Primer Parcial de Programación Orientada a Objetos (72.33)

15/04/2020

Ejercicio 1

Dada la siguiente jerarquía de clases, con los métodos de instancia indicados para cada una, se cuenta con tres instancias homónimas a la clase a la cual pertenecen.

Completar el cuadro de doble entrada (clase y mensaje) indicando qué se obtiene al enviar cada uno de los mensajes a instancias de cada una de las clases.

	m1	m2	m3
A			3
B			
C			

```
class A {
   int m1() {
       return
this.m3();
   }
   int m2() {
       return m3();
   }
   int m3() {
       return 3;
   }
}
```

```
class B extends A {
   int m1() {
       return
super.m2();
   }
   int m2() {
       return 2;
   }
   int m3() {
       return m2();
   }
}
```

```
class C extends B {
   int m1() {
       return 1;
   }
   int m2() {
       return m2();
   }
   int m3() {
       return super.m3();
   }
}
```

Escribir en el cuadro texto, en un renglón por cada método. Por ejemplo, para el ejemplo ya provisto en la tabla, deberá escribir:

> *a.m3() = 3*

En caso de que se obtenga error indicarlo y escribir una breve descripción del error.

Ejercicio 2

Se cuenta con la siguiente interfaz **`MathFunction`**:

```
@FunctionalInterface
public interface MathFunction < T extends Comparable < ? super T > > {

   T evaluate(T x);

}
```

Se desea modelar un **intervalo genérico** a partir de un **inicio**, un **fin** y una función a aplicar en cada **paso**. **Implementar la clase `MathFunctionInterval` y todo lo necesario para que con el siguiente programa de prueba se obtenga la salida indicada en los comentarios:**

```java
public class MathFunctionIntervalTester {

   public static void main(String[] args) {
       // Ejemplo 1: Intervalo de enteros (Integer)
       // De 1 a 10 con paso 2
       MathFunction < Integer > stepTwo = new MathFunction < Integer > () {
           @Override
           public Integer evaluate(Integer x) {
               return x + 2;
           }
       };
       MathFunctionInterval < Integer > integerInterval =
               new MathFunctionInterval < > (1, 10, stepTwo);
       for(int element : integerInterval) { // Imprime 1 3 5 7 9
           System.out.println(element);
       }

       // Ejemplo 2: Intervalo de números reales (Double)
       // De 1 a 10 con paso 2.5
       MathFunctionInterval < Double > doubleInterval =
               new MathFunctionInterval < > (1.0, 10.0, new MathFunction < Double > () {
                   @Override
                   public Double evaluate(Double x) {
                       return x + 2.5;
                   }
               });
       for(double element : doubleInterval) { // Imprime 1.0 3.5 6.0 8.5
           System.out.println(element);
       }

       // Ejemplo 3: Intervalo de FloatWrapper
       // De 1 a 10 con paso 2.5
       MathFunctionInterval < FloatWrapper > floatWrapperInterval =
               new MathFunctionInterval < > (new FloatWrapper(1f), new FloatWrapper(10f),
                       new MathFunction < FloatWrapper >() {
                   @Override
                   public FloatWrapper evaluate(FloatWrapper x) {
                       return new FloatWrapper(x.f + 2.5f);
                   }
               });
       for(FloatWrapper element : floatWrapperInterval) { // Imprime 1.0 3.5 6.0 8.5
           System.out.println(element.f);
       }

       // Ejemplo 4: Sólo debe funcionar con start < end
       try {
           new MathFunctionInterval < > (10.0, 1.0, x -> x - 2.5);
       } catch (Exception ex) { // Imprime Start no es menor que end
           System.out.println(ex.getMessage());
       }

       // Ejemplo 5: Intervalo incorrecto donde no converge
       // De 1 a 10 con paso -1 generará un intervalo infinito
       // y no es necesario validarlo
       MathFunctionInterval < Integer > wrongInterval =
           new MathFunctionInterval < > (1, 10, x -> x - 1);
       for(int element : wrongInterval) { // Imprime 1 0 -1 -2 -3 ... y no termina
```

```
            System.out.println(element);
        }
    }

    static class FloatWrapper implements Comparable < FloatWrapper > {

        public float f;

        public FloatWrapper(float f) {
            this.f = f;
        }

        @Override
        public int compareTo(FloatWrapper o) {
            return Float.compare(f, o.f);
        }

    }

}
```

No se puede modificar la interfaz `MathFunction` ni la clase `MathFunctionIntervalTester`.

Ejercicio 3

Se cuenta con la siguiente clase abstracta `Element` que modela un elemento:

```
public abstract class Element {

    public abstract String getContents();

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

}
```

Implementar todo lo necesario, pudiéndose agregar métodos a la clase `Element`, para que con el siguiente programa de prueba:

```
public class ElementTester {

    public static void main(String[] args) {
        TextElement element1 = new TextElement("mundo");
        System.out.println(element1);
        System.out.println("##########");

        TextElement element2 = new TextElement("hola");
        System.out.println(element2);
        System.out.println("++++++++++");

        Element element21 = element2.above(element1);
```

```
        System.out.println(element21);
        System.out.println("----------");

        element1.setText("world");
        System.out.println(element21);
        System.out.println("**********");

        TextElement element3 = new TextElement("start");
        Element element321 = element21.below(element3);
        System.out.println(element321);
        System.out.println(".........");

        element2.setText("hello");
        System.out.println(element321);
        System.out.println("//////////");

        Element element111 = element1.repeat(3);
        System.out.println(element111);
        System.out.println("]]]]]]]]]]");

        Element element2121 = element21.repeat(2);
        System.out.println(element2121);
    }

}
```

se obtenga la siguiente salida:

```
mundo
##########
hola
++++++++++
hola
mundo
----------
hola
world
**********
start
hola
world
.........
start
hello
world
//////////
world
world
world
]]]]]]]]]]
hello
world
hello
world
```

No se puede modificar el programa de prueba.