UNIVERSITÉ TOULOUSE III PAUL SABATIER

Rédigé avec LaTeX
Version du 21 janvier 2014

Construction et réutilisation de composants logiciel

Collections et généricité

L3 Informatique
Semestre 6

Cours donné par
Rédigé par Antoine de Roquemaurel

2014

Table des matières

1 ArrayList et Iterator

1.1 Détection d'un Palindrome

```
1  boolean isPalindrome(String s) {
     ArrayList<Character> l;
3    // l contient les charactères de la String
     ListIterator iBegin = l.iterator();
5    ListIterator iEnd = l.iterator(l.size());
     boolean out = true;
7
     while(iBegin.hasNext() && iEnd.hasPrevious() &&
9        iBegin.nextIndex() <= iEnd.previousIndex() && out) {
       out = iBegin.netx().equals(iEnd.previous());
11   }

13   return out;
}
```

Listing 1.1 – Palinrome

1.2 Gestion de tâches

```
public abstract class Tache {
2   public abstract String toString();
}
4
public final class TacheCodage extends Tache {
6   private final String spec;
    public TacheCodage(final String spec) {
8     this.spec = spec;
    }
10
    public String getSpec() {
12    return spec;
    }
14
    public String toString() {
16    return "Code "+spec;
    }
18 }

20 public final class TacheTelephone extends Tache {
    private final String nom;
22  private final String numero;

24  public TacheTelephone(final String nom, final String numero) {
      this.nom = nom;
```

```
26      this.numero = numero;
    }
28
    public String getNom() {
30      return nom;
    }
32  pulic String getNumero() {
      return numero;
34  }

36  public String toString() {
      return "Telephone "+nom;
38  }
}
```

Listing 1.2 – Classes Tache TacheCodage et TacheTelephone

```
1  Tache appelerEric = new TacheTelephone("Ertineric", "0211223344");
   Tache appelerMartine = new TacheTelephone("", "0211223344");
3
   Tache coderBd = new TacheCodage("bd");
5  Tache coderIHM = new TacheCodage("ihm");
   Tache coderLogique = new TacheCodage("logique");
7
   // Le paramètre correspond au nombre d'élément intialement alloués.
9  // Ca permet d'éviter la réallocation inutile
   // Par défaut = 10
11 List tachsAppel = new ArrayList();
   List tachsCodage = new ArrayList();
13 List tachsLundi = new ArrayList(8);
   List tachsMardi = new ArrayList(8);
15
   tachesAppel.add(appelerEric);
17 tachesAppel.add(appelerMartine);

19 tachesCodage.add(coderBd);
   tachesCodage.add(coderLogique);
21 tachesCodage.add(1,coderIHM);

23 tachesLundi.add(coderLogique);
   tachesLundi.add(appelerMartine);
25 tachesLundi;set(1,appelerEric);

27 Tache toutesLesTaches = new ArrayList(tachesLundi);
   toutesLesTaches.addAll(tachesMardi);
29
   tachesLundi.remove(appelerEric);
31
   List tachesMardiNonAppel = new ArrayList(tachesMardi);
33 tachesMardiNonAppel.removeAll(tachsAppel);

35 tachesMardi.contains(appelerMartine);
   tachesMardi.containsAll(tachesMardiAppel);
```

Listing 1.3 – Exercices sur les taches

2 Généricité

```
public abstract class Valeur {
  public abstract String toString();
  public abstract boolean egale(Valeur valeur);
}

public class Nombre extends Valeur {
  private int nombre;
  public Nombre(int nombre) {
    this.nombre = nombre;
  }

  public int getNombre() {
    return nombre;
  }

  public String toString() {
    return "" + nombre;
  }

  public boolean egale(Valeur valeur) {
    return ((Nombre)valeur).getNombre() == nombre;
  }
}

public enum Image { ROI, DAME, VALET, AS };

public class Figure extends Valeur {
  private Image image;
  private String nom;

  public Figure(Image image) {
    this.image = image;
    this.nom = (image.toString().toLowerCase();
  }

  public Image getImage() {
    return image;
  }
  public String toString() {
    return nom;
  }

  public boolean egale(Valeur valeur) {
    return ((Figure)valeur).getImage() == image;
}

public Carte extends Pair<Valeur,Couleur> {
  public Carte(Valeur valeur, Couleur couleur) {
    super(valeur, couleur);
```

```
50    }
  }
52
  abstract public class Genre {
54    protected List valeurs;

56    public Genre(List pValeurs) {
        valeurs = pValeurs;
58    }
  }
60
  public class Atout extends Genre {
62    public Atout(List l) {
        super(l);
64    }
  }
66
  public class HorsAtout extends Genre {
68    public Atout(List l) {
        super(l);
70    }
  }
72
  public class Test {
74    public static void main(String[]s) {
        Carte asCarreau = new Carte(new Figure(Image.AS, Couleur.CARREAU));
76      Carte roiPiqe = new Carte(new Figure(Image.ROI, Couleur.PIQUE));
      }
78 }
```

Listing 2.1 – TD belotte

A Liste des codes sources