

Práctica N°13 de SIS103

Nombre: Gonzales Suyo Franz Reinaldo

C.U. 111-500

Carrera: Ing. Ciencias de la Computación

Ejercicio 4.7. Clases abstractas

Clase Animal:

```
/**
 * Esta clase abstracta denominada Animal modela un animal genérico
 * que cuenta con atributos como un sonido, alimentos que consume,
 * un haabitat y un nombre científico.
 *
 * @version 1.2/2020
 */
public abstract class Animal {
    protected String sonido; /*
        * Atributo que identifica el sonido emitido
        * por un animal
        */
    protected String alimentos; /*
        * Atributo que identifica los alimentos
        * que consume un animal
        */
    protected String haabitat; /*
        * Atributo que identifica el haabitat de un
        * animal
        */
    protected String nombreCientifico; /*
        * Atributo que identifica el
        * nombre científico de un animal
        */

    /**
     * Método abstracto que permite obtener el nombre científico del animal
     *
     * @return El nombre científico del animal
     */
    public abstract String getNombreCientifico();

    /**
     * Método abstracto que permite obtener el sonido producido por el
     * animal
     *
     * @return El sonido producido por el animal
     */
}
```

```

    public abstract String getSonido();

    /**
     * Método abstracto que permite obtener los alimentos que consume
     * un animal
     *
     * @return Los alimentos que consume el animal
     */
    public abstract String getAlimentos();

    /**
     * Método abstracto que permite obtener el haabitat de un animal
     *
     * @return El haabitat del animal
     */
    public abstract String getHaabitat();
}

```

Clase Canido:

```

/**
 * Esta clase abstracta denominada Cánido modela esta familia de
 * animales. Es una subclase de Animal.
 * @version 1.2/2020
 */
public abstract class Canido extends Animal {
}

```

Clase Perro:

```

/**
 * Esta clase concreta denominada Perro es una subclase de Canido.
 *
 * @version 1.2/2020
 */
public class Perro extends Canido {

    /**
     * Método que devuelve un String con el sonido de un perro
     * @return Un valor String con el sonido de un perro: "Ladrado"
     */
    public String getSonido() {
        return "Ladrado";
    }

    /**

```

```

    * Método que devuelve un String con los alimentos de un perro
    * @return Un valor String con la alimentación de un perro: "Carnívoro"
    */
    public String getAlimentos() {
        return "Carnívoro";
    }

    /**
    * Método que devuelve un String con el hábitat de un perro
    * @return Un valor String con el hábitat de un perro: "Doméstico"
    */
    public String getHaabitat() {
        return "Doméstico";
    }

    /**
    * Método que devuelve un String con el nombre científico de un perro
    * @return Un valor String con el nombre científico de un perro:
    * "Canis lupus familiaris"
    */
    public String getNombreCientifico() {
        return "Canis lupus familiaris";
    }
}

```

Clase Lobo:

```

/**
 * Esta clase concreta denominada Lobo es una subclase de Canido.
 *
 * @version 1.2/2020
 */
public class Lobo extends Canido {
    /**
    * Método que devuelve un String con el sonido de un lobo
    * @return Un valor String con el sonido de un lobo: "Aullido"
    */
    public String getSonido() {
        return "Aullido";
    }

    /**
    * Método que devuelve un String con los alimentos de un lobo
    * @return Un valor String con el tipo de alimentación de un lobo:
    * "Carnívoro"
    */
    public String getAlimentos() {
        return "Carnívoro";
    }
}

```

```

/**
 * Método que devuelve un String con el hábitat de un lobo
 * @return Un valor String con el hábitat de un lobo: "Bosque"
 */
public String getHaabitat() {
    return "Bosque";
}

/**
 * Método que devuelve un String con el nombre científico de un lobo
 * @return Un valor String con el nombre científico de un lobo:
 * "Canis lupus"
 */
public String getNombreCientifico() {
    return "Canis lupus";
}
}

```

Clase Felino:

```

/**
 * Esta clase abstracta denominada Felino modela esta familia de
 * animales. Es una subclase de Animal.
 *
 * @version 1.2/2020
 */
public abstract class Felino extends Animal {
}

```

Clase Leon:

```

/**
 * Esta clase concreta denominada León es una subclase de Felino.
 *
 * @version 1.2/2020
 */
public class Leon extends Felino {
    /**
     * Método que devuelve un String con el sonido de un león
     * @return Un valor String con el sonido de un león: "Rugido"
     */
    public String getSonido() {
        return "Rugido";
    }

    /**
     * Método que devuelve un String con los alimentos de un león

```

```

    * @return Un valor String con la alimentación de un león: "Carnívoro"
    */
    public String getAlimentos() {
        return "Carnívoro";
    }

    /**
     * Método que devuelve un String con el hábitat de un león
     * @return Un valor String con el hábitat de un león: "Praderas"
     */

    public String getHaabitat() {
        return "Praderas";
    }

    /**
     * Método que devuelve un String con el nombre científico de un león
     * @return Un valor String con el nombre científico de un león:
     * "Panthera leo"
     */
    public String getNombreCientifico() {
        return "Panthera leo";
    }
}

```

Clase Gato:

```

/**
 * Esta clase concreta denominada Gato es una subclase de Felino.
 *
 * @version 1.2/2020
 */
public class Gato extends Felino {
    /**
     * Método que devuelve un String con el sonido de un gato
     * @return Un valor String con el sonido de un gato: "Mauellido"
     */
    public String getSonido() {
        return "Mauellido";
    }

    /**
     * Método que devuelve un String con los alimentos de un gato
     * @return Un valor String con la alimentación de un gato: "Ratones"
     */
    public String getAlimentos() {
        return "Ratones";
    }

    /**
     * Método que devuelve un String con el hábitat de un gato
     * @return Un valor String con el hábitat de un gato: "Doméstico"
     */
}

```

```

    */
    public String getHaabitat() {
        return "Doméstico";
    }

    /**
     * Método que devuelve un String con el nombre científico de un gato
     * @return Un valor String con el nombre científico de un gato:
     * "Felis silvestris catus"
     */
    public String getNombreCientifico() {
        return "Felis silvestris catus";
    }
}

```

Clase Prueba:

```

/**
 * Esta clase prueba diferentes animales y sus métodos.
 *
 * @version 1.2/2020
 */
public class Prueba {
    /**
     * Método main que crea un array de varios animales y para cada uno
     * muestra en pantalla su nombre científico, su sonido, alimentos y
     * hábitat
     */
    public static void main(String[] args) {

        Animal[] animales = new Animal[4]; /* Define un array de cuatro
elementos de tipo Animal */
        animales[0] = new Gato();
        animales[1] = new Perro();
        animales[2] = new Lobo();
        animales[3] = new Leon();

        for (int i = 0; i < animales.length; i++) {

            /* Recorre el array de animales */
            System.out.println(animales[i].getNombreCientifico());
            System.out.println("Sonido: " + animales[i].getSonido());
            System.out.println("Alimentos: " + animales[i].getAlimentos());

            System.out.println("Hábitat: " + animales[i].getHaabitat());
            System.out.println();
        }
    }
}

```

Prueba de Ejecución del programa:

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS ... powershell + - [] [X] ... ^ X

PS C:\GONZALES\SIS103\Clases_Abstractas> javac Prueba.java Canido.java Perro.java Lobo.j
ava Felino.java Leon.java Gato.java
PS C:\GONZALES\SIS103\Clases_Abstractas> java Prueba
Felis silvestris catus
Sonido: Maullido
Alimentos: Ratones
Hábitat: Doméstico

Canis lupus familiaris
Sonido: Ladrido
Alimentos: Carnívoro
Hábitat: Doméstico

Canis lupus
Sonido: Aullido
Alimentos: Carnívoro
Hábitat: Bosque

Panthera leo
Sonido: Rugido
Alimentos: Carnívoro
Hábitat: Praderas

PS C:\GONZALES\SIS103\Clases_Abstractas> |
```

Ejercicios Propuestos:

Clase Numerica:

```
public abstract class Numerica {

    // Convierte el número a String
    public abstract String toString();

    // Compara si dos objetos son iguales, con el parámetro
    public abstract boolean equals(Object obj);

    // Retorna la suma de dos números
    public abstract Numerica sumar(Numerica numero);

    // Retorna la resta de dos números
    public abstract Numerica restar(Numerica numero);

    // Retorna el producto de dos números
    public abstract Numerica multiplicar(Numerica numero);

    // Retorna la división de dos números
    public abstract Numerica dividir(Numerica numero);
}
```

Clase Fraccion:

```
public class Fraccion extends Numerica {

    private int numerador;
    private int denominador;

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

    @Override
    public String toString() {
        return numerador + "/" + denominador;
    }

    @Override
    public boolean equals(Object objeto) {
        if (this == objeto) {
            return true;
        }
        if (objeto == null || getClass() != objeto.getClass()) {
            return false;
        }
        Fraccion fraccion = (Fraccion) objeto;
        return numerador == fraccion.numerador && denominador ==
fraccion.denominador;
    }

    @Override
    public Numerica sumar(Numerica numero) {

        if (numero instanceof Fraccion){
            Fraccion fraccion = (Fraccion) numero;
            int numerador = this.numerador * fraccion.denominador +
this.denominador * fraccion.numerador;
            int denominador = this.denominador * fraccion.denominador;
            return new Fraccion(numerador, denominador);
        } else {
            return null;
        }
    }

    @Override
    public Numerica restar(Numerica numero) {

        if (numero instanceof Fraccion){
            Fraccion fraccion = (Fraccion) numero;
            int numerador = this.numerador * fraccion.denominador -
this.denominador * fraccion.numerador;
            int denominador = this.denominador * fraccion.denominador;
            return new Fraccion(numerador, denominador);
        } else {
            return null;
        }
    }
}
```



```

        return null;
    }
}

@Override
public Numerica multiplicar(Numerica numero) {

    if (numero instanceof Fraccion){
        Fraccion fraccion = (Fraccion) numero;
        int numerador = this.numerador * fraccion.numerador;
        int denominador = this.denominador * fraccion.denominador;
        return new Fraccion(numerador, denominador);
    } else {
        return null;
    }
}

@Override
public Numerica dividir(Numerica numero) {

    if (numero instanceof Fraccion){
        Fraccion fraccion = (Fraccion) numero;
        int numerador = this.numerador * fraccion.denominador;
        int denominador = this.denominador * fraccion.numerador;
        return new Fraccion(numerador, denominador);
    } else {
        return null;
    }
}
}

```

Clase PruebaFraccion:

```

public class PruebaFraccion {

    public static void main(String[] args) {

        Fraccion fraccion1 = new Fraccion(3, 7);
        Fraccion fraccion2 = new Fraccion(4, 5);

        System.out.println("Fracción 1: " + fraccion1);
        System.out.println("Fracción 2: " + fraccion2);

        System.out.println("Suma: " + fraccion1.sumar(fraccion2));
        System.out.println("Resta: " + fraccion1.restar(fraccion2));
        System.out.println("Multiplicación: " +
fraccion1.multiplicar(fraccion2));
        System.out.println("División: " + fraccion1.dividir(fraccion2));

        if (fraccion1.equals(fraccion2)) {
            System.out.println("Las fracciones son iguales");
        }
    }
}

```

```

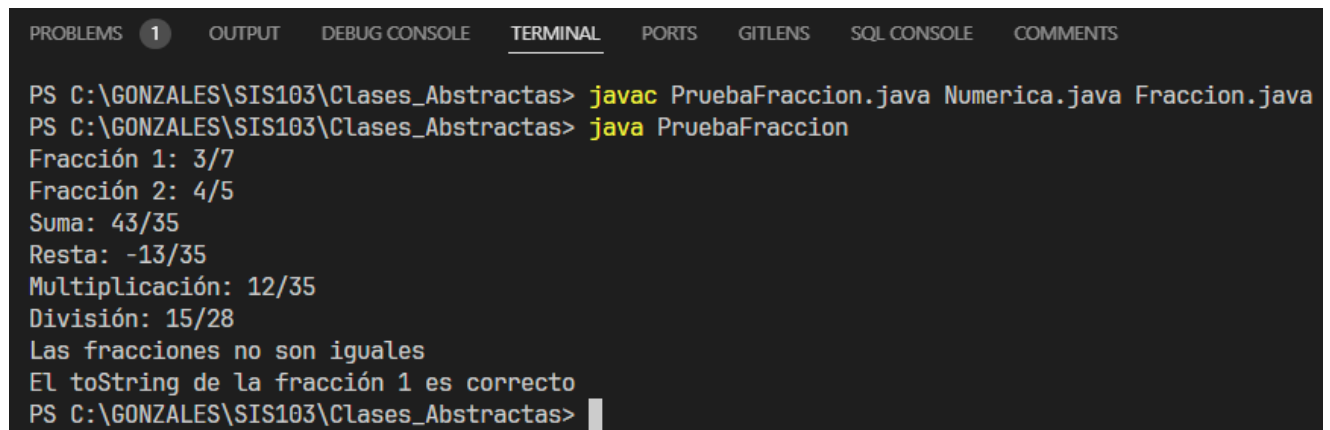
    } else {
        System.out.println("Las fracciones no son iguales");
    }

    String toString = fraccion1.toString();

    if (toString.equals("3/7")) {
        System.out.println("El toString de la fracción 1 es correcto");
    } else {
        System.out.println("El toString de la fracción 1 no es correcto");
    }
}
}

```

Prueba de ejecución del programa:



```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SQL CONSOLE COMMENTS
PS C:\GONZALES\SIS103\Clases_Abstractas> javac PruebaFraccion.java Numerica.java Fraccion.java
PS C:\GONZALES\SIS103\Clases_Abstractas> java PruebaFraccion
Fracción 1: 3/7
Fracción 2: 4/5
Suma: 43/35
Resta: -13/35
Multiplicación: 12/35
División: 15/28
Las fracciones no son iguales
El toString de la fracción 1 es correcto
PS C:\GONZALES\SIS103\Clases_Abstractas>

```