SESIÓN 08

La práctica corresponde al
Tema 5: Estructuras arbóreas

Peso total en la nota final:
0%

Objetivos

Con esta práctica se pretende que el alumno se familiarice y comprenda:

1. Estructura de la jerarquía del *Java Collections Framework*
   a) Se estudia el interfaz Set
   b) Se estudia el interfaz SortedSet
   c) Se estudian las clases que implementan el interfaz
2. Cómo se emplean las clases de *tipo Set (Conjunto)*.
   a) Condiciones que deben cumplir los objetos a almacenar sobre los mismos (métodos *hashCode*, *equals*...etc.)
3. Como se realizan consultas sobre clases del tipo Set.

Descripción

Código Fuente

Se pretende generar una clase que simplifique a otros programadores la selección de elementos de un conjunto que cumplan una condición de igualdad/desigualdad con otro objeto que se pase como referencia. Se desea obtener elementos únicos (no repetidos) de la colección.

La clase encargada de realizar estas operaciones se llamará `SeleccionPredicados`. Se deberá contar internamente con cualquiera de las clases que implementan el interfaz `Set` en el *Java Collections Framework*.

Para completar las operaciones pedidas, se trabajará con cualquiera de las clases que implementan el interfaz `Set`. Desde esta colección, se pasará a obtener aquellos elementos que cumplan con la con la condición facilitada, devolviendo a su vez otro objeto del interfaz `Set`, que contendrá solo los objetos solicitados. Todos los métodos de selección deberán ser **No destructivos**.

La clase SeleccionPredicados tendrá un **tipo Enumerado**, de nombre Condicion, que permitirá definir que predicados reconoce para realizar la selección. Para este ejercicio, los predicados serán: `MAYOR`, `MAYORIGUAL`, `IGUAL`, `MENORIGUAL` y `MENOR`.

Los métodos públicos a incluir en la clase `SeleccionPredicados` son:

- `public <E> Set<E> seleccionaPredicado(Collection<E> coleccion, Condicion condicion, E referencia)`
- `public <E> Set<E> eliminaPredicado(Collection<E> coleccion,`

```
Condicion condicion, E referencia)
```

- `public <E> Set<E> seleccionaPredicado(Collection<E> coleccion, Condicion condicion, E referencia, Comparator<E> comparador)`
- `public <E> Set<E> eliminaPredicado(Collection<E> coleccion, Condicion condicion, E referencia, Comparator<E> comparador)`

Los primeros dos métodos se emplearán cuando los objetos contenidos en la colección sean comparables (no necesitan de comparador), mientras que los dos últimos métodos se emplearán cuando se trabaje con elementos no comparables.

Se deberá tener en cuenta que las colecciones que se pasan como parámetro pueden o no ser objetos que implementan el interfaz Set. En el caso de que además sean objetos que implementan el interfaz `SortedSet`, se deberá tener en cuenta que dicho interfaz proporciona métodos para el acceso a sus objetos ordenados y, por tanto, se deberán emplear éstos.

De esa forma, aunque exteriormente la clase solo presente 2 métodos de selección, se solicita que el alumno desarrolle dos métodos diferentes para realizar esa selección: una para las clases que implementen `SortedSet` (empleando métodos característicos de esta clase) y otra para el resto (empleando iteradores u otros accesos a la estructura completa). No se debe dar por supuesto que las clases de `SortedSet` facilitadas vaya a implementar también el interfaz `NavigableSet`.

Consideraciones: Se debe considerar que en el caso de trabajar con los `SortedSet` o `NavigableSet`, los objetos contenidos en la colección puede o no implementar Comparable. En caso de que no sea así, la forma de poder comparar entre si será el propio objeto `Comparator` que se pueda obtener del propio `Set`.

Se espera que las implementaciones facilitadas por el alumno sean lo más eficientes posibles desde el punto de vista de la complejidad algorítmica. Este aspecto se tendrá en cuenta a la hora de la evaluación de la práctica.

Informe

Se pide al alumno que, una vez completada la clase a programar, realice el un informe sencillo en el que se incluya un razonamiento de la complejidad algorítmica de cada uno de los métodos que se han implementado. No es necesario realizar un análisis completo y detallado de cada método (es decir, implementar las pruebas de tiempo de ejecución para el análisis de complejidad), pero sí incluir un razonamiento que relacione el código fuente que se facilita y a la complejidad que se considera que tiene el mismo.

La extensión del informe será de aproximadamente 5 páginas (incluyendo portadas, índices, gráficas...etc.).

Recursos Necesarios

Se facilita junto con este enunciado un fichero de test de jUnit que permite comprobar que el código generado responde a unos mínimos de corrección incluidos en la descripción de métodos anterior. Estos métodos de pruebas aparecen en el fichero `TestSeleccionPredicados`. Además, se incluyen los ficheros fuentes de las clases `Coche` y `GeneradorCoches` como parte de los tests.

Para obtener información adicional, se sugiere la consulta de los recursos:

http://docs.oracle.com/javase/tutorial/collections/interfaces/set.html

http://docs.oracle.com/javase/tutorial/collections/interfaces/sorted-set.html

http://docs.oracle.com/javase/tutorial/collections/implementations/set.html

Entrega

- La práctica se realizará en grupos de 1 o 2 personas.
- La entrega se hará SOLAMENTE por medio de la plataforma UBUVirtual. Cada miembro del grupo deberá subir su propia copia de la solución. No se admitirán soluciones fuera de plazo o por otros medios de entrega (e-mail, mensajería interna, ...).
- La entrega incluirá la información de quienes son los autores de la misma, tanto en los ficheros de código fuente como en los documentos asociados.
- Cada entrega consistirá en un fichero comprimido (formato .zip), con la estructura de nombre
  "`Apellidos1Nombre1_Apellidos2Nombre2`"
- Qué deberá incluir el fichero comprimido:
  - Código fuente de la solución (dentro de la estructura de paquetes necesaria)
  - Documentación en formato Javadoc
  - Informe en formato PDF (en aquellas prácticas que se solicite).
- No hace falta entregar ficheros binarios

Criterios de Evaluación

Se valorarán como puntos importantes de la práctica:

- **Corrección del funcionamiento**. Se valorará como aspecto más importante que el funcionamiento de la estructura de datos implementada cumpla con todas las condiciones solicitadas, tanto en la documentación oficial como en el enunciado. Se facilitan al alumno un subconjunto de los tests que se emplearán en la corrección, pero se aconseja que el alumno realice de forma adicional los que crea conveniente.

- **Complejidad algorítmica**: Como parte fundamental de los conocimientos de la asignatura, se considerará la complejidad algorítmica de las soluciones proporcionadas el segundo aspecto más importante a valorar en los ejercicios. Penalizará en la nota aquellas soluciones que, a pesar de proporcionar salidas correctas, lo hagan con una complejidad mucho mayor

de la estrictamente necesaria.

- **Informe**: se tendrá en cuenta la corrección del informe de complejidad algorítmica de la solución. No es necesario un informe excesivamente extenso, sino que se centre en la comparación de la complejidad algorítmica solicitada. La extensión aproximada es de 5 páginas más la portada y la tabla de contenidos.
- **Corrección del código**: Ausencia de warnings u otros elementos no deseables como variables no utilizadas, código que no se emplea, catch vacíos, ausencia de marcas sobre los métodos (@Override, @...) etc.
- **Documentación**: Las explicaciones y razonamientos que se incluyan a modo de comentarios en el código que entrega el alumno. Se solicita que el alumno entregue adicionalmente la documentación generada en formato javadoc, junto con los ficheros de código fuente.