

Taller de Lenguajes II

Práctica nº 9

Temas: Colecciones - Uso de Tipos Genéricos

1. Un diccionario es un “libro donde se relacionan palabras de significado similar”. Se cuenta con el siguiente **diccionario de sinónimos** que ud. debe representar empleando el framework de colecciones provisto por Java.

Palabras y sinónimos a almacenar	
PALABRA	SINÓNIMO
sillón	asiento, butaca, silla
casa	edificación, inmueble, hogar, obra
libro	ejemplar, manual, texto, obra
computador	equipo, ordenador, pc

- a. Escriba una clase llamada **DiccionarioDeSinonimos**, con la estructura adecuada para almacenar la información del diccionario de sinónimos, de modo que:
 1. Se inicialice en el **constructor** ó en un “**bloque de inicialización**” con los valores indicados en el cuadro.
 2. Posea un método llamado **getSinonimos** que recibe como argumento una palabra e imprime en pantalla los sinónimos asociados.
 3. Posea un método que imprime en pantalla todos los valores **claves** existentes en el diccionario
 4. Posea un método que imprime **TODO el contenido** del diccionario (puede verificar en <https://www.geeksforgeeks.org/iterate-map-java/> las distintas formas de iterar sobre la estructura)
 - b. Escriba una clase llamada **TestSinonimos** donde verifique el funcionamiento de los puntos anteriores.
 - c. Considere adicionalmente que el diccionario debería estar ordenado.
 1. ¿Qué estructura resulta más adecuada en este caso?
 2. Modifique su clase **DiccionarioDeSinonimos** de modo que haga uso de esta estructura. Vuelva a probarlo con el Test.
-
2. **Par Genérico.**
 - a. Escriba una clase que permite representar un “Par genérico”, es decir, que permita almacenar 2 elementos de cualquier tipo (Los tipos serán conocidos **al momento de la instanciación del Par**).
 - b. Escriba una clase **TestParGenerico** donde se verifique que la clase funciona correctamente.

3. Par Genérico Comparable.

- a. Ahora se quiere ampliar la funcionalidad del par genérico provisto en el punto 2, de modo que sólo pueda ser usada para almacenar elementos que sean **comparables**. Escriba la clase **ParGenericoComparable**
- b. Escriba una clase **TestParGenerico** donde se verifique que la clase funciona correctamente.

4. En el siguiente ejemplo, **HashSetCuentaAgregados** es un tipo especial de **HashSet** con la característica de poder consultar la cantidad total de elementos que se agregaron al mismo.

- a. Cree un proyecto Java y agregue la clase **HashSetCuentaAgregados**

```
public class HashSetCuentaAgregados<E> extends HashSet<E> {  
    private int cantidadAgregados = 0;  
    public HashSetCuentaAgregados() {  
    }  
  
    public HashSetCuentaAgregados(int initCap, float loadFactor){  
        super(initCap, loadFactor);  
    }  
  
    @Override  
    public boolean add(E e) {  
        cantidadAgregados++;  
        return super.add(e);  
    }  
  
    @Override  
    public boolean addAll(Collection<? extends E> c) {  
        cantidadAgregados += c.size();  
        return super.addAll(c);  
    }  
    public int getCantidadAgregados() {  
        return cantidadAgregados;  
    }  
}
```

- b. Genere una clase **TestHashSetCuentaAgregados** que servirá para realizar el siguiente test:
 - i. Agregue elementos a una instancia de **HashSetCuentaAgregados**. Use el método **addAll(...)** que permite agregar directamente una colección de objetos.
 - ii. Analice la cantidad de elementos resultantes. ¿Es correcta?, si es así **JUSTIFIQUE**, caso contrario indique **donde se produce el error**. Si es necesario adjunte en Eclipse el código fuente de la API de Java.