Rompecabezas

Objetivos

Con el presente ejercicio el estudiante:

- Repasará algunos de los conceptos vistos en el curso anterior (Mundo e Interfaz).
- Desarrollará una aplicación siguiendo un proceso incremental.
- Construirá los invariantes de las clases del mundo del ejercicio.
- Utilizará la instrucción *assert* de Java para verificar invariantes.
- Desarrollará pruebas unitarias en junit para las clases del ejercicio.
- Entenderá y aplicará el concepto de comparación.
- Entenderá y desarrollará tres algoritmos de ordenamiento (burbuja, inserción y selección).
- Entenderá y desarrollará algoritmos de búsqueda binaria y secuencial sobre una lista ordenada o no ordenada.
- Aprenderá a dar la información de un objeto como un texto con el método toString().
- Utilizará JList y JScrollPane para presentar listas en la interfaz gráfica.

Preparación

1. Localice el archivo n7_demo.zip, decomprímalo y ejecute el programa .exe que muestra una ejecución del programa. Estudie el funcionamiento esperado del programa.
2. Localice y descomprima el archivo esqueleto.zip.
3. Cree el proyecto en eclipse llamado n7_rompecabezas con el contenido.
4. Lea el enunciado del problema disponible en n7_rompecabezas/docs/specs/Descripcion.doc. Lea también el documento RequerimientosFuncionales.doc en el mismo directorio.
5. Asegúrese de tener activado el uso de aserciones para la ejecución del programa. Ver el tutorial en http://cupi2.uniandes.edu.co/cursos/apo2/docs/n7_assert.pdf

Parte1: Construcción de invariantes

1. Revise el modelo del mundo en n7_rompecabezas/docs/specs/Modelo.mdl o ModeloConceptual.JPG e identifique las clases del mundo.

2. Complete la clase **Figura**

    a. Defina el invariante de esta clase y documéntelo en la cabecera de la clase, siguiendo las normas explicadas para ello.

    b. Cree en la clase **Figura** el método `private void verificarInvariante()`. Utilice aserciones para validar el invariante que definió en el punto anterior.

c. Utilice el método para verificar el invariante en todos aquellos métodos de la clase que modifican el estado. ¿Dónde debe ir situado el llamado?

3. Revise la clase **JuegoRompecabezas**

a. Defina el invariante de esta clase y documéntelo en la cabecera de la clase, siguiendo las normas explicadas para ello.

b. Cree en la clase **JuegoRompecabezas** el método `private void verificarInvariante()`. Utilice aserciones para validar el invariante que definió en el punto anterior.

c. Utilice el método para verificar el invariante en todos aquellos métodos de la clase que modifican el estado.

Los siguientes pasos conforman el plan sugerido para desarrollar el ejercicio. La idea central es ir desarrollando y probando incrementalmente los métodos de las clases.

Ud. debe verificar que todos los métodos de todas las clases queden definidos correctamente.

Parte2: Comparaciones, ordenamiento y búsqueda

1. Verifique el funcionamiento correcto de la clase Ficha.java, mediante la ejecución de las pruebas en la clase **FichaTest**
2. Complete la clase **Figura**:
   - Complete los tres métodos de comparación, utilizando el método compareTo de la clase String y la instrucción condicional if-else:
     - `public int compararPorDificultad ( Figura f )`
     - `public int compararPorCategoria ( Figura f )`
     - `public int compararPorNombre ( Figura f )`
   - Defina y documente la manera como debe probarse cada uno de los métodos: Objetivo de la prueba, estado inicial, operación que se realiza, estado final esperado. Utilice el formato propuesto en las notas de clase para especificar los casos de prueba, en el archivo CasosPrueba.doc en el directorio docs/specs del proyecto.
   - Implemente en la clase de pruebas **FiguraTest** los métodos de prueba para verificar el correcto funcionamiento de los métodos desarrollados en el punto anterior. Utilice el método setupEscenario1 () para definir el estado inicial de la prueba y el resultado esperado. Estos son:
     - `public void testCompararPorDificultad ( )`

- `public void testCompararPorCategoria ( )`
- `public void testCompararPorNombre ( )`

- Verifique que al ejecutar la clase de prueba **FiguraTest** los casos de prueba se ejecutan sin errores.

3. Complete la clase **JuegoRompecabezas** con ordenamientos:
   - Complete el método `public void ordenarFigurasPorCategoria( )` que ordena los Figuras alfabéticamente según su categoría, utilizando el algoritmo de **selección**.
   - Complete el método `public void ordenarFigurasPorDificultad( )` que ordena los Figuras alfabéticamente por su dificultad utilizando el algoritmo de **burbuja**.
   - Complete el método `public void ordenarFigurasPorNombre( )` que ordena los Figuras alfabéticamente por nombre, utilizando el algoritmo de **inserción**.
   - Defina y documente la manera como debe probarse cada uno de los métodos: Objetivo de la prueba, estado inicial, operación que se realiza, estado final esperado. Utilice el formato propuesto en las notas de clase para especificar los casos de prueba, en el archivo CasosPrueba.doc en el directorio docs/specs del proyecto.
   - Implemente en la clase de pruebas **JuegoRompecabezasTest** los métodos de prueba para verificar el correcto funcionamiento de los métodos de ordenamiento desarrollados anteriormente. Utilice el método setupEscenario3 () para definir el estado inicial de la prueba y el resultado esperado. Estos son:
     - `public void testOrdenarPorCategoria ( )`
     - `public void testOrdenarPorDificultad ( )`
     - `public void testOrdenarPornombre ( )`
   - Verifique que al ejecutar la clase de prueba **JuegoRompecabezasTest** los casos de prueba se ejecutan sin errores.

4. Complete la clase **JuegoRompecabezas** con búsqueda:
   - Implemente el método `public int buscarFigura( String nombre )` que busca un Figura a partir de su nombre.
   - Implemente el método `public int buscarBinarioPorNombre( String nombre )` que busca un figura a partir de su nombre utilizando una búsqueda binaria. Recuerde que la precondición de la búsqueda binaria es que el conjunto en donde se busca esté ordenado.
   - Implemente en la clase de pruebas **JuegoRompecabezasTest** los métodos de prueba para verificar el correcto funcionamiento de los métodos de búsqueda desarrollados en anteriormente. Utilice el método setupEscenario1 () para definir el estado inicial de la prueba y el resultado esperado. Estos son:
     - `public void testBuscarFigura ( )`
     - `public void testBuscarBinarioPorNombre( )`
   - Verifique que al ejecutar la clase de prueba **JuegoRompecabezasTest** los casos de prueba se ejecutan sin errores.

Parte3: Creación de listas en la interfaz

1. Cree el método `toString()` de la clase **Ficha** para que retorne una cadena que muestra la posición de la ficha en la figura y la región a la que pertenece separadas por un guión, tal como se muestra en la panel de Controles de fichas de la interfaz (Ver descripción.doc)

2. Complete la clase **PanelOrdenamientoFichas**:
   - Cree el atributo listaFichas de tipo JList
   - En el constructor:
     i. Inicialice la lista
     ii. Cree una variable de tipo JScrollPane para incluir a la lista anterior
   - Complete el método `public void actualizarListaFichas ( ArrayList fichas)` que actualiza el JList con el contenido del vector fichas.
   - Cree el método `public void valueChanged( ListSelectionEvent e )` que resalta la ficha que se está mostrando. Utilice el método resaltarFicha() de la clase **InterfazRompecabezas**.