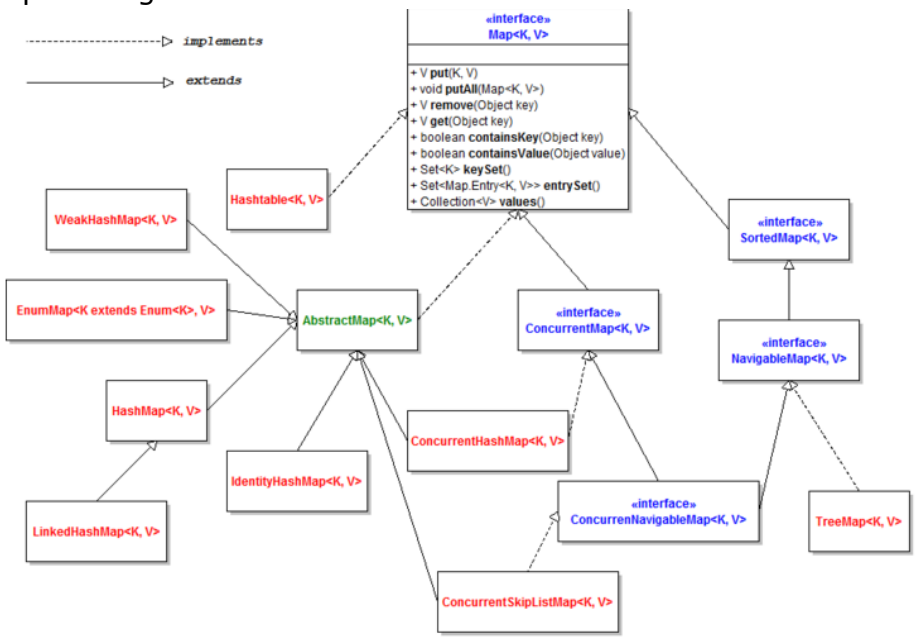
MAPAS

Perspectiva general



Mapa es el término java para designar a una estructura que permite el acceso directo por clave. Otros términos equivalentes son *array asociativo, tabla hash* o *diccionario(en Python)*. Se le llama mapa porque "mapea" "hace corresponder" una clave con un valor. Un mapa no puede contener claves con valores duplicados o nulos ya que no tienen sentido.

En el mapa se almacenan objetos del mismo tipo, es pues una colección como LinkedList o TreeSet. La novedad es que a los objetos del mapa se puede acceder con una suerte de acceso directo(no secuencial recorriendo la colección hasta encontrar el objeto buscado) a través de una clave. También los arrays permiten acceso directo pero en los arrays *la clave* sólo puede ser el índice de array, es decir un valor entero.

Los diferentes tipos de mapas java existen por cuestiones de almacenamiento. Cada forma de almacenamiento será más apropiada para un contexto determinado, por ejemplo:

- HashMap: Los elementos que inserta en el map no tendrán un orden específico.
- TreeMap: El Mapa está ordenado. Algo parecido al TreeSet.
- LinkedHashMap: Inserta en el Map los elementos en el orden en el que se van insertando; Algo parecido a nuestra tabla hash casera del boletín de estructuras dinámicas. El acceso directo no estan bueno como las otras pero para hacer barridos secuenciales no está mal.

Nos limitamos a manejar el interface *Map* y la clase *HashMap* (pero fíjate que hay muchas otras implementaciones de la clase Map en el API)

EL INTERFACE MAP

Mejor usar referencias tipo Map

Como siempre,es preferible, siempre que se pueda, trabajar al nivel más abstracto posible, por tanto, si no voy a utilizar métodos específicos de HashMap

```
HashMap<String,Double> hm= new HashMap<>();
```

Debería escribirlo como

```
Map<String,Double> miMapa= new HashMap<>();
```

Ejemplo:

Vemos como funciona los métodos put(),get(), containsKey() y containsValue(). Consulta estos métodos en el api

```
import java.util.HashMap;
import java.util.Map;
public class App {

    public static void main(String[] args) {
        Map<String,Double> hm= new HashMap<>();
        hm.put("Elías", new Double(1500));
       //mejor con  autoboxing
        hm.put("Román", 1900.0);
        hm.put("Telma", 2400.0);
        System.out.println("Sueldo de Román: "+ hm.get("Román"));
        System.out.println("Está en el mapa la llave Telma?: "+ hm.containsKey("Telma"));
        System.out.println("Está en el mapa el valor Telma?: "+ hm.containsValue("Telma"));
        Double d=1900.0;
        System.out.println("Está en el mapa el valor 1900?: "+ hm.containsValue(d));
        d=1899.98;
        System.out.println("Está en el mapa el valor 1899.98?: "+ hm.containsValue(d));
    }
}
```

Ejercicio U7_B7_E1: Los mapas nos interesan para hacer accesos directos por clave, no obstante en alguna situación puede interesarnos hacer un recorrido secuencial de todo el mapa. Hay varias formas de conseguir esto, la más sencilla es con el método KeySet(). Consulta en el API KeySet() y obtén los valores del mapa anterior con un bucle for mejorado que recorre las Key obtenidas por el método KeySet().

Las claves son únicas

No puede haber en un mapa dos entradas con la misma clave, es decir, no puede constar una clave más de una vez en un mapa.

Ejercicio U7_B7_E2: Fíjate en el API para ver cómo se comporta put() si se indica una clave existente.

put

```
public V put(K key,
    V value)
```

Associates the specified value with the specified key in this map. If the map previously contained a mapping for the key, the old value is replaced.

Specified by:

Como extensión del ejercicio anterior, comprueba que ocurre cuando intentas insertar algo con una clave existente en el mapa.

Un ejemplo de oracle:

En el siguiente ejemplo, queda bien claro que no se permiten claves duplicadas y que cuando insertamos(put) una clave que ya existe, no se genera un duplicado si no que el valor asociado a la clave "viejo" se sustituye por el "nuevo". Observa que aprovechando como trabaja un mapa tenemos un sencillo contador de frecuencias de apariciones de palabras. Fíjate en el uso "muy chulo" del operador condicional en el siguiente ejemplo.

https://docs.oracle.com/javase/tutorial/collections/interfaces/map.html

```
import java.util.*;
public class Freq {
    public static void main(String[] args) {
        Map<String, Integer> m = new HashMap<String, Integer>();

        // Initialize frequency table from command line
        for (String a : args) {
            Integer freq = m.get(a);
            m.put(a, (freq == null) ? 1 : freq + 1);
        }

        System.out.println(m.size() + " distinct words:");
        System.out.println(m);
    }
}
```

Y se ejecuta desde línea de comandos, por ejemplo

```
java Freq if it is to be it is up to me to delegate
```

Observa que el ejemplo debe de tener unos años y no utiliza el operador diamante pero eso es un pequeño detalle sintáctico sin importancia.

Inicialización rápida con of()

Para pequeños ejemplos, nos viene bien usar sintaxis de inicialización rápida para evitar muchos put. Hay muchas opciones, lo más cómodo es con el método of(), que como veremos en otro boletín se emplea mucho en entornos concurrentes pues devuelve una colección inmutable. Por ello, en el siguiente ejemplo el put() falla pues el mapa es inmutable

```
class App {
    public static void main(String[] args) {
            Map<String, String> map = Map.of("key1","value1", "key2", "value2");
            map.put("nuevaKey", "nuevoValor");

    }
}
```

Lo que podemos hacer para crear un nuevo mapa "normal" con of, es crear uno nuevo a partir del inmutable, simplemente pasando al constructor el mapa inmutable(lo copia y genera un nuevo mapa no inmutable)

```
import java.util.HashMap;
import java.util.Map;
class App {
    public static void main(String[] args) {
        Map<String, String> map = new HashMap<String,String>(Map.of("key1","value1", "key2", "value2"));
        map.put("nuevaKey", "nuevoValor");
        System.out.println(map.get("nuevaKey"));
    }
}
```

Ejercicio U7_B7_E3

Tenemos la siguiente tabla de latitudes/longitudes de ciudades nacionales e internacionales.

CIUDAD	LATITUD	LONGITUD
LUGO	43.01 N	7.33 O
BARCELONA	41.23 N	2.11 E
MADRID	40.24 N	3.41 O
LIMA	12.03 S	77.03 0

Las coordenadas son obligatoriamente objetos de la siguiente clase, a la que añadirás los métodos necesarios

```
class Coordenadas{
    private String latitud;
    private String longitud;
}
```

Los datos de la tabla anterior se almacenan obligatoriamente en un hashmap cuya clave es el nombre de la ciudad, y el valor almacenado será un objeto coordenada. Tu programa debe:

- Insertar los datos (para simplificar, directamente en el código, no por teclado).
- Imprimir el contenido del mapa.

Extraer los datos de un mapa sin usar acceso directo

Aunque el objetivo de los mapas es recuperar datos con acceso directo por clave, a menudo interesa hacer un tratamiento secuencial de los datos de un mapa. Podemos hacer el tratamiento secuencial de diversa formas por ejemplo, teniendo en cuenta los recursos aprendidos hasta el momentoimport java.util.*;

```
public class Freq {

    public static void main(String[] args) {

        Map<String, Integer> m = new HashMap<String, Integer>();



        // Initialize frequency table from command line

        for (String a : args) {
```

```
            Integer freq = m.get(a);
            m.put(a, (freq == null) ? 1 : freq + 1);
        }

        System.out.println(m.size() + " distinct words:");
        System.out.println(m);
    }
}
```

:

- Método KeySet():Podemos querer tener una lista secuencial de todas las claves. Más que lista será un conjunto ya que no hay claves duplicadas ni orden establecido entre ellas. Este método lo usamos en los ejemplos anteriores.
- Método public Collection<V> values(). Si queremos tener una relación de valores.
- public Set<Map.Entry<K,V>> entrySet(). Si queremos barrer el mapa secuencialmente, donde cada elemento de la secuencia es un par K,V. Este método devuelve un conjunto de "Entradas" donde cada entrada es un objeto Entry se parece a nuestra clase Pair de ejemplos anteriores.

Nota: Entry es un interface anidada del interface map (consulta en el api de Map "nested") de ahí ese nombre de sintaxis peculiar con un punto en el medio. De momento no sabemos lo que es una clase anidada pero no es necesario para entender este ejemplo.

Ejemplo: En el siguiente ejemplo observamos la utilización de los 3 métodos mencionados para el acceso secuencial a un mapa.

```
import java.util.*;
public class App {

    public static void main(String args[]) {

        Map<String, Double> sueldos = new HashMap<>();
        sueldos.put("Elías", 1500d);
        sueldos.put("Román", 1900d);
        sueldos.put("Telma", 2400d);

        Set<String> keySet = sueldos.keySet();
        System.out.println("Claves:" + keySet);
        Collection<Double> values = sueldos.values();
        System.out.println("Valores:" + values);
        System.out.println("Pares clave-valor:");
        Set<Map.Entry<String, Double>> entrySet = sueldos.entrySet();
        for (Map.Entry<String, Double> entry : entrySet) {
            System.out.println("key=" + entry.getKey() + " value="
                    + entry.getValue());
        }

    }
```

```
}
```

Cómo Map.Entry es static se ve mucho el import

```
import java.util.Map.Entry;
Lo que permite poner simplemente Entry

import java.util.*;
import java.util.Map.Entry;

public class App {

   public static void main(String args[]) {

      Map<String, Double> sueldos = new HashMap<>();
      sueldos.put("Elías", 1500d);
      sueldos.put("Román", 1900d);
      sueldos.put("Telma", 2400d);

      Set<String> keySet = sueldos.keySet();
      System.out.println("Claves:" + keySet);
      Collection<Double> values = sueldos.values();
      System.out.println("Valores:" + values);
      System.out.println("Pares clave-valor:");
      Set<Entry<String, Double>> entrySet = sueldos.entrySet();
      for (Entry<String, Double> entry : entrySet) {
         System.out.println("key=" + entry.getKey() + " value="
               + entry.getValue());
      }

   }
}
```

Ejercicio U7_B7_E4: Acepta el reto va de modas 152
Ya lo propusimos con arrays y con el boletín introductorio a mapas. Si no llegaste a hacerlo en su momento hazlo ahora. Con mapas es más fácil.

Ejercicio U7_B7_E5: reto Liga de pádel id 109
https://www.aceptaelreto.com/problem/statement.php?id=109&cat=18