecimal());

        Fraccion resta = f1.restar(f2);
        System.out.println("Resta: " + resta.mostrarFraccion() + " o " + resta.mostrarDecimal());

        Fraccion multiplicacion = f1.multiplicar(f2);
        System.out.println("Multiplicación: " + multiplicacion.mostrarFraccion() + " o " + multiplicacion.mostrarDecimal());

        Fraccion division = f1.dividir(f2);
        System.out.println("División: " + division.mostrarFraccion() + " o " + division.mostrarDecimal());

        Fraccion f3 = new Fraccion(16, 28);
        Fraccion f3Simplificada = f3.simplificar();
        System.out.println("Fracción original: " + f3.mostrarFraccion());
        System.out.println("Fracción simplificada: " + f3Simplificada.mostrarFraccion());
    }
}
```

Consola:

```
PS C:\Users\jhona\OneDrive\Documentos\University\Universidad Nacional
s\jhona\OneDrive\Documentos\University\Universidad Nacional de San Agu
jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' '
icio 4_3a082971\bin' 'Main'
Fracción 1: 3/4
Fracción 1 en decimal: 0.750
Fracción 2: 5/6
Fracción 2 en decimal: 0.833
Suma: 38/24 o 1.583
Resta: -2/24 o -0.083
Multiplicación: 15/24 o 0.625
División: 18/20 o 0.900
Fracción original: 16/28
Fracción simplificada: 4/7
```

Diagrama UML:



Ejercicio 5:

- Crear la clase Fraccion como en el anterior ejemplo, pero que permita el encadenamiento de llamadas a métodos (f1 puede cambiar su valor)

```
f = f1.sumar(f2).sumar(f3).restar(f4).mult(f5);
```

Clase Fraccion.java:

Tomando como referencia la anterior clase `Fraccion.java` no es necesario volver a hacer los constructores, getters, setters y los métodos para mostrar la fracción, en su lugar, solo se ha modificado la manera en la que se trabajan las *operaciones (incluido simplificar)* modificando el objeto que se llama en cuestión, es decir, usando `this`, los cambios se aplican al objeto al que se está invocando en los métodos de instancia, no se crea uno nuevo para retornar.

Además, implementé el método `simplificar()` dentro de las operaciones para evitar que los valores de numerador y denominador puedan escalar a cantidades enormes que sobrepasen el límite de caracteres y también para obtener un resultado simplificado de manera directa desde la operación.

```java
// Operaciones
public Fraccion sumar(Fraccion f) {
    this.numerador = this.numerador * f.denominador + f.numerador * this.denominador;
    this.denominador = this.denominador * f.denominador;
    simplificar();
    return this;
}

public Fraccion restar(Fraccion f) {
    this.numerador = this.numerador * f.denominador - f.numerador * this.denominador;
    this.denominador = this.denominador * f.denominador;
    simplificar();
    return this;
}

public Fraccion multiplicar(Fraccion f) {
    this.numerador = this.numerador * f.numerador;
    this.denominador = this.denominador * f.denominador;
    simplificar();
    return this;
}

public Fraccion dividir(Fraccion f) {
    this.numerador = this.numerador * f.denominador;
    this.denominador = this.denominador * f.numerador;
    simplificar();
    return this;
}

public Fraccion simplificar() {
    int mcd = mcd(numerador, denominador);
    this.numerador /= mcd;
    this.denominador /= mcd;
    return this;
}
```

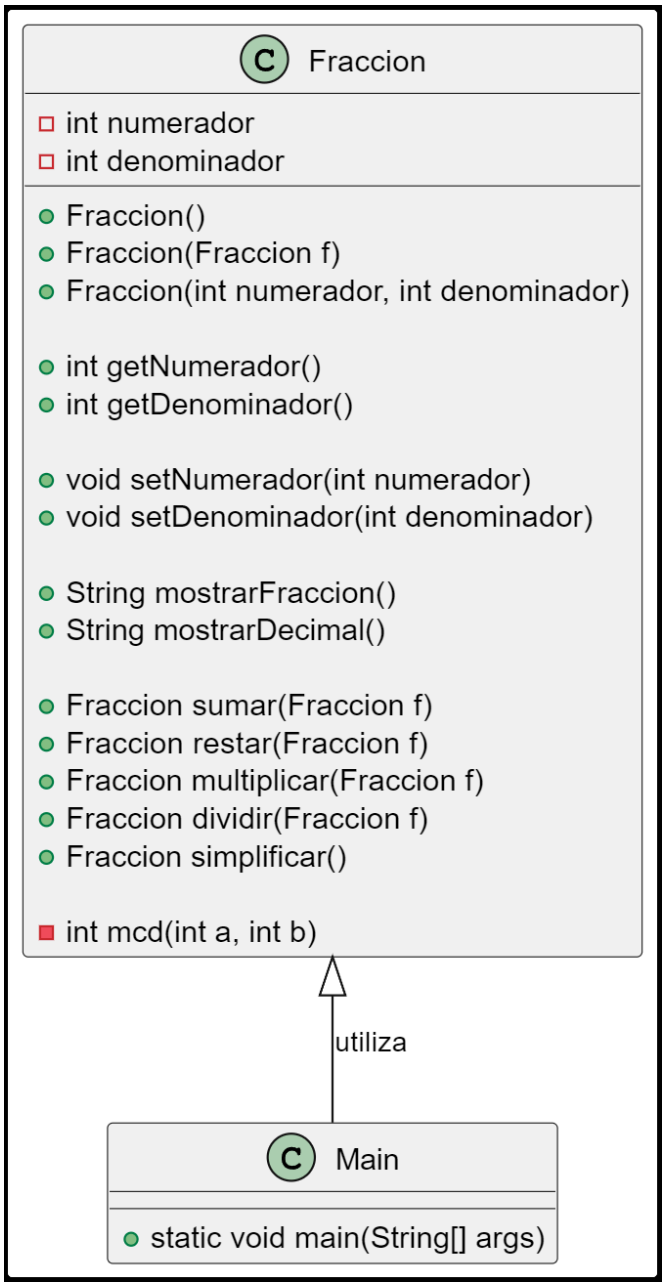
Clase Main.java:

```java
public class Main {
    public static void main(String[] args) {
        Fraccion f1 = new Fraccion(3, 4);
        Fraccion f2 = new Fraccion(7, 9);
        Fraccion f3 = new Fraccion(2, 6);
        Fraccion f4 = new Fraccion(5, 20);
        Fraccion f5 = new Fraccion(3, 7);

        // Encadenamiento de llamadas a métodos
        f1.sumar(f2).sumar(f3).restar(f4).multiplicar(f5);

        System.out.println("Resultado final: " + f1.mostrarFraccion() + " o " + f1.mostrarDecimal());
    }
}
```

Consola:

```
PS C:\Users\jhona\OneDrive\Documentos\Unive
sers\jhona\OneDrive\Documentos\University\U
\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDe
_ws\Ejercicio 5_3a082972\bin' 'Main'
Resultado final: 29/42 o 0.690
```

Diagrama UML:



Ejercicio 6:

- Crear la clase Fraccion como en el anterior ejemplo, pero que permita el encadenamiento de llamadas a métodos (f1 NO puede cambiar su valor)

```
f = f1.sumar(f2).sumar(f3).restar(f4).mult(f5);
```

Clase Fraccion.java:

En este ejercicio únicamente cambio el `this` de las operaciones para trabajar con la instancia a la que se llama, en este caso, un nuevo objeto `Fraccion` que contiene el resultado.

```java
// Operaciones que no modifican la instancia actual
public Fraccion sumar(Fraccion f) {
    int nuevoNumerador = this.numerador * f.denominador + f.numerador * this.denominador;
    int nuevoDenominador = this.denominador * f.denominador;
    Fraccion resultado = new Fraccion(nuevoNumerador, nuevoDenominador);
    resultado.simplificar();
    return resultado;
}

public Fraccion restar(Fraccion f) {
    int nuevoNumerador = this.numerador * f.denominador - f.numerador * this.denominador;
    int nuevoDenominador = this.denominador * f.denominador;
    Fraccion resultado = new Fraccion(nuevoNumerador, nuevoDenominador);
    resultado.simplificar();
    return resultado;
}

public Fraccion multiplicar(Fraccion f) {
    int nuevoNumerador = this.numerador * f.numerador;
    int nuevoDenominador = this.denominador * f.denominador;
    Fraccion resultado = new Fraccion(nuevoNumerador, nuevoDenominador);
    resultado.simplificar();
    return resultado;
}

public Fraccion dividir(Fraccion f) {
    int nuevoNumerador = this.numerador * f.denominador;
    int nuevoDenominador = this.denominador * f.numerador;
    Fraccion resultado = new Fraccion(nuevoNumerador, nuevoDenominador);
    resultado.simplificar();
    return resultado;
}

public Fraccion simplificar() {
    int mcd = mcd(numerador, denominador);
    this.numerador /= mcd;
    this.denominador /= mcd;
    return this;
}
```

Clase Main.java:

```java
public class Main {
    public static void main(String[] args) {
        Fraccion f1 = new Fraccion(3, 4);
        Fraccion f2 = new Fraccion(7, 9);
        Fraccion f3 = new Fraccion(2, 6);
        Fraccion f4 = new Fraccion(5, 20);
        Fraccion f5 = new Fraccion(3, 7);

        // Encadenamiento de llamadas a métodos, sin modificar f1
        Fraccion resultado = f1.sumar(f2).sumar(f3).restar(f4).multiplicar(f5);

        System.out.println("Resultado final: " + resultado.mostrarFraccion() + " o " + resultado.mostrarDecimal());
        System.out.println("Valor original de f1: " + f1.mostrarFraccion()); // Aqui puedo comprobar que f1 sigue siendo 3/4
    }
}
```

Consola:

```
PS C:\Users\jhona\OneDrive\Documentos\University\
m Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeD
ava\jdt_ws\Ejercicio 6_3a082973\bin' 'Main'
Resultado final: 29/42 o 0.690
Valor original de f1: 3/4
```

Diagrama UML: