A3P/IPO 2021/2022

Suppléments (hors unité)

Sommaire

Projet

4.1 HashMap

- HashMap ≈ « tableau associatif » qui contient un ensemble d'associations clé → valeur
- Classe du JDK => `import java.util.HashMap;`
- Plus général qu'un tableau classique qui impose le type `int` pour les indices (clé, valeur) => 2 types objets
- Donc pas `typeValeurs[]` mais

```
HashMap<typeClés, typeValeurs> vMaHM;
```

- `vMaHM.put( maClé, maValeur );`
- `quelleValeur = vMaHM.get( maClé );`

4.2 `HashMap` : exemple

- Plutôt que de stocker les sorties d'une pièce dans un tableau de 4 Rooms :
  tab[0] serait le nord ? tab[2] l'est ou l'ouest ?
- Associons chaque sortie Room à la direction "North" ou "South" …
- Donc dans la classe Room : un attribut
  `HashMap<String, Room> aSorties;`
- `aSorties.put( "North", vCuisine );`
- `quelleRoom = aSorties.get( "North" );`
- `null` si la clé n'est pas présente

4.3 `Set`, `keySet`, *for each*, `remove`

- `keySet` est une fonction de HashMap qui retourne l'ensemble des clés :

```
?vMesClés? = vMaHM.keySet();
```

- Donc pour `aSorties`, un ensemble de String, classe du JDK :

```
Set<String> vMesClés;
```

- On peut facilement parcourir tous ses éléments :

```
for ( String vS : vMesClés ) {
   S.o.p( vMaHM.get(vS).getDesc() );
} // boucle for each
```

- On peut aussi écrire `vMaHM.remove(vS);`

5. Extraction d'un nombre à partir d'une `String`

- entier ==> `Integer` ==> `parseInt`

```
int vN;
try {
    vN = Integer.parseInt( uneString );
}
catch ( final NumberFormatException pE ) {
    vN = uneValeurParDéfaut;
} // t/c
```

- réel ==> `Double` ==> `parseDouble`

6. Générateur aléatoire

- **Générateur de nombres (pseudo-)aléatoires :**
- `Random vMonGen = new Random();`
- `vMonGen.nextDouble()`
  retourne un réel dans l'intervalle [0.0, 1.0]
- `vMonGen.nextInt(N)`
  retourne un entier dans l'intervalle [0, `N`-1]

- `Random vMonGen = new Random( seed );`
  permet d'obtenir toujours la même suite de nombres.

7.1 Les classes anonymes

- Sans nom, définies 'à la volée' :
- Dans `UserInterface`
  `monBouton.addActionListener( ? );`
- `?` doit être un `ActionListener`, donc sa classe doit implémenter l'interface `ActionListener`.
- Dans *zuul-with-images*, on met `this` car `UserInterface implements ActionListener`, ce qui veut dire qu'elle redéfinit la méthode `actionPerformed`.

7.2 Les classes anonymes

```
monBouton.addActionListener(
   new ActionListener() {
     public void actionPerformed(
       final ActionEvent pE ) {
         actions à réaliser
     } // actionPerformed(.)
} );
```

- On crée une instance d'une classe anonyme (puisqu'on ne lui donne pas de nom !) qui implémente l'interface `ActionListener` en redéfinissant la méthode `actionPerformed`.
- **Attention !** Lorsqu'on utilise `this` entre les accolades, cela fait référence à l'instance de la classe anonyme.