

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:

```
1 public class Car {  
2     private String make;  
3     private int year;  
4     private String color;  
5  
6     public void setMake(String make) {  
7         this.make = make;  
8     }  
9  
10    public void setYear(int year) {  
11        this.year = year;  
12    }  
13  
14    public void setColor(String color) {  
15        this.color = color;  
16    }  
17  
18    public Car makeCopy() {  
19        Car copy = new Car();  
20        copy.setMake(this.make);  
21        copy.setYear(this.year);  
22        copy.setColor(this.color);  
23        return copy;  
24    }  
25 }
```

Clase Main.java:

```
1 public class Main {
2     public static void main(String[] args){
3         Car johnCar = new Car();
4         Car stacyCar;
5
6         johnCar.setMake("Honda");
7         johnCar.setYear(2003);
8         johnCar.setColor("silver");
9         stacyCar = johnCar.makeCopy();
10        stacyCar.setColor("peach");
11    }
12 }
```

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:

```
1 public class Car {
2     private String make;
3     private int year;
4     private String color;
5
6     public void setMake(String make) {
7         this.make = make;
8     }
9
10    public void setYear(int year) {
11        this.year = year;
12    }
13
14    public void setColor(String color) {
15        this.color = color;
16    }
17
18    public Car makeCopy() {
19        Car copy = new Car();
20        copy.setMake(this.make);
21        copy.setYear(this.year);
22        copy.setColor(this.color);
23        return copy;
24    }
25
26    public boolean equals(Car c) {
27        return this.make.equals(c.make) && this.year == c.year && this.color.equals(c.color);
28    }
29 }
```

Clase AplicacionAuto.java:

```
1 public class AplicacionAuto {
2     public static void main(String[] args){
3
4         Car johnCar = new Car();
5         Car stacyCar;
6
7         johnCar.setMake("Honda");
8         johnCar.setYear(2003);
9         johnCar.setColor("silver");
10        stacyCar = johnCar.makeCopy();
11
12        if(johnCar.equals(stacyCar))
13            System.out.println("iguales");
14        else
15            System.out.println("diferentes");
16
17    }
18 }
```

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

Consola:

```
PS C:\Users\jhona\OneDrive\Documents\University\
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:



```
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:

```
1 public class Person {
2     private String name;
3
4     public void setName(String name) {
5         this.name = name;
6     }
7
8     public String getName() {
9         return name;
10    }
11
12    public void swapPerson(Person p) {
13        String tempName = this.name;
14        this.name = p.name;
15        p.name = tempName;
16    }
17 }
```

Clase PersonDriver.java:

```
1 public class PersonDriver {
2     public static void main(String[] args){
3         Person person1 = new Person();
4         Person person2 = new Person();
5
6         person1.setName("Jonathan");
7         person2.setName("Benji");
8         System.out.println(person1.getName()
9         + ", " + person2.getName());
10
11        person1.swapPerson(person2);
12        System.out.println(person1.getName()
13        + ", " + person2.getName());
14    }
15 }
```

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:

```
1  public class Fraccion {
2      private int numerador;
3      private int denominador;
4
5      public Fraccion() {
6          this.numerador = 0;
7          this.denominador = 1;
8      }
9
10     public Fraccion(Fraccion f) {
11         this.numerador = f.numerador;
12         this.denominador = f.denominador;
13     }
14
15     // Aquí se maneja la consistencia de los atributos de la fracción
16     // También se le da los valores por defecto en caso de estar erróneos
17     public Fraccion(int numerador, int denominador) {
18         if (denominador == 0) {
19             this.numerador = 0;
20             this.denominador = 1;
21         } else {
22             this.numerador = numerador;
23             this.denominador = denominador;
24         }
25     }
26
27     public int getNumerador() {
28         return numerador;
29     }
30
31     public int getDenominador() {
32         return denominador;
33     }
34
35     public void setNumerador(int numerador) {
36         this.numerador = numerador;
37     }
38
39     public void setDenominador(int denominador) {
40         if (denominador != 0) {
41             this.denominador = denominador;
42         }
43     }
44 }
```

```

44
45 // Los formatos para mostrar la fracción
46 public String mostrarFraccion() {
47     return numerador + "/" + denominador;
48 }
49
50 public String mostrarDecimal() {
51     return String.format("%.3f", (double) numerador / denominador);
52 }
53
54 // Operaciones
55 public Fraccion sumar(Fraccion f) {
56     int nuevoNumerador = this.numerador * f.denominador + f.numerador * this.denominador;
57     int nuevoDenominador = this.denominador * f.denominador;
58     return new Fraccion(nuevoNumerador, nuevoDenominador);
59 }
60
61 public Fraccion restar(Fraccion f) {
62     int nuevoNumerador = this.numerador * f.denominador - f.numerador * this.denominador;
63     int nuevoDenominador = this.denominador * f.denominador;
64     return new Fraccion(nuevoNumerador, nuevoDenominador);
65 }
66
67 public Fraccion multiplicar(Fraccion f) {
68     int nuevoNumerador = this.numerador * f.numerador;
69     int nuevoDenominador = this.denominador * f.denominador;
70     return new Fraccion(nuevoNumerador, nuevoDenominador);
71 }
72
73 public Fraccion dividir(Fraccion f) {
74     int nuevoNumerador = this.numerador * f.denominador;
75     int nuevoDenominador = this.denominador * f.numerador;
76     return new Fraccion(nuevoNumerador, nuevoDenominador);
77 }
78
79 public Fraccion simplificar() {
80     int mcd = mcd(numerador, denominador);
81     return new Fraccion(numerador / mcd, denominador / mcd);
82 }
83
84 // Maximo comun divisor privado
85 private int mcd(int a, int b) {
86     while (b != 0) {
87         int temp = b;
88         b = a % b;
89         a = temp;
90     }
91     return a;
92 }
93 }

```

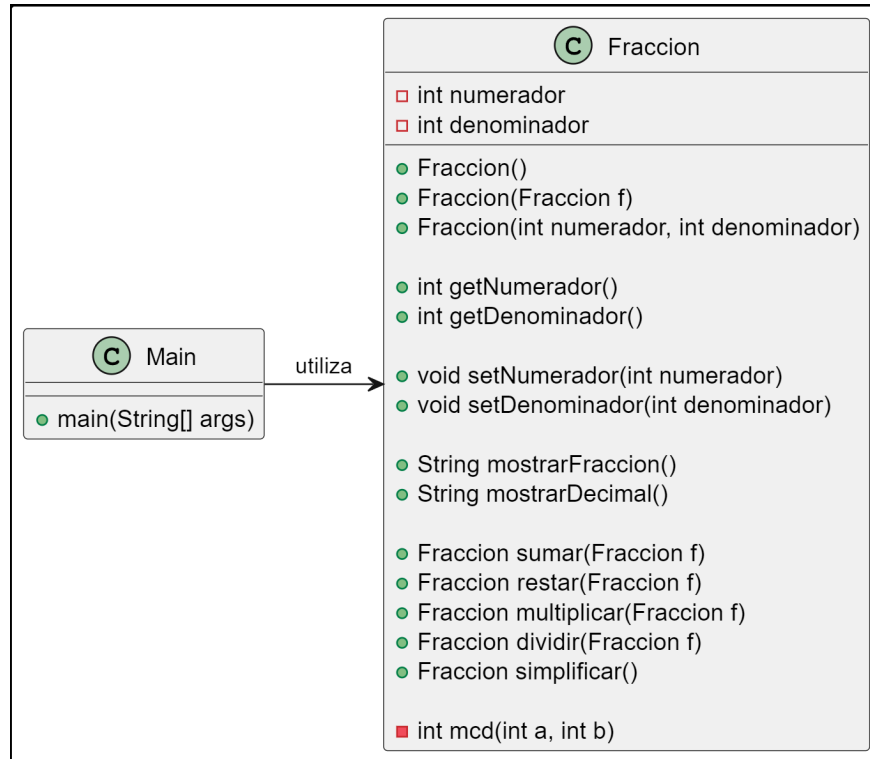
Clase Main.java:

```
1 public class Main {
2     public static void main(String[] args) {
3         Fraccion f1 = new Fraccion(3, 4);
4         Fraccion f2 = new Fraccion(5, 6);
5
6         System.out.println("Fracción 1: " + f1.mostrarFraccion());
7         System.out.println("Fracción 1 en decimal: " + f1.mostrarDecimal());
8
9         System.out.println("Fracción 2: " + f2.mostrarFraccion());
10        System.out.println("Fracción 2 en decimal: " + f2.mostrarDecimal());
11
12        Fraccion suma = f1.sumar(f2);
13        System.out.println("Suma: " + suma.mostrarFraccion() + " o " + suma.mostrarDecimal());
14
15        Fraccion resta = f1.restar(f2);
16        System.out.println("Resta: " + resta.mostrarFraccion() + " o " + resta.mostrarDecimal());
17
18        Fraccion multiplicacion = f1.multiplicar(f2);
19        System.out.println("Multiplicación: " + multiplicacion.mostrarFraccion() + " o " + multiplicacion.mostrarDecimal());
20
21        Fraccion division = f1.dividir(f2);
22        System.out.println("División: " + division.mostrarFraccion() + " o " + division.mostrarDecimal());
23
24        Fraccion f3 = new Fraccion(16, 28);
25        Fraccion f3Simplificada = f3.simplificar();
26        System.out.println("Fracción original: " + f3.mostrarFraccion());
27        System.out.println("Fracción simplificada: " + f3Simplificada.mostrarFraccion());
28    }
29 }
```

Consola:

```
PS C:\Users\jhona\OneDrive\Documentos\University\Universidad Nacional de San Agustín\jdhonaj\jdk-21\bin> java.exe -XX:+ShowCodeDetailsInExceptionMessages -cp 'C:\Users\jhona\OneDrive\Documentos\University\Universidad Nacional de San Agustín\jdhonaj\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.


```

1  // Operaciones
2  public Fraccion sumar(Fraccion f) {
3      this.numerador = this.numerador * f.denominador + f.numerador * this.denominador;
4      this.denominador = this.denominador * f.denominador;
5      simplificar();
6      return this;
7  }
8
9  public Fraccion restar(Fraccion f) {
10     this.numerador = this.numerador * f.denominador - f.numerador * this.denominador;
11     this.denominador = this.denominador * f.denominador;
12     simplificar();
13     return this;
14 }
15
16 public Fraccion multiplicar(Fraccion f) {
17     this.numerador = this.numerador * f.numerador;
18     this.denominador = this.denominador * f.denominador;
19     simplificar();
20     return this;
21 }
22
23 public Fraccion dividir(Fraccion f) {
24     this.numerador = this.numerador * f.denominador;
25     this.denominador = this.denominador * f.numerador;
26     simplificar();
27     return this;
28 }
29
30 public Fraccion simplificar() {
31     int mcd = mcd(numerador, denominador);
32     this.numerador /= mcd;
33     this.denominador /= mcd;
34     return this;
35 }

```

Clase Main.java:

```

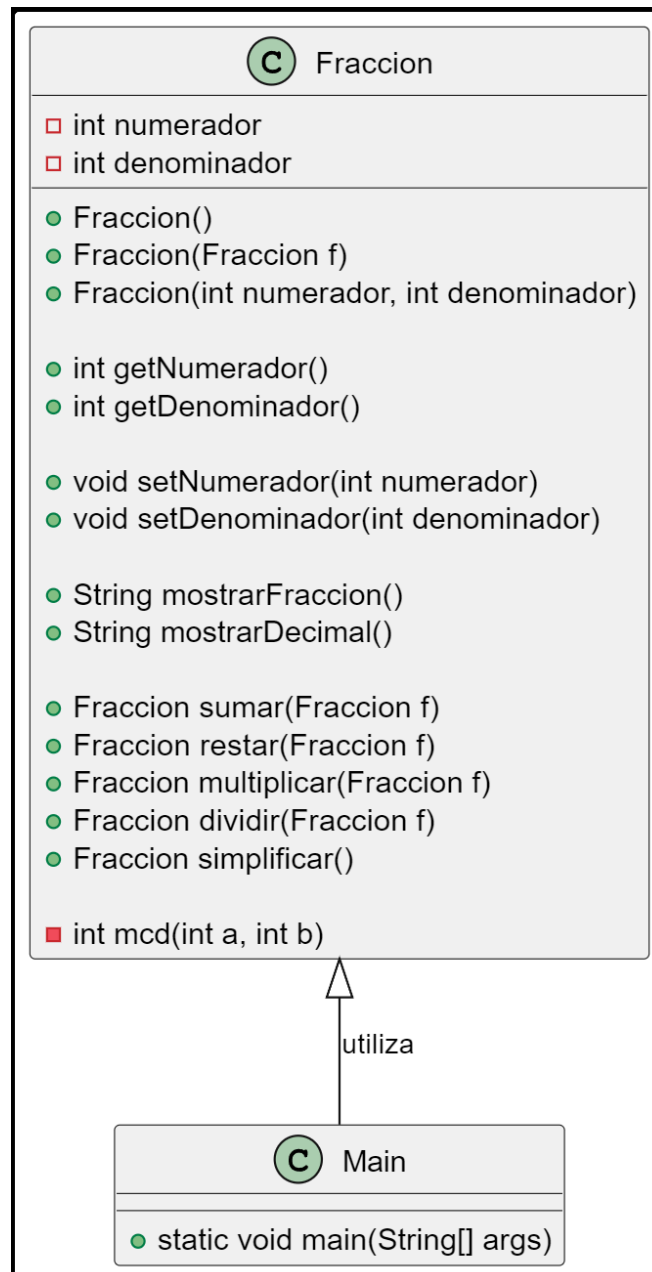
1  public class Main {
2      public static void main(String[] args) {
3          Fraccion f1 = new Fraccion(3, 4);
4          Fraccion f2 = new Fraccion(7, 9);
5          Fraccion f3 = new Fraccion(2, 6);
6          Fraccion f4 = new Fraccion(5, 20);
7          Fraccion f5 = new Fraccion(3, 7);
8
9          // Encadenamiento de llamadas a métodos
10         f1.sumar(f2).sumar(f3).restar(f4).multiplicar(f5);
11
12         System.out.println("Resultado final: " + f1.mostrarFraccion() + " o " + f1.mostrarDecimal());
13     }
14 }

```

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.

```
1 // Operaciones que no modifican la instancia actual
2 public Fraccion sumar(Fraccion f) {
3     int nuevoNumerador = this.numerador * f.denominador + f.numerador * this.denominador;
4     int nuevoDenominador = this.denominador * f.denominador;
5     Fraccion resultado = new Fraccion(nuevoNumerador, nuevoDenominador);
6     resultado.simplificar();
7     return resultado;
8 }
9
10 public Fraccion restar(Fraccion f) {
11     int nuevoNumerador = this.numerador * f.denominador - f.numerador * this.denominador;
12     int nuevoDenominador = this.denominador * f.denominador;
13     Fraccion resultado = new Fraccion(nuevoNumerador, nuevoDenominador);
14     resultado.simplificar();
15     return resultado;
16 }
17
18 public Fraccion multiplicar(Fraccion f) {
19     int nuevoNumerador = this.numerador * f.numerador;
20     int nuevoDenominador = this.denominador * f.denominador;
21     Fraccion resultado = new Fraccion(nuevoNumerador, nuevoDenominador);
22     resultado.simplificar();
23     return resultado;
24 }
25
26 public Fraccion dividir(Fraccion f) {
27     int nuevoNumerador = this.numerador * f.denominador;
28     int nuevoDenominador = this.denominador * f.numerador;
29     Fraccion resultado = new Fraccion(nuevoNumerador, nuevoDenominador);
30     resultado.simplificar();
31     return resultado;
32 }
33
34 public Fraccion simplificar() {
35     int mcd = mcd(numerador, denominador);
36     this.numerador /= mcd;
37     this.denominador /= mcd;
38     return this;
39 }
```

Clase Main.java:

```
1 public class Main {
2     public static void main(String[] args) {
3         Fraccion f1 = new Fraccion(3, 4);
4         Fraccion f2 = new Fraccion(7, 9);
5         Fraccion f3 = new Fraccion(2, 6);
6         Fraccion f4 = new Fraccion(5, 20);
7         Fraccion f5 = new Fraccion(3, 7);
8
9         // Encadenamiento de Llamadas a métodos, sin modificar f1
10        Fraccion resultado = f1.sumar(f2).sumar(f3).restar(f4).multiplicar(f5);
11
12        System.out.println("Resultado final: " + resultado.mostrarFraccion() + " o " + resultado.mostrarDecimal());
13        System.out.println("Valor original de f1: " + f1.mostrarFraccion()); // Aquí puedo comprobar que f1 sigue siendo 3/4
14    }
15 }
```

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:

