TP 5

M.Adam – N.Delomez – JF.Kamp – L.Naert

9 août 2022

Objectifs du TP

- Utiliser des méthodes
- Écrire des méthodes
- Concevoir un programme en utilisant des méthodes

Exercice

L'objectif de cet exercice est de mettre en place un programme pour le jeu du Pendu. L'utilisateur doit deviner un mot en proposant des lettres. Quand la lettre proposée est dans le mot, elle est affichée au bon endroit. Si elle n'est pas dans le mot, une potence se dessine. Quand la potence est intégralement dessinée le joueur a perdu. Si le mot est deviné avant, le joueur a gagné. Neuf erreurs sont admises au maximum.

Pour réaliser le programme, nous allons décomposer le programme avec plusieurs méthodes.

1. La première structure à mettre en place est un dictionnaire contenant les mots à faire deviner à l'utilisateur.
   Écrire la méthode `creerDico()` qui crée un dictionnaire. Les mots du dictionnaire sont ajoutés directement dans le programme **sans saisie de la part de l'utilisateur**.
   ```
   /**
    * Création d'un dictionnaire de mots
    * @return dictionnaire initialisé
    */
   String[] creerDico ()
   ```

2. **Rendre le code de la méthode `creerDico()`, sa méthode de test et le résultat de l'exécution.**
3. Écrire une méthode `choisirMot()` qui rend un des mots pris au hasard dans le dictionnaire passé en paramètre.
   ```
   /**
    * choix aléatoire d'un mot dans un dictionnaire
    * @param dico dictionnaire des mots à choisir
   ```

```
 * @return chaine choisie de manière aléatoire
 */
String choisirMot (String[] dico)
```

Pour rappel, il est possible d'obtenir un nombre aléatoire avec la méthode `Math.random()` :

```
/**
 * Returns a double value with a positive sign, greater than or equal to 0.0
 * and less than 1.0.
 * @return a pseudorandom double greater than or equal to 0.0 and less than 1.0.
 */
double random()
```

En outre, il est possible de convertir un `float` ou un `double` en `int` par un cast (int) :

```
int i ;
double d;

i = (int) 4.7; // i vaut 4
d = 4.7;
i = (int) d;   // i vaut 4
```

4. **Rendre le code de la méthode `choisirMot()`, sa méthode de test et le résultat de l'exécution.**
5. La réponse de l'utilisateur, avec les lettres qu'il a trouvées, est stockée dans un tableau de caractères :

   ```
   char[] reponse;
   ```

   Soit la méthode `afficherReponse()` qui affiche sur une ligne la réponse de l'utilisateur passée en paramètre. Chaque lettre est séparée par un espace.

   ```
   /**
    * affiche la réponse du joueur
    * @param reponse reponse du joueur
    */
   void afficherReponse(char[] reponse) {
       for (int i = 0 ; i < reponse.length; i++) {
           System.out.print (reponse[i] + " ");
       }
       System.out.println ();
   }
   ```

   La méthode de test associée :

   ```
   /**
    * Test de afficheReponse()
    */
   void testAfficherReponse () {

       System.out.println ();
       System.out.println ("*** testAfficherReponse()");
   ```

```
        char[] reponse1 = {'P', 'R', 'O', 'G', 'R', 'A', 'M', 'M', 'E'};
        testCasAfficherReponse (reponse1);

        char[] reponse2 = {};
        testCasAfficherReponse (reponse2);
    }

    /**
     * teste un appel à afficherReponse()
     * @param reponse tableau des réponse à afficher
     */
    void testCasAfficherReponse (char[] reponse) {

        System.out.print ("afficherReponse (" + Arrays.toString(reponse) + ") : ");
        afficherReponse (reponse);
    }
```

6. **Rendre le code de la méthode `afficherReponse()`, sa méthode de test et le résultat de l'exécution.**
7. Écrire une méthode `creerReponse()` qui crée un tableau pour la réponse contenant uniquement des `_`.

```
/**
 * création d'un tableau de reponse contenant des '_'
 * @param lg longueur du tableau à créer
 * @return tableau de reponse contenant des '_'
 */
char[] creerReponse(int lg)
```

8. **Rendre le code de la méthode `creerReponse()`, sa méthode de test et le résultat de l'exécution.**
9. Écrire une méthode `tester()` qui prend en paramètre le mot à deviner, le tableau des réponses de l'utilisateur, le caractère saisi par l'utilisateur et qui rend vrai si la lettre est présente dans le mot, faux sinon. Enfin, la méthode remplace dans le tableau `reponse` le signe `_` par le caractère testé.

```
/**
 * teste la présence d'un caractère dans le mot
 * et le place au bon endroit dans réponse
 * @param mot     mot à deviner
 * @param reponse réponse à compléter si le caractère est présent dans le mot
 * @param car     caractère à chercher dans le mot
 * @return vrai ssi le caractère est dans le mot à deviner
 */
boolean tester (String mot, char[] reponse, char car)
```

Pour rappel de `String` :

```
/**
 * Returns the length of this string.
 * The length is equal to the number of Unicode code units in the string.
 * @return the length of the sequence of characters represented by this object.
 */
```

```
int length()


/**
 * Returns the char value at the specified index.
 * An index ranges from 0 to length() - 1.
 * The first char value of the sequence is at index 0, the next at index 1,
 * and so on, as for array indexing.
 * @return the char value at the specified index of this string.
 * The first char value is at index 0.
 */
char charAt(int index)
```

10. **Rendre le code de la méthode `tester()`, sa méthode de test et le résultat de l'exécution.**
11. Écrire la méthode `estComplet()` qui rend vrai si la réponse de l'utilisateur correspond lettre à lettre au mot cherché, faux sinon.

```
/**
 * rend vrai ssi le mot est trouvé
 * @param mot     mot à deviner
 * @param reponse réponse du joueur
 * @return vrai ssi le mot est égal caractère par caractère à la réponse
 */
boolean estComplet (String mot, char[] reponse)
```

12. **Rendre le code de la méthode `estComplet()`, sa méthode de test et le résultat de l'exécution.**
13. Écrire une méthode `partie()` qui permet de jouer une partie. Le dictionnaire est passé en paramètre.

```
/**
 * lancement d'une partie du jeu du pendu
 * @param dico dictionnaire des mots à deviner
 */
void partie(String[] dico)
```

14. **Rendre le code de la méthode `partie()`, sa méthode de test et le résultat de l'exécution.**
15. Écrire la méthode `principal()` du jeu.
16. **Rendre le code de la méthode `principal()` et le résultat d'une l'exécution.**