Universidad de los Andes
Ingeniería de Sistemas y Computación

ISIS1205 – Algorítmica y Programación 2

Hoja de trabajo teórica Nivel 7

Cupi2

Problema:
La dificultad de una figura del juego se redefine como el valor de su número de fichas. Una figura de 5×4 es de mayor dificultad que una figura de 6×3.

En la clase **Figura** se redefinieron el atributo de la dificultad, el método constructor y el método analizador del atributo. En la definición de la clase **Figura** hay que hacer las siguientes modificaciones:

- [15%] Rescribir el método de comparación por dificultad (incluye su signatura o encabezado).
- [15%] Reescribir las validaciones de la dificultad en el método del invariante de la clase.

```
public class Figura
{

        /**
        * Dificultad para armar la figura
        */
        private int dificultad;
        /**
        * retorna el valor de la dificultad
        */
        public int obtenerDificultad( )
        …
        /**
        * compara dos figuras por su dificultad.
        * @param f La figura con la que se está comparando - f != null
        * @return Retorna 0 si las figuras son de la misma dificultad.
        * Retorna un valor positivo si la dificultad del objeto es mayor a la de la figura f.
        * Retorna un valor negativo si la dificultad del objeto es menor a la de la figura f.
        */
```

```
        /**
        * Verifica el invariante de la clase.
        */
        private void validarInvariante( )
        {
        …
        // Realizar la validación de la dificultad




        }
}
```

En la definición de la clase **JuegoRompecabezas** hay que hacer las siguientes modificaciones:

- [25%] Ordenar el conjunto de figuras descendentemente por dificultad.
- [25%] Definir una lista con las figuras que tienen un rango de dificultad conocido.

```
public class JuegoRompecabezas
{
        /**
        * Ordena las figuras descendentemente (de mayor a menor) por dificultad utilizando
        * el método de ordenamiento: _____
        */
        public void ordenarFigurasPorDificultadDescendentemente( )
        {
```

```
    }

    /**
    * Define una lista con las figuras que tienen un rango de dificultad conocido.
    * Precondición: Las figuras No se garantiza que estén ordenadas descendentemente por dificultad.
    * @param difMin Valor mínimo de dificultad buscada. 1 ≤ difMin
    * @param difMax Valor máximo de dificultad buscada. 1 ≤ difMin ≤ difMax
    * @return lista con las figuras que tienen una dificultad en el rango buscado.
    */
    ArrayList buscarPorDificultad( int difMin, int difMax )
    {

    }
}
```

[25%] En la clase **JuegoRompecabezasTest** hay que adicionar una prueba automática para validar el resultado del método de ordenamiento descendente por dificultad usando el escenario #2.

```
public class JuegoRompecabezasTest extends TestCase
{
        private JuegoRompecabezas juego;   // juego donde se haran las pruebas
        private int cantidadFiguras;

        private void setupEscenario2( )
        {
                juego = new JuegoRompecabezas( );
                Figura f1 = new Figura( "nombre1", 3, 4, "a", "ruta1", new Ficha[3][4] ); // dificultad 3 × 4
                Figura f2 = new Figura( "nombre2", 2, 2, "b", "ruta2", new Ficha[2][2] ); // dificultad 2 × 2
                Figura f3 = new Figura( "nombre3", 6, 3, "c", "ruta3", new Ficha[6][3] ); // dificultad 6 × 3
                juego.agregarFigura( f2 );
                juego.agregarFigura( f3 );
                juego.agregarFigura( f1 );
                cantidadFiguras = 3;
        }


        /*
        * Verifica el método de ordenamiento de las figuras por dificultad en orden descendente.
        *Utiliza el escenario2.
        * Se verifica el posicionamiento correcto de las figuras.
        */
        public void testOrdenarFigurasPorDificultadDescendentemente ( )
        {









                }
}
```