L'interface Map de java

L'interface **`java.util.Map`** contient les pairs <clé, valeur>. Chaque pair est connu comme entrée. Map contient des éléments à clé unique.

L'interface `Map` peut être implémenté avec les collections d'objets qui héritent de lui. Les utilisations les plus populaires de `Map` sont `HashMap` et `TreeMap`. Chaque implémentation de ces collections d'objets est différente concernant l'ordre des éléments pendant le parcours:

`HashTable` ne garantit pas l'ordre des éléments.
`TreeMap` garantit l'ordre des éléments selon le parcours des pairs clé-valeur.

Voici un exemple comment créer une instance de Map:

```
Map hmap = new HashMap();
Map tmap = new TreeMap();
```

Les méthodes de l'interface Map

`1. public Object put(object key,Object value):`
Ajoute une clé associée à sa valeur.

```
Map hmap = new HashMap();
hmap.put(1, "e1");
hmap.put(2, "e2");
hmap.put(3, "e3");
```

`2. public void putAll(Map map)`
Insère une map spécifique dans cette map.

```
Map hmap2 = new HashMap();
hmap2.put(4, "e4");
hmap2.put(5, "e5");
hmap.putAll(hmap2);
```

`3. public Object get(object key):`
Récupérer une valeur avec sa clé, on veut obtenir la valeur de clé 2 de l'exemple précédent:

```
String e2 = (String) hmap.get(2);
```

`4. public Object remove(object key):`
Supprime une entrée d'une clé spécifique.

```
hmap.remove(1);
```

`5. public boolean containsKey(Object key)`
Rechercher une clé spécifique dans cette map.

`6. public boolean containsValue(Object value)`
Rechercher une valeur spécifique dans cette map.

`7. public Set keySet():`
Retourne l'ensemble de clés, `keySet()` est utile lors du parcours de la liste:

```
for(Object key : hmap.keySet()) {
 Object value = hmap.get(key);
 System.out.println(value);
}
```

`8. public Set entrySet():`
Retourne l'ensemble de clés et valeurs, `entrysSet()` est utile lors du parcours de la liste :

```
for (Map.Entry entry : hmap.entrySet()
{
 System.out.println(entry.getKey() + "-" + entry.getValue());
}
```

Map générique

Par défaut, vous pouvez mettre n'importe quel type dans votre map, mais vous pouvez aussi limiter le type d'objets pour parcourir les clés et les valeurs sans utiliser le cast:

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

Cette map accepte seulement les objets `Integer` pour les clés et `String` pour les valeurs. L'avantage de la généricité est l'accès aux éléments sans faire le cast:

```
import java.util.Hashtable;
import java.util.Map;

public class Exemple {

 public static void main(String a[]){
 //création
 Map ht = new Hashtable();

 //ajouter les clés-valeurs
 ht.put(1, "java");
 ht.put(2, "C");
 ht.put(3, "C++");

 for (Map.Entry entry : ht.entrySet())
 {
 int clé = entry.getKey();
 String valeur = entry.getValue();
 System.out.println(clé + "-" + valeur);
```

```
 }
 }
}
```

Résultats:

```
3-C++
2-C
1-java
```