Université de Nice-Sophia Antipolis — mercredi 23 septembre 2020
PeiP2 — Durée: 3h

Algorithmique et Programmation Java
Travaux Pratiques – Séance n° 3

On veut représenter des cartes à jouer d'un jeu de 52 cartes. Une carte est représentée par la classe `Carte` qui possédera deux variables de *type énuméré*, une pour la valeur de la carte (du deux à l'as), et l'autre pour sa couleur (trèfle, carreau, cœur et pique) .

Les énumérations de Java ne permettent d'énumérer que des constantes représentées par des noms. La déclaration de type est introduite par le mot-clé `enum`, et ne peut pas être locale à une méthode.

On pourra déclarer les types énumérés `Couleur` et `Valeur` comme suit :

```
public enum Couleur { trèfle, carreau, coeur, pique }
public enum Valeur  { deux, trois, quatre, cinq, six, sept,
                      huit, neuf, dix, valet, dame, roi, as }
```

Pour déclarer une variable `c` de type `Couleur`, on écrira :

```
Couleur c;
```

Pour affecter une valeur particulière du type du type énuméré `Couleur` à la variable `c`, on pourra par exemple écrire :

```
c = Couleur.carreau;
```

1) Écrivez les déclarations de type énuméré précédentes dans deux fichiers, la première dans `Couleur.java`, et la seconde dans `Valeur.java`.

2) Déclarez une variable `v` de type `Valeur`, affectez lui la constante `Valeur.as`. Puis, à l'aide de `println`, écrivez la valeur de la variable `v` sur la sortie standard.

Le type `enum` de Java n'est pas un type élémentaire (comme c'est le cas dans de nombreux langages de programmation). Ce sont des objets issus de la classe `java.lang.Enum`. La manipulation des énumérations se fera à l'aide des méthodes de cette classe.

Par exemple, la méthode `values` renvoie l'ensemble des valeurs de l'énumération. À l'aide de l'énoncé *foreach*, le fragment de code suivant écrit sur la sortie standard toutes les valeurs du type énuméré `Couleur`.

```
for (Couleur c : Couleur.values())
    System.out.println(c);
```

3) Dans le fichier `Test.java`, déclarez la classe publique `Test` avec la méthode `main` contenant le fragment de code précédent. Compilez et exécutez votre classe `Test`.

4) Complétez la méthode `main` pour afficher sur la sortie standard toutes les valeurs du type énuméré `Valeur`.

Par défaut, à chaque valeur d'une énumération est associé son numéro d'ordre. Pour la première c'est 0, pour la seconde 1, *etc.* La méthode `ordinal()` permet d'obtenir ce numéro d'ordre. Par exemple, `Valeur.quatre.ordinal()` est égal à 2.

5) Écrivez le code qui affiche tous les numéros d'ordre des valeurs de `Couleur` et `Valeur`.

La déclaration d'une énumération peut être plus complexe que celles données précédemment. Une énumération peut comporter des variables, des méthodes et un constructeur, et il est également

possible d'associer d'autres valeurs de types *quelconques* à chaque valeur de l'énumération. Cette dernière possibilité va nous permettre d'associer une valeur numérique (un entier/`Integer`) à chaque carte (on considérera que le valet, la dame et le roi ont une valeur de 10, et l'as une valeur de 20).

Le type énuméré `Valeur` se récrit comme suit :

```
public enum Valeur {
       deux(2), trois(3), quatre(4), cinq(5),
       six(6), sept(7),  huit(8), neuf(9), dix(10),
       valet(10), dame(10), roi(10), as(20);

       private Integer valeur;
       private Valeur(Integer v) { valeur = v; }
       public Integer valeur() { return valeur; }
}
```

Le champ `valeur` contient la valeur numérique et la méthode `valeur()` renvoie cette valeur. Le constructeur `Valeur` initialise chaque valeur de l'énumération.

6) Modifiez votre programme précédent pour qu'il affiche sur la sortie standard toutes les valeurs du type énuméré `Valeur` avec leur valeur numérique correspondante. Cela doit donner :

```
deux : 2
trois : 3
quatre : 4
cinq : 5
six : 6
sept : 7
huit : 8
neuf : 9
dix : 10
valet : 10
dame : 10
roi : 10
as : 20
```

On peut maintenant écrire la classe `Carte` formée de deux variables privées de type `Couleur` et `Valeur` :

```
public class Carte {
    private Couleur couleur;
    private Valeur valeur;
}
```

7) Dans un fichier `Carte.java`, écrivez la classe `Carte` donnée ci-dessus.

8) Ajoutez à la classe `Carte` précédente le constructeur `Carte(Valeur v, Couleur c)` pour construire et initialiser une carte à jouer.

9) Dans un fichier `Jeu.java`, déclarez la classe publique `Jeu` munie de la méthode `main` dans laquelle vous créerez 2 cartes à jouer, par exemple la *dame de pique* et le *six de trèfle*.

10) Compilez et testez votre classe `Jeu`.

11) Ajoutez à la classe `Carte` la méthode `toString` pour qu'elle renvoie une représentation sous forme de `String` de la carte courante. Par exemple, pour la dame de pique et le six de trèfle, on

pourra avoir *"[dame(10),pique]"* et *"[six(6),trèfle]"*.

12) Dans la méthode `main` de votre classe `Jeu`, écrivez sur la sortie standard les deux cartes à jouer que vous avez précédemment créées. Compilez et testez à nouveau votre classe `Jeu`.

On souhaite maintenant comparer deux cartes à jouer selon leurs valeurs numériques. Par exemple, la dame de pique est supérieure au six de trèfle.

13) Ajoutez à la classe `Carte`, la méthode `compareTo` qui compare deux cartes, la carte courante et celle passée en paramètre. Cette méthode renvoie une valeur entière négative, égale à 0 ou positive, selon que la carte courante et inférieure, égale ou supérieure à la carte passée en paramètre. Cette méthode possède l'en-tête suivant :

```
public int compareTo(Carte c)
```

14) Dans la méthode `main`, affichez le résultat de la comparaison des deux cartes que vous avez créées précédemment.

On souhaite maintenant visualiser graphiquement les cartes à jouer avec la planche à dessin (*c.f.* td0). L'image d'une carte est donnée par un fichier *gif* dont le nom est de la forme *valeur-couleur*`.gif`. Par exemple, les fichiers des images de la dame de pique et du six de trèfle se nomment, respectivement, `dame-pique.gif` et `six-trèfle.gif`. Depuis la page `http:/users.polytech.unice.fr/~vg/index-peip2.html`, récupérez les images de toutes les cartes à jouer et installez-les dans votre espace utilisateur de façon qu'elles soient accessibles par votre application Java.

15) Dans votre classe `Carte`, ajoutez une variable `img` de type `PaD.Image` qui représentera l'image dessinable de la carte à jouer courante sur la planche à dessin. N'oubliez pas les directives d'importation pour accéder aux classes du paquettage `PaD`.

16) Modifiez le constructeur de la classe afin d'initialiser la variable `img`.

17) Dans la classe `Carte`, ajoutez la méthode `dessiner` pour dessiner l'image de la carte courante dans une planche à dessin à la coordonnée (x, y). L'en-tête de cette méthode est :

```
/**
 * Rôle : dessine la carte à jouer courante this en position (x,y)
 * sur la planche à dessiner graphique pad
 *
 *  @param x un <code>int</code>
 *  @param y un <code>int</code>.
 *  @param pad un <code>PlancheADessin</code>.
*/
public void dessiner(PlancheADessin pad, double x, double y)
```

18) Dans la méthode `main` de votre classe `Jeu`, visualisez dans une planche à dessin les deux cartes à jouer que vous avez créées précédemment. L'exécution de votre programme pourra donner :