IUT de Paris Descartes – Base de la Programmation Objet

Travaux Dirigés – Sujet n°1

Objectifs

- Définir un type de donnée
- Définir des fonctions
- Manipuler des tableaux et des chaînes de caractères

Le jeu du Pendu

À terme, l'objectif est de développer une application permettant de jouer au pendu. Ce jeu consiste à découvrir un mot en proposant une à une les lettres pouvant le composer. À chaque tour de jeu, le joueur propose une nouvelle lettre. Si celle-ci apparaît dans le mot et n'a pas déjà été jouée alors le coup est un succès. Dans le cas contraire, c'est une erreur et le nombre total d'erreurs est limité à 5. Pour l'aider à choisir ses coups, il est indiqué au joueur quelle est la longueur du mot et pour chaque lettre, sa valeur si elle a déjà été découverte.

Il vous a été confié la charge de développer un type de donnée représentant un tel jeu. Ce type sera la base de la future application. Toutefois, vous ne savez pas quelle sera l'interface utilisateur qui sera mise en œuvre (graphique, textuelle, ordinateur personnel, smartphone, etc.). En conséquence, vous définirez un ensemble de fonctions suffisamment générales pour s'adapter aux différents besoins. Elles ne devront faire aucune entrée/sortie.

À faire

1. Déterminez les différentes données devant caractériser le jeu.

   Solution:
   - Le mot devant être trouvé.
   - Les lettres déjà trouvées par le joueur.
   - Le nombre d'erreurs déjà réalisées par le joueur.

2. Déterminez le rôle des différentes fonctions que vous devez proposer.

   Solution:
   - Initialisation (i.e. construction) – le mot devant être trouvé devra être fourni en paramètre.
   - Est-ce que la partie est finie ?
   - Si elle est finie, est-ce qu'elle est gagnée ?
   - Si elle n'est pas finie, jouer une lettre.
   - Quelle est la longueur du mot à trouver.
   - Pour chaque position dans le mot à trouver, est-ce que la lettre correspondante a été découverte et si oui, quelle est sa valeur ?

3. Donnez le prototype de chaque fonction. Lorsque c'est nécessaire, vous préciserez les préconditions à satisfaire.

Solution:

```
public class FonctionsPendu {
  public static void init(Pendu p, String mot) {
    assert(mot.length() > 0);
  }
  public static boolean fini(Pendu p) { }
  public static boolean gagné(Pendu p) { }
  public static boolean jouer(Pendu p, char c) {
    assert (!fini(p));
  }
  public static int nbErreurs(Pendu p) { }
  public static int longueurMotATrouver(Pendu p) { }
  public static boolean estTrouvée(Pendu p, int i) {
    assert (0 <= i && i < longueurMotATrouver(p));
  }
  public static char lettre(Pendu p, int i) {
    assert (estTrouvée(p, i));
  }
}
```

4. Déclarez le type de donnée représentant le jeu.

Solution: Nous choisissons d'indiquer les lettres trouvées par le joueur par un tableau de booléens de même longueur que le mot à trouver. Nous pourrions faire d'autres choix. Par exemple, nous pourrions stocker uniquement l'ensemble des lettres déjà trouvées (indépendamment de leur position dans le mot).

```
public class Pendu {
  public static final int NB_MAX_ERREURS = 5; // par anticipation

  String motATrouver;
  boolean[] lettresTrouvées;
  int nbErreurs;
}
```

5. Programmez la fonction permettant de jouer une lettre. Vous trouverez en annexe une description des fonctionnalités utiles concernant les chaines de caractère.

Solution:

```
public class FonctionsPendu {
  ...
  public static boolean jouer(Pendu p, char c) {
    assert (!fini(p));
    boolean trouvé = false;
    for (int i = 0; i < longueurMotATrouver(p); ++i)
      if (!estTrouvée(p, i) && p.motATrouver.charAt(i) == c) {
        p.lettresTrouvées[i] = true;
        trouvé = true;
      }
    if (!trouvé)
```

```
      ++p.nbErreurs;
    return trouvé;
  }
  ...
}
```

6. Programmez la fonction permettant de déterminer si la partie a été gagnée.

Solution: La fonction `perdu` est donnée car `gagné` repose sur elle.

```
public class FonctionsPendu {
  ...
  public static boolean perdu(Pendu p) {
    return p.nbErreurs >= Pendu.NB_MAX_ERREURS;
  }

  public static boolean gagné(Pendu p) {
    if (perdu(p))
      return false;
    for (boolean b : p.lettresTrouvées)
      if (!b)
        return false;
    return true;
  }
  ...
}
```

7. Programmez les autres fonctions.

Solution:

```
import java.util.Arrays;

public class FonctionsPendu {
  ...
  public static void init(Pendu p, String mot) {
    assert(mot.length() > 0);
    p.motATrouver = mot;
    p.lettresTrouvées = new boolean[mot.length()];
    Arrays.fill(p.lettresTrouvées, false);
    p.nbErreurs = 0;
  }

  public static boolean fini(Pendu p) {
    return perdu(p) || gagné(p);
  }

  public static int nbErreurs(Pendu p) {
    return p.nbErreurs;
  }

  public static int longueurMotATrouver(Pendu p) {
    return p.motATrouver.length();
```

```
  }

  public static boolean estTrouvée(Pendu p, int i) {
    assert (0 <= i && i < longueurMotATrouver(p));
    return p.lettresTrouvées[i];
  }

  public static char lettre(Pendu p, int i) {
    assert (estTrouvée(p, i));
    return p.motATrouver.charAt(i);
  }
}
```

8. Écrire un programme permettant à deux joueurs de s'affronter.

Solution:

```
import java.util.Scanner;

public class Appli {
  public static void afficher(Pendu p) {
    String s = "";
    for (int i = 0; i < FonctionsPendu.longueurMotATrouver(p); ++i) {
      if (FonctionsPendu.estTrouvée(p, i))
        s += FonctionsPendu.lettre(p, i);
      else
        s += "_";
    }
    s += ", " + FonctionsPendu.nbErreurs(p) + " erreur(s)";
    System.out.println(s);
  }

  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Entrez le mot à découvrir : ");
    String m = sc.next();
    Pendu p = new Pendu();
    FonctionsPendu.init(p, m);
    while (!FonctionsPendu.fini(p)) {
      afficher(p);
      System.out.print("Entrez une lettre : ");
      char c = sc.next().charAt(0);
      FonctionsPendu.jouer(p, c);
    }
    if (FonctionsPendu.gagné(p))
      System.out.println("Bravo");
    else
      System.out.println("Perdu, il fallait trouver '" + m + "'");
    sc.close();
  }
}
```

Annexe

Le type `String` représente les chaines de caractère. Les chaines saisies au clavier peuvent être obtenues via `sc.next()` et `sc.nextLine()` (la première lit un mot alors que la seconde une ligne) où `sc` represente un `Scanner`. Le programme suivant lit au clavier la suite des mots saisis jusqu'à rencontrer "`fin`". La dernière lettre de chaque mot lu est affiché. Notez les invocations `s.equals()`, `s.length()` et `s.charAt()`.

```
import java.util.Scanner

public class Exemple {
        public static void main(String[] arg) {
                Scanner sc = new Scanner(System.in);
                String s = sc.next();
                while (!s.equals("fin")) {
                        System.out.println(s.charAt(s.length() - 1));
                        s = sc.next();
                }
                sc.close();
        }
}
```