Universidad de los Andes
Ingeniería de Sistemas y Computación
Algorítmica y Programación Por Objetos 2
Ejercicio Nivel 7: Huracanes

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 representar un objeto como texto con el método toString().
- Utilizará JList y JScrollPane para presentar listas en la interfaz gráfica.

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.

PREPARACIÓN

1. Descargue y ejecute el demo de la aplicación 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_huracanes** con el contenido.
4. Lea el enunciado del problema disponible en **n7_huracanes/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/sitio/images/cursosCupi2/apo2/tutoriales/n7_assert.pdf

PARTE 1: CONSTRUCCIÓN DE INVARIANTES

1. Revise el modelo del mundo en **n7_huracanes/docs/specs/ModeloConceptual.JPG** e identifique las clases del mundo.

2. Complete la clase **Huracan**
   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 **Huracan** el método `verificarInvariante()`. Utilice aserciones para validar el invariante que definió en el punto anterior.
   c. Llame el método para verificar el invariante en todos aquellos métodos de la clase que modifican el estado.

3. Revise la clase **SistemaMeteorologia**
   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 **SistemaMeteorologia** el método `verificarInvariante()`. Utilice aserciones para validar el invariante que definió en el punto anterior.

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

PARTE 2: COMPARACIONES, ORDENAMIENTO Y BÚSQUEDA

1. Complete la clase **Huracan**:
   a. Complete los métodos de comparación de la clase **Huracan**. Para ello utilice el método `compareTo` (para comparar cadenas de caracteres, ó `compareToIgnoreCase` para no tener en cuenta si son mayúsculas o minúsculas) o la expresión mayor o menor que (para comparar números).
   b. *(a) Defina y documente* la manera como debe probarse cada uno de los métodos que acaba de completar. Utilice el formato propuesto en las notas de clase para especificar los casos de prueba. Un posible formato lo encuentra en el archivo **CasosPrueba.doc** en el directorio **docs/specs** del proyecto. El escenario que debe usar está planteado en el método `setupEscenario1()` de la clase **HuracanTest**.
   (b) Implemente en la clase de pruebas **HuracanTest** 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 cada uno de los casos de prueba.
   c. Verifique que al ejecutar la clase de prueba **HuracanTest** los casos de prueba se ejecutan sin errores.

2. Complete la clase **SistemaMeteorologia** con ordenamientos:
   a. Complete el método `public void ordenarPorNombre()` que ordena los huracanes ascendentemente según su nombre, utilizando el algoritmo de **burbuja**.
   b. Complete el método `public void ordenarPorVelocidad()` que ordena los huracanes ascendentemente según la velocidad de sus vientos, utilizando el algoritmo de **inserción**.
   c. Complete el método `public void ordenarPorDanios()` que ordena los huracanes ascendentemente según el costo de los daños causados, utilizando el algoritmo de **selección**.
   d. *(a) Defina y documente* la manera como debe probarse cada uno de los métodos anteriores según el formato propuesto en las notas de clase para especificar los casos de prueba. Un posible formato lo encuentra en el archivo CasosPrueba.doc en el directorio docs/specs del proyecto.
   (b) Implemente en la clase de pruebas **SistemaMeteorologiaTest** los métodos de prueba para verificar el correcto funcionamiento de los métodos de ordenamiento desarrollados anteriormente. Utilice el método setupEscenario1 para definir el estado inicial de la prueba y los resultados esperados.
   e. **No** verifique todavía que al ejecutar la clase de prueba **SistemaMeteorologiaTest** los casos de prueba se ejecutan sin errores, pues aún falta implementar las siguientes modificaciones.

3. Complete la clase **SistemaMeteorologia** con las búsquedas:
   a. Implemente el método `public int buscarBinarioPorNombre(String)` que busca un huracán 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.
   b. Implemente el método `public int buscarHuracan(String)` que busca un huracán a partir de su nombre y retorna su posición. La búsqueda debe ser una búsqueda secuencial ya que no se tiene como precondición que la lista de huracanes esté ordenada
   c. Implemente el método `public int buscarHuracanMenorCostoDanios()` que busca el huracán con el menor valor de daños estimados y retorna su posición.
   d. Implemente el método `public int buscarHuracanMayorCostoDanios()` que busca el huracán con el mayor valor de daños estimados y retorna su posición.
   e. Implemente el método `public int buscarHuracanMayorVelocidad()` que busca el huracán con la mayor velocidad de vientos y retorna su posición.
   f. Implemente en la clase de pruebas **SistemaMeteorologiaTest** los métodos de prueba para verificar el correcto funcionamiento de los métodos de búsqueda desarrollados anteriormente. Utilice el método `setupEscenario1()` para definir el estado inicial de la prueba.

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

PARTE 3: CREACIÓN DE LISTAS EN LA INTERFAZ

1. Complete el método `toString()` de la clase **Huracan** para que retorne una cadena de caracteres con el nombre del huracán.

2. Complete la clase **PanelListaHuracanes**:
    a. Cree el atributo listaHuracanes de tipo JList
    b. En el constructor:
        i. Inicialice la lista
        ii. Cree una variable de tipo JScrollPane para incluir a la lista anterior
    c. Complete el método `public void refrescarLista(ArrayList nuevaLista)` que actualiza el JList con el contenido de la lista que entra como parámetro. Recuerde primero agregar los nuevos elementos (entran como parámetro) y luego establecer el índice seleccionado en el 0.
    d. Cree el método `public void valueChanged(ListSelectionEvent e)` el cual actualiza la información del panel de información de huracanes, según el huracán que se seleccionó. Para esto utilice el método `verDatos(Huracan huracán)` de la clase **InterfazHuracanes**.