TP N°6: Interfaces y Excepciones

La siguiente guía cubre los contenidos vistos en la clase teórica **6**

Ejercicio 1

Indicar si los siguientes fragmentos de código compilan o no compilan, y en los casos que sí compila, indicar la salida obtenida. Justificar.

1.1

```
package ar.edu.itba.poo.tp6.ej1.sub1;

public interface A {

   boolean even(int value);

}
```

```
package ar.edu.itba.poo.tp6.ej1.sub1;

public interface B {

   boolean even(int value);

}
```

```
package ar.edu.itba.poo.tp6.ej1.sub1;

public class C implements A, B {

   @Override
   public boolean even(int value) {
       return value % 2 == 0;
   }

}
```

Si el numero es par o no

1.2

```
package ar.edu.itba.poo.tp6.ej1.sub2;

public interface Greeting {

   String initialGreeting();

}
```

```
package ar.edu.itba.poo.tp6.ej1.sub2;

public abstract class GreetingImpl implements Greeting {

   public String finalGreeting();

}
```

Tira error porque le falta el codigo al metodo, o porque no esta declarado como abstracto

1.3

```
package ar.edu.itba.poo.tp6.ej1.sub3;

public interface Greeting {

   String initialGreeting();

}
```

```
package ar.edu.itba.poo.tp6.ej1.sub3;

public class GreetingImpl implements Greeting {

   @Override
   public String initialGreeting() {
       return "Hola";
   }

}
```

```
package ar.edu.itba.poo.tp6.ej1.sub3;

public class GreetingTester {

   public static void main(String[] args) {
       Greeting greeting = new GreetingImpl();
       if (greeting instanceof Greeting) {
           Greeting var = (Greeting) greeting;
           System.out.println(var.initialGreeting());
       }
       if (greeting instanceof GreetingImpl) {
           GreetingImpl var = (GreetingImpl) greeting;
           System.out.println(var.initialGreeting());
       }
   }

}
```

```
Hola
Hola
```

1.4

```
package ar.edu.itba.poo.tp6.ej1.sub4;

public interface A {

   int sum(int num1, int num2);

}
```

```
package ar.edu.itba.poo.tp6.ej1.sub4;

public interface B extends A {

   double sum(double num1, double num2);
```

```
}
```

```
package ar.edu.itba.poo.tp6.ej1.sub4;

public class C implements B {

   @Override
   public double sum(double num1, double num2) {
       return num1 + num2;
   }

}
```

Tira error porque falta implementar el de la clase A

~~*Ejercicio 2*~~

Implementar en Java la familia de clases que modelan los nodos HTML del ejercicio ***Ejercicio 1*** del ***TP Nº3*** y realizar el diagrama de clases correspondiente.

Ahora, el módulo de Ruby `HTMLText` es la interfaz `HTMLText`:

```
package ar.edu.itba.poo.tp6.html;

public interface HTMLText {

   String source();

}
```

¿Por qué no se puede incluir en la interfaz un método `default` con la implementación del método `toString()`?

El programa de prueba es el siguiente:

```
package ar.edu.itba.poo.tp6.html;

public class HTMLTester {

   public static void main(String[] args) {
       PlainText text = new PlainText("Hola");
       HTMLText boldText = new BoldText(text);
       HTMLText italicText = new ItalicText(text);
       System.out.println(boldText);
       System.out.println(italicText);
       HTMLText boldItalicText = new BoldText(italicText);
       System.out.println(boldItalicText);
       text.setText("ITBA");
       System.out.println(boldText);
       System.out.println(italicText);
       System.out.println(boldItalicText);
       HTMLText linkText = new LinkText(text, "itba.edu.ar");
       HTMLText linkBoldText1 = new LinkText(boldItalicText, "itba.edu.ar");
       HTMLText linkBoldText2 = new BoldText(linkText);
       System.out.println(linkText);
       System.out.println(linkBoldText1);
       System.out.println(linkBoldText2);
       text.setText("Ejemplo");
```

```
        System.out.println(linkBoldText1);
        System.out.println(linkBoldText2);
    }

}
```

~~***Ejercicio 3***~~

Implementar en Java la familia de clases que modelan las funciones matemáticas del ejercicio ***Ejercicio 3*** del ***TP Nº3*** y realizar el diagrama de clases correspondiente.

Ahora, el módulo de Ruby `Function` es la interfaz `Function`:

```
package ar.edu.itba.poo.tp6.function;

public interface Function {

    double evaluate(double x);

}
```

El programa de prueba es el siguiente:

```
package ar.edu.itba.poo.tp6.function;

public class FunctionTester {

    public static void main(String[] args) {
        Function f1 = new LinearFunction(2, 0); // y = 2x
        Function f2 = new QuadraticFunction(1, 0, 0); // y = x^2
        Function f3 = new CompositeFunction(f1, f2); // y = (2x)^2
        System.out.println(f3.evaluate(1)); // 4.0
        System.out.println(f3.evaluate(2)); // 16.0
        Function f4 = new SineFunction(); //y = sin(x)
        Function f5 = new CompositeFunction(f1, f4); // y = sin(2x)
        Function f6 = new CompositeFunction(f5, f1); // y = 2 sin(2x)
        System.out.println(f6.evaluate(0)); // 0.0
        System.out.println(f6.evaluate(Math.PI / 4.0)); // 2.0
    }

}
```

Ejercicio 4

Implementar la clase `Interval`, que permite representar un conjunto de valores en un intervalo dado. Se debe ofrecer como mínimo el siguiente comportamiento:

- `Interval(double start, double end, double increment)` Crea un intervalo con todos los valores entre start y end, separados entre sí por una distancia de increment.
- `Interval(double start, double end)` Crea un intervalo con todos los valores entre start y end, en donde el incremento es 1.
- `long size()` Devuelve la cantidad de elementos que posee el intervalo.
- `double at(long index)` Devuelve el elemento que ocupa el lugar 1, 2, etc.

- `long indexOf(double valor)` Devuelve el lugar que ocupa el valor en el intervalo (1, 2, etc) o 0 si no pertenece al mismo.
- `boolean includes(double valor)` Devuelve true si el valor pertenece al intervalo y false en caso contrario.
- `String toString()`
- `boolean equals(Object other)`
- `int hashCode()`

En los casos en donde los parámetros podrían ser inválidos (por ejemplo si en el método `at` el índice es inválido, o si en el constructor se especifica un incremento 0) validarlos, y lanzar una excepción de tipo `IllegalArgumentException`. ¿Es necesario agregar la cláusula `throws IllegalArgumentException` a estos métodos?

Escribir un programa de prueba para testear los métodos de la clase `Interval`.

Ejercicio 5

Añadir a la implementación anterior el método `count`, que cuente la cantidad de elementos dentro del intervalo que cumplen con una determinada condición especificada por el usuario. La idea consiste en modelar la condición a través de una interfaz con un único método, que se utilizará para evaluar cada elemento del intervalo. Diseñar esta interfaz `IntervalCondition` e implementar el método `count`.

Modificar el programa de prueba de forma que la invocación al método `count` reciba una clase anónima que implemente la interfaz `IntervalCondition`.

Ejercicio 6

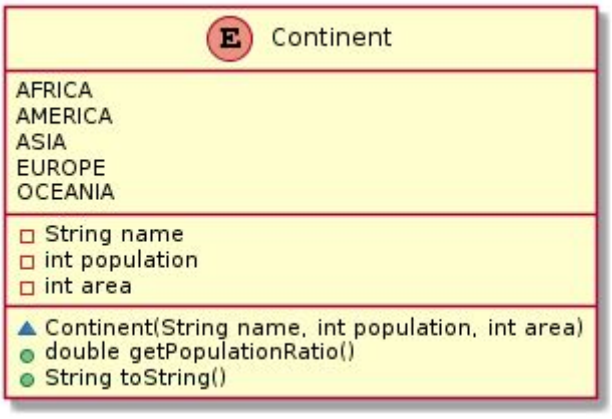
Implementar en Java la familia de clases que modelan las expresiones de verdad del ***Ejercicio 4*** del ***TP Nº3***.

El módulo `Expression` implementado en Ruby ¿corresponde que sea una clase abstracta o una interfaz en Java?

¿Cómo modificaría la relación entre `AndExpression` y `OrExpression` para ahorrar unas líneas de código? ¿Qué se puede hacer en Java que Ruby no permitía?

Ejercicio 7

Se cuenta con el siguiente enumerativo `Continent` que modela a los continentes:



```java
package ar.edu.itba.poo.tp6.continents;

public enum Continent {

   AFRICA("África", 1100, 30),
   AMERICA ("América", 990, 42),
   ASIA("Asia", 4400, 43),
   EUROPE("Europa", 730, 10),
   OCEANIA("Oceanía", 39, 9);

   private String name;
   private int population;
   private int area;

   Continent(String name, int population, int area) {
       this.name = name;
       this.population = population;
       this.area = area;
   }

   public double getPopulationRatio() {
       return (double) population / area;
   }

   @Override
   public String toString() {
       return name;
   }

}
```

Investigar en la documentación de Java los métodos de clase de los enumerativos y completar el siguiente programa de prueba

```java
package ar.edu.itba.poo.tp6.continents;

public class ContinentTester {

   public static void main(String[] args) {
       System.out.println("Densidades de población:");
       for(Continent continent : Continent.............) {
           System.out.println(String.format("%s = %.2f", continent,
continent.getPopulationRatio()));
       }
       System.out.printf("%.2f",
Continent..........("AMERICA").getPopulationRatio());
   }

}
```

para que imprima la siguiente salida

```
Densidades de población:
África = 36.67
América = 23.57
Asia = 102.33
```

```
Europa = 73.00
Oceanía = 4.33
23,57
```

Ejercicio 8

Implementar el enumerativo `BasicOperation` que modele las cuatro operaciones básicas: suma, resta, multiplicación y división.

Implementar además el enumerativo `ExtendedOperation` que modele las operaciones de potencia y módulo.

¿Cómo se puede modelar el comportamiento en común que tienen ambos enumerativos?

El siguiente programa de prueba:

```
package ar.edu.itba.poo.tp6.operations;

public class OperationTester {

   public static void main(String[] args) {
       double x = 4;
       double y = 2;
       for(BasicOperation operation : BasicOperation.values()) {
           System.out.printf("%.2f %s %.2f = %.2f\n", x, operation, y,
operation.apply(x,y));
       }
       for(ExtendedOperation operation : ExtendedOperation.values()) {
           System.out.printf("%.2f %s %.2f = %.2f\n", x, operation, y,
operation.apply(x,y));
       }
   }

}
```

imprime la siguiente salida:

```
4,00 + 2,00 = 6,00
4,00 - 2,00 = 2,00
4,00 * 2,00 = 8,00
4,00 / 2,00 = 2,00
4,00 ^ 2,00 = 16,00
4,00 % 2,00 = 0,00
```

Ejercicio 9

Se cuenta con la siguiente interfaz para representar una lista lineal simplemente encadenada:

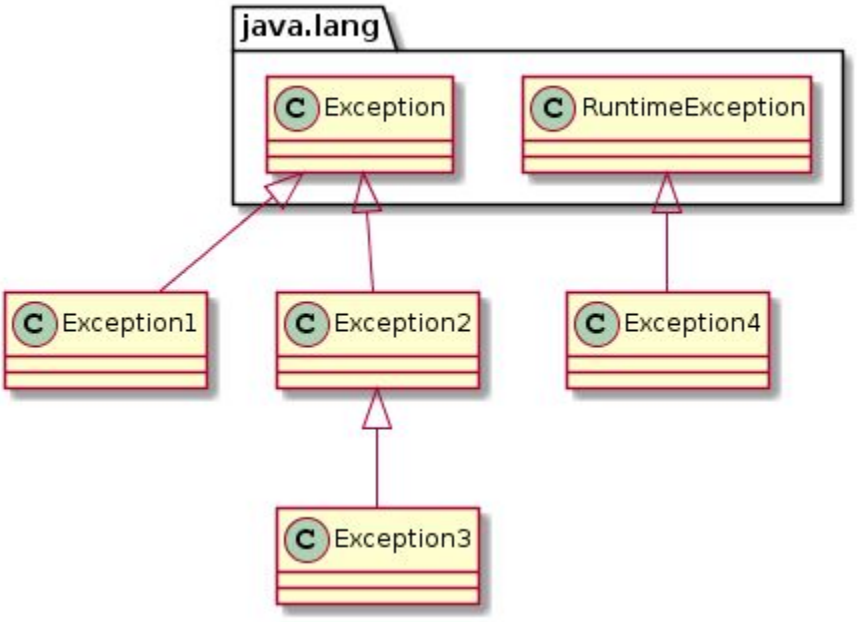
```
package ar.edu.itba.poo.tp6.list;

public interface LinearList {

   /**
    * Agrega un elemento al final de la lista.
    */
   void add(Object obj);
```

```
    /**
     * Obtiene el i-ésimo elemento de la lista.
     */
    Object get(int i);

    /**
     * Modifica el i-ésimo elemento de la lista colocando un nuevo valor.
     */
    void set(int i, Object obj);

    /**
     * Elimina el i-ésimo elemento de la lista.
     */
    void remove(int i);

    /**
     * Busca el índice de la primer ocurrencia de un objeto en la lista.
     */
    int indexOf(Object obj);

    /**
     * Retorna el tamaño de la lista.
     */
    int size();

}
```

Escribir un programa de testeo que use esta interfaz. Luego realizar una implementación de la misma y utilizar dicha implementación en el programa anterior para verificar su correcto funcionamiento.

Ejercicio 10

Dada la siguiente jerarquía de excepciones:



indicar para cada uno de los siguientes programas si compilan o no. En el caso de que compilen, indicar la salida obtenida. En caso contrario explicar el motivo.

```
package ar.edu.itba.poo.tp6.exceptions;

public class Ej1 {

   public static void main(String[] args) {
       Ej1 ej1 = new Ej1();
       try {
           ej1.method();
           System.out.println("Método ejecutado");
       } catch (Exception2 e) {
           System.out.println("Excepción 2 capturada");
       } finally {
           System.out.println("Finalizando");
       }
   }

   public void method() throws Exception3 {
       throw new Exception3();
   }

}
```

```
package ar.edu.itba.poo.tp6.exceptions;

public class Ej2 {

   public static void main(String[] args) {
       Ej2 ej2 = new Ej2();
       try {
           try {
               ej2.m3();
           } catch (Exception3 e) {
               System.out.println("Excepción 3 capturada");
           } finally {
               System.out.println("Finalizando 3");
           }
       } catch (Exception2 e) {
           System.out.println("Excepción 2 capturada");
       } finally {
           System.out.println("Finalizando 2");
       }
       try {
           ej2.m1();
       } catch (Exception4 e) {
           System.out.println("Excepción 4 capturada");
       }
   }

   public void m1() {
       throw new Exception4();
   }

   public void m2() throws Exception4 {
```

```
        throw new Exception4();
    }

    public void m3() throws Exception2 {
        throw new Exception3();
    }

}
```

```
package ar.edu.itba.poo.tp6.exceptions;

public class Ej3 {

    public static void main(String[] args) {
        Ej3 ej3 = new Ej3();
        try {
            ej3.method();
        } catch (Exception2 e) {
            System.out.println("Excepción 2 capturada");
        } catch (Exception3 e) {
            System.out.println("Excepción 3 capturada");
        }
    }

    public void method() throws Exception3 {
        throw new Exception3();
    }

}
```

```
package ar.edu.itba.poo.tp6.exceptions;

public class Ej4 {

    public static void main(String[] args) {
        throw new Exception2();
    }

}
```

```
package ar.edu.itba.poo.tp6.exceptions;

public class Ej5 {

    public static void main(String[] args) {
        throw new Exception4();
    }

}
```

```
package ar.edu.itba.poo.tp6.exceptions;

public class Ej6 {
```

```
    public static void main(String[] args) {
        try {
            System.out.println("Dentro del bloque try");
        } catch (Exception4 e) {
            System.out.println("Dentro del bloque catch");
        }
    }

}
```

```
package ar.edu.itba.poo.tp6.exceptions;

public class Ej7 {

    public static void main(String[] args) {
        try {
            System.out.println("Dentro del bloque try");
        } catch (Exception2 e) {
            System.out.println("Dentro del bloque catch");
        }
    }

}
```

```
package ar.edu.itba.poo.tp6.exceptions;

public class Ej8 {

    public static void main(String[] args) {
        Ej8 ej8 = new Ej8();
        try {
            try {
                ej8.method();
            } catch (Exception3 e) {
                ej8.method();
                System.out.println("Excepción 3 capturada");
            } finally {
                System.out.println("Finalizando 3");
            }

        } catch (Exception2 e) {
            System.out.println("Excepción 2 capturada");
        } finally {
            System.out.println("Finalizando 2");
        }
    }

    public void method() throws Exception2 {
        throw new Exception3();
    }

}
```