IPST-CNAM
Programmation JAVA
NFA 002
Jeudi 30 Juin 2011

Avec document
Durée : **2 h30**
Enseignant : LAFORGUE Jacques

1ère Session NFA 002 (CORRECTION)

1. Probleme 1 [10 points]

1/ et 2/

```
import java.util.*;

public class Recette{
   // Le nom de la recette est une chaine de caractère
   private String nomRecette;

   // La catégorie est une chaine parmis celles définies
   private String categorie;
   final static private String[] categories =
                  {"entree", "plat","dessert","boisson"};

   // Durée approximative de la recette exprimée en minutes
   private int duree;

   // La liste des ingrédients est un ArrayList sans limites
   private ArrayList<Ingredient> ingredients;

   // Le texte de la recette est un long texte
   private StringBuffer texte;

   // Constructeur d'une recette vide
   public Recette()
   {
      nomRecette="";
      categorie="";
      duree=0;
      ingredients = new ArrayList<Ingredient>();
      texte = new StringBuffer();
   }

   // Saisie de toutes les informations de la recette
   public void saisir()
   {
      Terminal.ecrireString("Le nom de la recette: ");
      nomRecette = Terminal.lireString();

      categorie = saisirCategorie();

      Terminal.ecrireString("La durée: ");
      duree = Terminal.lireInt();

      // Saisie des ingredients
      boolean fini=false;
      do{
         Ingredient ingr = new Ingredient();
         ingr.saisir();
         ingredients.add(ingr);
         Terminal.ecrireString("Voulez-vous saisir
                        un autre ingredient (o,n)? ");
         String rep = Terminal.lireString();
         if (rep.equals("n")) fini=true;
      }while(! fini);

      // Saisie du texte
      saisirTexteRecette();
   }

   // On verifie que la catégorie est bonne sinon
   //  on redemande
   static private String saisirCategorie()
   {
      String cat = "";
      boolean ok = false;
      do{
         Terminal.ecrireString("Categorie de la recette: ");
         cat = Terminal.lireString();
         for(String s:categories)
            if (cat.equals(s)) ok=true;
      }while(!ok);
      return(cat);
   }

   // On saisie ligne par ligne. Une ligne vide arrête la saisie
   private void saisirTexteRecette()
   {
      Terminal.ecrireStringln("Entrez le texte:");
      String ligne="";
      do{
         Terminal.ecrireString(">");
         ligne = Terminal.lireString();
         texte.append(ligne+"\n");
      }while(! ligne.equals(""));
   }

   // Pour tester le programme (hors sujet)
   public String toString()
   {
      return nomRecette+"\n"+categorie+"\n"+duree+"\n"+
         ingredients.toString()+"\n"+texte;
   }
}
```

```
// Classe de définition d'un ingredient
public class Ingredient
{
   private String quantite; //Texte libre
   private String nom;      // Texte libre

   // Saisie des deux informations
   public void saisir()
   {
      Terminal.ecrireString("Quantite de l'ingredient: ");
      quantite = Terminal.lireString();
```

```
        Terminal.ecrireString("Nom de l'ingredient: ");
        nom = Terminal.lireString();
    }

    // Pour tester le programme (hors sujet)
    public String toString()
    {
        return quantite+" de "+nom+"\n";
    }

}
```

3/

```
import java.util.*;
// Classe de définition d'un livre de cuisine
public class LivreCuisine
{
    // Les recette de cuisine sont rangés dans un ArrayList
    //  dans l'ordre d'arrivée (pas de classement)
    private ArrayList<Recette> recettes;

    // Constructeur : init de la collection
    public LivreCuisine()
    {
        recettes = new ArrayList<Recette>();
    }

    // Ajoute une nouvelle recette en demandant la saisie
    public void ajouterRecette()
    {
        Recette recette = new Recette();
        recette.saisir();
        recettes.add(recette);
    }

    // Pour tester le programme (hors sujet)
    public String toString()
    {
        return recettes.toString();
    }
}
```

2. Problème 2 [6 points]

1/

```
// Classe de définition du thread
public class DownLoadThread extends Thread
{
   // Le nbre de tentative pour télécharger un fichier
   //  (attribut statique car commun a tous les threads
   private static int nbTentative = 3;

   // Nom du fichier à télécharger
   private String nomFichier;

   // Constructeur du thread
   public DownLoadThread(String nomFichier)
   {
      this.nomFichier = nomFichier;
   }

   // Traitement du thread
   public void run()
   {
      try{
         boolean fini = false;
         int n=1;  // Initialisation du nombre de tentative
         while(! fini)
            {
               // Telechargement du fichier
               try{
                  Ftp.downLoadFile(nomFichier);
                  fini=true; // Si pas d'exception alors fini
               }
               // Le telechargement n'a pas abouti
               catch(FtpDownLoadException exFtp)
                  {
                     System.out.println("Erreur download "+n);
                     // Si le thread a fait nbTentative alors fini
                     if (n==nbTentative)
                        fini=true;
                     // sinon on continue le thread
                     else n++;
                  }
            }
      }
      catch(Exception l_ex){}
   }
}
```

2/

```
public class Probleme2
{
   public static void main(String[] args)
   {
      // Lecture du fichier
      StringBuffer fichiers = Terminal.lireFichier
                                    ("ToDownLoad.txt");
      // Decodage des lignes de texte
      String[] lignes = fichiers.toString().split("\n");

      // Pour chaque ligne qui est le nom du fichier,
      //  creer le thread et le lancer
      for(String fic:lignes)
         {
            DownLoadThread thread =
                              new DownLoadThread(fic);
            thread.start();
         }
   }
}
```

3. Problème 3 [4 points]

1/

```
// Tracage de l'objet graphique
public void draw()
{
   // On recupere tous les points du contour de l'objet
   ArrayList<Point> points = objetDrawable.getContour();

   // Pour chaque point on trace le trait
   //
   for(int i=0;i<points.size()-1;i++)
      {
         objetDrawable.tracer(points.get(i),
                              points.get(i+1),
                              foreground);
      }
   objetDrawable.tracer(points.get(points.size()-1),
                        points.get(0),
                        foreground);

}
```

2/

```
import java.util.*;
// Classe de definition de l'écran
public class Ecran
{
   // Les objets graphiques sont dans une collection
   //  (Pas d'ordre particulier)
   private ArrayList<ObjetGraphique> objetsGraphiques;

   // Constructeur de l'écran
   public Ecran()
   {
      objetsGraphiques = new ArrayList<ObjetGraphique>();
   }

   // Ajout de l'objet graphique dans la collection
   public void add(ObjetGraphique og)
   {
      objetsGraphiques.add(og);
   }

   // Tracage de tous les objets graphiques
   public void draw()
   {
      for(ObjetGraphique og:objetsGraphiques)
         og.draw();
   }
}
```

3/

Les classes Voiture et Individu doivent implémenter l'interface Drawable et donc implémenter les méthodes de l'interface Drawable. Ainsi c'est bien chaque classe qui sait quels points définissent sa forme et sa façon de dessiner les traits.

Ensuite pour chaque objet d'Individu ou de Voiture, on crée des instances de ObjetGraphique que l'on ajoute à un ecran (instance de Ecran).
Ainsi, on peut appeler la méthode draw de Ecran pour dessiner tous les objets.

(Voir l'exemple de code dans les sources de correction).

(Fin du sujet)