Exercices de java :

Introduction aux applications java :

Ecrire un programme qui demande à l'utilisateur de saisir 2 nombres entiers (boîte de dialogue).
Le programme doit ensuite afficher si le premier est multiple du second.

Solution :

```
/*
 * Fichier: Multiple.java
 * Crée le: 02 décembre 2006.
 * Auteurs: Sébastien ESTIENNE.
 * SiteWeb: http://www.prog-info.org/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 */

package chapitre1;


import javax.swing.JOptionPane;


/**
 * <p>Vérifie si un premier nombre est un multiple d'un second.</p>
 * @author Sébastien ESTIENNE.
 */
public class Multiple
{
   /**
    * <p>Débute l'exécution de l'application.</p>
    * @param args Les paramètres de la ligne de commande.
    */
   public static void main(String[] args)
   {
      // Chaines saisies par l'utilisateur.
      String premierNombre = "", deuxiemeNombre = "";

      // Nombres correspondants aux chaines saisies.
      int nombre1 = 0, nombre2 = 0;

      // Lire les nombres de l'utilisateur sous forme de String.
      premierNombre = JOptionPane.showInputDialog("Entrer le premier entier :");
      deuxiemeNombre = JOptionPane.showInputDialog("Entrer le deuxième entier :");

      // Converti les nombres du type String en type int.
      try
      {
         nombre1 = Integer.parseInt(premierNombre);
         nombre2 = Integer.parseInt(deuxiemeNombre);
      }
      // Si au moins une des deux valeurs n'est pas un nombre entier, cela génère exception qui
      // implique un traitement particulier.
      catch(NumberFormatException e)
      {
```

```
            // Afficher le message d'erreur.
            JOptionPane.showMessageDialog(null,
                  "Erreur, les 2 saisies doivent être des nombres entiers.", "Calculs",
                  JOptionPane.ERROR_MESSAGE);

            // Terminer l'application.
            System.exit(0);
         }

         // Affiche le résultat.
         if(nombre2 % nombre1 == 0)
         {
            JOptionPane.showMessageDialog(null, nombre2 + " est un multiple de " + nombre1,
                  "Multiple", JOptionPane.INFORMATION_MESSAGE);
         }
         else
         {
            JOptionPane.showMessageDialog(null, nombre2 + " n'est pas un multiple de " + nombre1,
                  "Multiple", JOptionPane.INFORMATION_MESSAGE);
         }

         // Terminer l'application.
         System.exit(0);
      }
}
```

STRUCTEUR DE CONTROLES :

Gestion de clients

Ecrire une application qui vérifie si un client (d'un magasin) a dépassé sa limite de crédit sur son compte débiteur. Pour chaque client, vous avez les données suivantes :
- le numéro de compte;
- le solde au début du mois;
- le total des articles portés en compte de ce client ce mois-ci;
- le total de tous les crédits appliqués au compte de ce client ce mois-ci;
- la limite de crédit autorisée.

L'utilisateur doit pouvoir saisir chaque donnée sous forme d'entiers dans des boîtes de dialogue d'entrée, calculer le nouveau solde (solde début - débits + crédits), afficher le nouveau solde et déterminer si le nouveau solde dépasse la limite de crédit du client (dans ce cas, il faut afficher le message "Limite de crédit dépassée").

SOLUTION :

```
/*
 * Fichier: Client.java
 * Crée le: 03 décembre 2006.
 * Auteurs: Sébastien ESTIENNE.
 * SiteWeb: http://www.prog-info.org/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 */

package chapitre3;
```

```java
import javax.swing.JOptionPane;



/**
 * <p>Gestion de comptes clients dans un magasin.</p>
 * @author Sébastien ESTIENNE.
 */
public class Client
{
   /**
    * <p>Débute l'exécution de l'application.</p>
    * @param args Les paramètres de la ligne de commande.
    */
   public static void main(String[] args)
   {
      // Entrées de l'utilisateur:
      // Le numéro du compte.
      String sNoCompte = "";

      // Le solde au début du mois.
      String sSolde = "";

      // Le total des articles portés en compte de ce client ce mois-ci.
      String sDebits = "";

      // Le total de tous les crédits appliqués au compte de ce client ce mois-ci.
      String sCredits = "";

      // La limite de crédit autorisée.
      String sLimite = "";

      // Données entrées par l'utilisateur converties.
      int noCompte = -1;
      double solde = 0, debits = 0, credits = 0, limite = 0;

      // Chaine de caractères pour l'affichage du nouveau solde.
      String sortie = "";

      // Demander à l'utilisateur le numero du compte.
      sNoCompte = JOptionPane.showInputDialog("Entrez le numéro du compte (-1 "
            + "pour terminer) :\n\n");

      // Convertir le numero du compte de String en entier.
      try
      {
         noCompte = Integer.parseInt(sNoCompte);
      }
      // Si la valeur n'est pas un nombre entier, cela génère une exception qui implique un
      // traitement particulier.
      catch(NumberFormatException e)
      {
         JOptionPane.showMessageDialog(null,
               "La valeur saisie est incorrecte, l'application va se terminer.", "Erreur",
               JOptionPane.ERROR_MESSAGE);
         noCompte = -1;
      }

      while(noCompte != -1)
      {
         // Demander à l'utilisateur le Solde, les Débits, les Crédits, la Limite du compte.
         sSolde = JOptionPane.showInputDialog("Entrez le solde au début du mois :\n\n");
         sDebits = JOptionPane.showInputDialog("Entrez le total des articles portés en "
               + "compte ce mois :\n\n");
         sCredits = JOptionPane.showInputDialog("Entrez le total des crédits appliqués "
               + "au compte ce mois :\n\n");
         sLimite = JOptionPane.showInputDialog("Entrez la limite de crédit autorisée " + ":\n\n");

         // Convertir le Solde, les Débits, les Crédits, la Limite du compte de String en double.
         try
         {
            solde = Double.parseDouble(sSolde);
            debits = Double.parseDouble(sDebits);
            credits = Double.parseDouble(sCredits);
            limite = Double.parseDouble(sLimite);
         }
```

```
            // Si une des valeurs n'est pas un nombre, cela génère une exception qui implique un
            // traitement particulier.
            catch(NumberFormatException e)
            {
               JOptionPane.showMessageDialog(null,
                     "Au moins une des valeurs saisies est incorrecte.\n"
                           + "Ces valeurs seront initialisées à 0.", "Erreur",
                     JOptionPane.ERROR_MESSAGE);
               noCompte = -1;
            }

            // Calculer la valeur du nouveau solde du mois.
            solde = solde - debits + credits;

            // Mettre à jour la chaine de sortie.
            sortie = "Le nouveau solde du compte numéro " + noCompte + " est : " + solde + ".\n";

            // Vérifier si la limite de crédit a été dépassée.
            if(solde + limite < 0)
               // On modifie la chaine de sortie.
               sortie += "La limite de crédit est dépassée.";

            // Afficher le nouveau solde du compte pour ce mois.
            JOptionPane.showMessageDialog(null, sortie, "Solde du mois",
                  JOptionPane.INFORMATION_MESSAGE);

            // Demander à l'utilisateur le numero du compte.
            sNoCompte = JOptionPane.showInputDialog("Entrez le numéro du compte (-1 "
                  + "pour terminer) :\n\n");

            // Convertir le numero du compte de String en entier.
            try
            {
               noCompte = Integer.parseInt(sNoCompte);
            }
            // Si la valeur n'est pas un nombre entier, cela génère une exception qui implique un
            // traitement particulier.
            catch(NumberFormatException e)
            {
               JOptionPane.showMessageDialog(null,
                     "La valeur saisie est incorrecte, l'application va se terminer.", "Erreur",
                     JOptionPane.ERROR_MESSAGE);
               noCompte = -1;
            }
         }

         // Terminer l'application.
         System.exit(0);
      }
   }
```

Maximum

Ecrire une application qui demande une suite de 10 nombres à un seul chiffre (sous la forme de caractères) puis détermine et affiche le plus grand de ces nombres.
Astuce : utilisez 3 variables :
- compteur : qui compte jusqu'à 10;
- nombre : le chiffre courant;
- plusGrand : le plus grand nombre connu jusqu'ici.

⊞
⊟Cacher la solution.

```
/*
 * Fichier: Maximum.java
 * Crée le: 03 décembre 2006.
 * Auteurs: Sébastien ESTIENNE.
 * SiteWeb: http://www.prog-info.org/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
```

```
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 */

package chapitre3;



import javax.swing.JOptionPane;



/**
 * <p>Calcule le chiffre maximum d'une série de 10 chiffres.</p>
 * @author Sébastien ESTIENNE.
 */
public class Maximum
{
   /**
    * <p>Débute l'exécution de l'application.</p>
    * @param args Les paramètres de la ligne de commande.
    */
   public static void main(String[] args)
   {
      // Le compteur.
      int compteur = 1;

      // Le nombre saisi par l'utilisateur.
      long nombre = 0;

      // Le chiffre maximum.
      int plusGrand = 0;

      // Demander à l'utilisateur les 10 nombres à 1 chiffre.
      try
      {
         nombre = Long.parseLong(JOptionPane.showInputDialog("Entrez les 10 nombres "
               + "à 1 chiffre :\n\n"));
      }
      // Si la valeur n'est pas un nombre entier, cela génère une exception qui implique un
      // traitement particulier.
      catch(NumberFormatException e)
      {
         JOptionPane.showMessageDialog(null,
               "La valeur saisie est incorrecte, l'application va se terminer.", "Erreur",
               JOptionPane.ERROR_MESSAGE);

         // Termine l'application.
         System.exit(0);
      }

      while(compteur <= 10)
      {
         // Tester si le chiffre courant est plus grand.
         if(nombre % 10 > plusGrand)
            plusGrand = (int) (nombre % 10);
         // Permet de passer au chiffre suivant.
         nombre /= 10;
         // Indique q'un nombre de plus vient d'être traité.
         compteur++;
      }

      // Afficher le chiffre maximum.
      JOptionPane.showMessageDialog(null, "Le plus grand nombre à un chiffre est : " + plusGrand
            + ".", "Maximum", JOptionPane.INFORMATION_MESSAGE);

      // Termine l'application.
      System.exit(0);
   }
}
```

Palindrome

Un palindrome est un nombre, un mot ou une phrase qui se lit de façon identique dans un sens et dans l'autre. Exemple de nombres qui sont des palindromes : 45654, 77777, 11511.
Ecrivez une application qui demande un entier de 5 chiffres et détermine si c'est un palindrome. Si le nombre ne comporte pas 5 chiffres affichez dans une boîte de message une erreur. Si l'utilisateur annule la boite de message, permettez-lui d'entrer une nouvelle valeur.

Cacher la solution.

```
/*
 * Fichier: Palindrome.java
 * Crée le: 03 décembre 2006.
 * Auteurs: Sébastien ESTIENNE.
 * SiteWeb: http://www.prog-info.org/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 */

package chapitre3;



import javax.swing.JOptionPane;



/**
 * <p>Indique si un nombre à 5 chiffres est un palindrome.</p>
 * @author Sébastien ESTIENNE.
 */
public class Palindrome
{
   /**
    * <p>Débute l'exécution de l'application.</p>
    * @param args Les paramètres de la ligne de commande.
    */
   public static void main(String[] args)
   {
      // Le nombre entré par l'utilisateur.
      String sNombre = "";

      // Le nombre converti.
      int nombre = -1;

      do
      {
         // Demander à l'utilisateur le nombre à 5 chiffres.
         sNombre = JOptionPane.showInputDialog("Entrez un nombre de 5 chiffres :\n\n");

         // Convertir sNombre de String en int.
         try
         {
            nombre = Integer.parseInt(sNombre);
         }
         // Si la valeur n'est pas un nombre entier, cela génère une exception qui implique un
         // traitement particulier.
         catch(NumberFormatException e)
```

```
            {
               // Ne rien faire.
            }
         }
         while(sNombre == null || sNombre.length() != 5 || nombre == -1);

         // Vérifier si le nombre est un palindrome (xyzyx)
         if(nombre / 10000 == nombre % 10 && nombre / 1000 % 10 == nombre / 10 % 10)
            // Afficher que le nombre est un palindrome.
            JOptionPane.showMessageDialog(null, nombre + " est un palindrome.", "Palindrome",
                  JOptionPane.INFORMATION_MESSAGE);
         else
            // Afficher que le nombre n'est pas un palindrome.
            JOptionPane.showMessageDialog(null, nombre + " n'est pas un palindrome.", "Palindrome",
                  JOptionPane.INFORMATION_MESSAGE);

         // Termine l'application.
         System.exit(0);
      }
   }
```

Vente d'articles

Une entreprise de vente par correspondance vend 5 produits dont les prix sont les suivants :
- produit numéro 1 : 2.67€,
- produit numéro 2 : 9.65€,
- produit numéro 3 : 3.78€,
- produit numéro 4 : 5.26€,
- produit numéro 5 : 7.21€,

Ecrivez une application qui lise une suite de paires de nombres (numéro du produit & quantité vendue dans la journée). Il faut utiliser une structure de contrôle switch pour déterminer le prix d'un produit. Il faut ensuite calculer et afficher le détail des produits vendus en une semaine. Pour la saisie des données, il faut utiliser un JTextField ; une boucle contrôlée par sentinelle permet d'afficher le résultat et d'arrêter le programme.

Cacher la solution.

```
/*
 * Fichier: Vente.java
 * Crée le: 03 décembre 2006.
 * Auteurs: Sébastien ESTIENNE.
 * SiteWeb: http://www.prog-info.org/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 */

package chapitre4;



import java.text.NumberFormat;
import java.util.Locale;

import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
```

```java
/**
 * <p>Simulation de vente d'articles.</p>
 * @author Sébastien ESTIENNE.
 */
public class Vente
{
   /**
    * <p>Trouve tous les triangles droits dont les côtés sont tous inférieurs à 500.</p>
    * @param args Les paramètres de la ligne de commande.
    */
   public static void main(String[] args)
   {
      // Le numéro du produit entré par l'utilisateur.
      String sNoProduit;
      int noProduit = 0;

      // La quantité du produit entré par l'utilisateur.
      String sQuantite;
      int quantite = 0;

      // 1 produit * quantite.
      double sousTotal;

      // Le total des produits.
      double total;

      // Indique si la saisie de l'utilisateur est valide.
      boolean saisieValide;

      // Créer un JTextArea pour afficher les résultats.
      JTextArea zoneSortie = new JTextArea(10, 28);

      // Relier le JTextArea à un JScrollPane pour que l'utilisateur puisse faire défiler
      // les résultats.
      JScrollPane defilant = new JScrollPane(zoneSortie);

      // Le contenu du JTextArea.
      String sortie;

      // Créer un decimalFormat pour mettre en forme les nombres à virgules flottantes avec
      // deux chiffres à droite du point décimal et un signe $.
      NumberFormat dollard = NumberFormat.getCurrencyInstance(Locale.FRANCE);

      // Demander le numéro du produit à l'utilisateur.
      do
      {
         sNoProduit = JOptionPane.showInputDialog("Entrez le numéro du produit[1-5] "
               + "(0 pour terminer) :\n\n");

         // Convertir sNoProduit de String en int.
         try
         {
            noProduit = Integer.parseInt(sNoProduit);
            saisieValide = true;
         }
         // La valeur saisie comme numéro de produit est invalide.
         catch(NumberFormatException e)
         {
            saisieValide = false;
         }
      }
      while(!saisieValide || noProduit < 0 || noProduit > 5);

      // Continuer d'ajouter des produits tant que la valeur sentinelle n'est pas rentrée.
      total = 0.0;
      sortie = "Produit\t Quantite\t Total\n";
      while(noProduit != 0)
      {
         // Demander le numéro du produit à l'utilisateur.
         do
         {
            sQuantite = JOptionPane.showInputDialog("Entrez la quantité du produit :\n\n");

            // Convertir sNoProduit de String en int.
            try
            {
               quantite = Integer.parseInt(sQuantite);
```

```
                saisieValide = true;
            }
            // La valeur saisie comme quantité est invalide.
            catch(NumberFormatException e)
            {
                saisieValide = false;
            }
        }
        while(!saisieValide || quantite < 0);

        // Calculer le sous total pour 1 produit donné.
        switch(noProduit)
        {
        case 1:
            sousTotal = 2.67 * quantite;
            break;
        case 2:
            sousTotal = 9.65 * quantite;
            break;
        case 3:
            sousTotal = 3.78 * quantite;
            break;
        case 4:
            sousTotal = 5.26 * quantite;
            break;
        case 5:
            sousTotal = 7.21 * quantite;
            break;
        default:
            sousTotal = 0.0;
            break;
        }

        // Calcul du montant total.
        total += sousTotal;

        // Pour ne pas afficher les produits non valides.
        if(noProduit > 0 && noProduit < 6)
            sortie += noProduit + "\t " + quantite + "\t " + dollard.format(sousTotal) + "\n";

        // Demander le numéro du produit à l'utilisateur.
        do
        {
            sNoProduit = JOptionPane.showInputDialog("Entrez le numéro du produit[1-5] "
                  + "(0 pour terminer) :\n\n");

            // Convertir sNoProduit de String en int.
            try
            {
                noProduit = Integer.parseInt(sNoProduit);
                saisieValide = true;
            }
            // La valeur saisie comme numéro de produit est invalide.
            catch(NumberFormatException e)
            {
                saisieValide = false;
            }
        }
        while(!saisieValide || noProduit < 0 || noProduit > 5);
    }

    // Ajouter le total à la chaine de sortie.
    sortie += "\nTOTAL : " + dollard.format(total);

    // Placer les résultats dans un JTextArea.
    zoneSortie.setText(sortie);

    // Afficher le contenu de la sortie dans le JTextArea avec la barre de défilement.
    JOptionPane.showMessageDialog(null, defilant, "Vente", JOptionPane.INFORMATION_MESSAGE);

    // Terminer l'application.
    System.exit(0);
  }
}
```

Méthodes :

Programme d'éducation

Construisez un programme qui aide les étudiants de niveau élémentaire à apprendre l'addition, la soustraction, la multiplication et la division.
L'étudiant doit pouvoir choisir entre 5 types de problèmes arithmétiques :
- 1 : addition
- 2 : soustraction
- 3 : multiplication
- 4 : division
- 5 : tous types

L'étudiant doit ensuite pouvoir sélectionner le niveau de difficulté :
- 1 : nombres à un seul chiffre.
- 2 : nombres à deux chiffres.
- 3 : nombres à 3 chiffres.

Utilisez Math.random pour générer aléatoirement des nombres.
Le programme affiche ensuite la question dans la barre d'état. Exemple : Combien fait 27 + 35? L'étudiant entre sa réponse dans un JTextField.
Si la réponse est correcte un des messages suivants est dessinée sur l'applet :
- C'est très bien!
- Excellent!
- Tu fais du beau travail!
- Bonne réponse, continue

Et une nouvelle question est posée.
Si la réponse est incorrecte un des messages suivants est dessinée sur l'applet :
- Désolé, essaie encore.
- Mauvaise réponse. Essaie une autre fois.
- Non, essaie à nouveau.

Et la même question est reposée.
Pour déterminer la phrase qui sera affichée utilisez un nombre aléatoire entre 1 et 4. Utilisez ensuite une structure switch pour afficher les messages.
Le programme doit compter le nombre de réponses correctes et incorrectes entrées par l'étudiant. Au bout de 10 réponses, calculez le taux de réponses correctes. Si celui-ci est inférieur à 75%, affichez le message "SVP, Demande à ton professeur de t'aider".
Un nouvelle session doit ensuite démarrer.

⊞
⊟Cacher la solution.

```
/*
 * Fichier: Education.java
 * Crée le: 03 décembre 2006.
 * Auteurs: Sébastien ESTIENNE.
 * SiteWeb: http://www.prog-info.org/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 */

package chapitre5;
```

```java
import java.awt.Container;
import java.awt.FlowLayout;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JApplet;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;



/**
 * <p>programme qui aide les étudiants de niveau élémentaire à apprendre l'addition, la
 * soustraction, la multiplication et la division.</p>
 * @author Sébastien ESTIENNE.
 */
public class Education extends JApplet implements ActionListener
{
   /**
    * <p>Serial version UID.</p>
    */
   private static final long serialVersionUID = 1L;

   // Le premier et second nombre.
   int nombre1, nombre2;

   // Réponse de l'utlisateur.
   int reponse;

   // Nombre de réponses correctes.
   int nbRepCorrectes;

   // Nombre de réponses fausses.
   int nbRepFausses;

   // Niveau de difficulté.
   int niveau;

   // Type d'exercice (+, -, *, /, tous).
   int typeExo;

   // Validité de la réponse de l'utilisateur.
   String commentaire = "";

   // L'opération entre les deux nomrbes.
   char cOperateur;

   // Composants de l'interface utilisateur graphique.
   JLabel etiquetteReponse;

   JTextField champReponse;


   /**
    * <p>Initialise les composants GUI de l'applet.</p>
    */
   @Override
   public void init()
   {
      // Obtenir le panneau de contenu et changer son layout en un FlowLayout.
      Container conteneur = getContentPane();
      conteneur.setLayout(new FlowLayout());

      // Créer l'étiquette et le champ texte du nombre à deviner.
      this.etiquetteReponse = new JLabel("Votre réponse : ");
      conteneur.add(this.etiquetteReponse);
      this.champReponse = new JTextField(10);
      this.champReponse.setEditable(true);
      conteneur.add(this.champReponse);
      this.champReponse.addActionListener(this);

      // Initialiser le niveau de difficulté.
```

```
    iniNiveau();

    // Initialiser le type d'exercice.
    iniExercice();

    // Initialiser la question.
    nouvelleQuestion();

    // Initialiser le nombre de réponses correctes et fausses.
    this.nbRepCorrectes = 0;
    this.nbRepFausses = 0;
}


/**
 * <p>Traiter une réponse.</p>
 * @param actionEvent Un événement sémantique qui indique qu'une action s'est produite dans le
 *            composant concerné.
 */
public void actionPerformed(ActionEvent actionEvent)
{
    int solution;

    // Convertir le nombre entré par l'utilisateur.
    this.reponse = Integer.parseInt(this.champReponse.getText());

    // Vider le champ réponse de l'utilisateur.
    this.champReponse.setText("");

    // Calculer la solution.
    switch(this.cOperateur)
    {
    case '+':
       solution = this.nombre1 + this.nombre2;
       break;
    case '-':
       solution = this.nombre1 - this.nombre2;
       break;
    case '*':
       solution = this.nombre1 * this.nombre2;
       break;
    default: // '/'
       solution = this.nombre1 / this.nombre2;
       break;
    }

    // Vérifier la validité de la réponse.
    if(this.reponse == solution)
    {
       this.commentaire = reponseCorrecte();
       this.nbRepCorrectes++;
       nouvelleQuestion();
    }
    else
    {
       this.commentaire = reponseFausse();
       this.nbRepFausses++;
    }

    // Cas de fin de l'exercice.
    if(this.nbRepCorrectes + this.nbRepFausses == 10)
    {
       // Message de taux de réussite.
       String message;

       // Taux de réussite.
       int taux;

       taux = this.nbRepCorrectes * 100 / (this.nbRepCorrectes + this.nbRepFausses);
       message = "Taux de réussite : " + taux + "%.";

       if(taux < 75)
          message += "\n\nDemande à ton professeur de t'aider.";

       // Affiche le résultat.
       JOptionPane.showMessageDialog(null, message, "Taux de réussite",
             JOptionPane.INFORMATION_MESSAGE);
```

```
      // Initialisation pour l'exercice suivant.
      nouvelleQuestion();
      this.nbRepCorrectes = 0;
      this.nbRepFausses = 0;
   }

   // Redessiner la fenêtre.
   repaint();
}


/**
 * <p>Affiche un message indiquant si la réponse de l'utilisateur est correcte.</p>
 * @param g La fenêtre graphique spécifiée.
 */
@Override
public void paint(Graphics g)
{
   // Appeler la version héritée de la méthode paint.
   super.paint(g);

   // Afficher la validité de la réponse de l'utilisateur.
   g.drawString(this.commentaire, 20, 75);

   // Poser la question dans la barre d'état.
   showStatus("Combien font " + this.nombre1 + " " + this.cOperateur + " " + this.nombre2 + ".");
}


/**
 * <p>Génère un nombre aléatoire.</p>
 * @return Le nombre généré aléatoirement.
 */
public int genereNombre()
{
   return (1 + (int) (Math.random() * Math.pow(10.0, this.niveau)));
}


/**
 * <p>Afficher une nouvelle question dans la barre d'état.</p>
 */
public void nouvelleQuestion()
{
   // Initialiser les nombres.
   this.nombre1 = genereNombre();
   this.nombre2 = genereNombre();

   if(this.typeExo == 5)
   {
      // Nombre aléatoire entre 0 et 3.
      int i = (int) (Math.random() * 4);

      // Sélectionne un type d'opération au hasard.
      switch(i)
      {
      case 1:
         this.cOperateur = '+';
         break;
      case 2:
         this.cOperateur = '-';
         break;
      case 3:
         this.cOperateur = '*';
         break;
      default:
         this.cOperateur = '/';
         break;
      }
   }
}


/**
 * <p>Sélectionne au hasard une phrase de réponse correcte.</p>
 * @return La réponse correcte au hasard.
```

```java
 */
public String reponseCorrecte()
{
   // Nombre aléatoire entre 0 et 3.
   int i = (int) (Math.random() * 4);

   // Initialisation de la phrase en fonction du nombre aléatoire.
   switch(i)
   {
   case 0:
      return "C'est très bien!";
   case 1:
      return "Excellent!";
   case 2:
      return "Tu fais du beau travail!";
   default:
      return "Bonne réponse, continue!";
   }
}


/**
 * <p>Sélectionne au hasard une phrase de réponse fausse.</p>
 * @return La réponse fausse au hasard.
 */
public String reponseFausse()
{
   // Nombre aléatoire entre 0 et 3.
   int i = (int) (Math.random() * 4);

   // Initialisation de la phrase en fonction du nombre aléatoire.
   switch(i)
   {
   case 0:
      return "Désolé, essai encore.";
   case 1:
      return "Mauvaise réponse. Essaie une autre fois.";
   case 2:
      return "Ne lâche pas, tu y es presque!";
   default:
      return "Non, essaie à nouveau.";
   }
}


/**
 * <p>Initialisation du niveau de difficulté.</p>
 */
public void iniNiveau()
{
   String sNiveau;

   this.niveau = 0;

   // Demander à l'utilisateur le niveau de difficulté tant que le niveau saisi est incorrect.
   while(this.niveau < 1 || this.niveau > 3)
   {
      sNiveau = JOptionPane.showInputDialog("Entrez le niveau de difficulté : \n"
            + " 1 - Facile\n" + " 2 - Moyen\n" + " 3 - Difficile\n\n");
      this.niveau = Integer.parseInt(sNiveau);
   }
}


/**
 * <p>Initialisation du type d'exercice.</p>
 */
public void iniExercice()
{
   String sTypeExo;

   this.typeExo = 0;

   // Demander à l'utilisateur le type d'exercice tant que le niveau saisi est incorrect.
   while(this.typeExo < 1 || this.typeExo > 5)
   {
      sTypeExo = JOptionPane.showInputDialog("Entrez le niveau de difficulté : \n"
```

```
                + " 1 - Addition\n" + " 2 - Soustraction\n" + " 3 - Multiplication\n"
                + " 4 - Division\n" + " 5 - Tous types\n\n");
            this.typeExo = Integer.parseInt(sTypeExo);
        }
    }
}
```

Puissance

Ecrivez une méthode récursive puissance(bas, exposant) qui renvoie base à la puissance exposant.
Exemple : puissance(5, 3) = 5 * 5 * 5.
Intégrez cette méthode à un applet qui permette à l'utilisateur d'entrer la base et l'exposant.


Cacher la solution.

```
/*
 * Fichier: Puissance.java
 * Crée le: 03 décembre 2006.
 * Auteurs: Sébastien ESTIENNE.
 * SiteWeb: http://www.prog-info.org/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 */

package chapitre5;



import java.awt.Container;

import javax.swing.JApplet;
import javax.swing.JOptionPane;
import javax.swing.JTextArea;



/**
 * <p>Méthode récursive puissance.</p>
 * @author Sébastien ESTIENNE.
 */
public class Puissance extends JApplet
{
    /**
     * <p>Serial version UID.</p>
     */
    private static final long serialVersionUID = 1L;

    // Base entré par l'utilisateur.
    String sBase;

    // Exposant entré par l'utilisateur.
    String sExposant;

    // Base convertie en nombre.
```

```
    int base;

    // Exposant converti en nombre.
    int exposant;


    /**
     * <p>Initialise les composants GUI de l'applet.</p>
     */
    @Override
    public void init()
    {
        // Calcul de la puissance.
        int resultat;

        // Créer un JTextArea pour afficher les résultats.
        JTextArea aireSortie = new JTextArea();

        // Obtenir le panneau de contenu.
        Container conteneur = getContentPane();

        // Lier aireSortie au conteneur.
        conteneur.add(aireSortie);

        // Demander à l'utilisateur un premier nombre et le convertir.
        this.sBase = JOptionPane.showInputDialog("Entrez la base :\n\n");
        this.base = Integer.parseInt(this.sBase);

        // Demander à l'utilisateur un deuxième nombre et le convertir.
        this.sExposant = JOptionPane.showInputDialog("Entrez l'exposant :\n\n");
        this.exposant = Integer.parseInt(this.sExposant);

        // Calculer base à la puissance exposant.
        resultat = puissance(this.base, this.exposant);

        aireSortie.setText(this.base + " puissance " + this.exposant + " est égal à "
              + Integer.toString(resultat));
    }


    /**
     * <p>Calculer une 'base' à la puissance 'exposant'.</p>
     * @param b Le nombre à élever (la base).
     * @param e L'exposant du nombre.
     * @return Le résultat du nombre élevé à la puissance.
     */
    public int puissance(int b, int e)
    {
        // b : base & e : exposant
        if(e == 1)
            return b;

        return b * puissance(b, e - 1);
    }
}
```

Tableaux :

Recherche binaire

La recherche binaire fonctionne sur un tableau trié. L'algorithme compare à la clé recherchée la valeur de l'élément au milieu du tableau. Si les deux valeurs sont égales alors on renvoie l'indice sinon on continue la recherche avec la moitié du tableau correspondant au résultat de la comparaison des deux valeurs. La recherche continue jusqu'à ce que l'on trouve l'élément ou jusqu'à ce que le sous tableau ne contienne plus qu'un seul élément différent de la clé, indiquant que celle-ci n'est pas dans le tableau.
Ecrivez un tel programme.

Cacher la solution.

```java
/*
 * Fichier: RechercheBinaire.java
 * Crée le: 04 janvier 2007.
 * Auteurs: Sébastien ESTIENNE.
 * SiteWeb: http://www.prog-info.org/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 */

package chapitre6;



import java.awt.Container;

import javax.swing.JApplet;
import javax.swing.JTextArea;



/**
 * <p>Effectue une recherche binaire avec un tableau trié.</p>
 * @author Sébastien ESTIENNE.
 */
public class RechercheBinaire extends JApplet
{
   /**
    * <p>Serial version UID.</p>
    */
   private static final long serialVersionUID = 1L;


   /**
    * <p>Initialise l'applet.</p>
    */
   @Override
   public void init()
   {
      // Tableau des valeurs à trier.
      int tableau[] = {5, 8, 17, 25, 31, 34, 43, 52, 56, 65};

      // Chaine de caractère de sortie.
      String sortie = "";

      // Zone de texte de sortie.
      JTextArea zoneSortie = new JTextArea();

      // Conteneur.
      Container conteneur = this.getContentPane();

      // Ajoute la zone de sortie au conteneur.
      conteneur.add(zoneSortie);

      // Ajoute à la chaine de sortie les valeurs du tableau.
      sortie += "Valeurs du tableau:\n";
      for(int i = 0; i < tableau.length; i++)
         sortie += "  " + tableau[i];
      sortie += "\n\n";

      // Effectue une recherche binaire.
      int valeur = 17;
```

```
        int element = rechercheBinaire(tableau, valeur);
        if(element != -1)
            sortie += "Valeur " + valeur + " trouvée à l'élément " + element + "\n";
        else
            sortie += "Valeur " + valeur + " introuvable!\n";

        // Met à jour la zone de sortie.
        zoneSortie.setText(sortie);
    }


    /**
     * <p>Effectue une recherche binaire.</p>
     * @param tableau Le tableau de valeurs.
     * @param cle L'élément à rechercher.
     * @return L'indice de la clé ou -1 si elle n'a pas été trouvée.
     */
    public int rechercheBinaire(int tableau[], int cle)
    {
        // Indice de l'élément du bas.
        int bas = 0;

        // Indice de l'élément du haut.
        int haut = tableau.length - 1;

        // Indice de l'élément du milieu.
        int milieu;

        // Parcours jusqu'à l'indice haut.
        while(bas <= haut)
        {
            // Calcule l'élément du milieu.
            milieu = (bas + haut) / 2;

            // La clé a été trouvée.
            if(cle == tableau[milieu])
                return milieu;

            // Modifie l'indice du haut du tableau.
            if(cle < tableau[milieu])
                haut = milieu - 1;
            // Modifie l'indice du bas du tableau.
            else
                bas = milieu + 1;
        }

        // Clé introuvable.
        return -1;
    }
}
```

Programmation basée sur les objets :

Livre

Ecrivez une classe Livre avec les attributs suivants:

- **titre**: Le titre du livre,
- **auteur**: L'auteur du livre,
- **prix**: Le prix du livre,
- **annee**: L'année du livre.

La classe Livre doit disposer des constructeurs suivants:

- **Livre**(),
- **Livre**(titre),
- **Livre**(titre, auteur),
- **Livre**(titre, auteur, prix),
- **Livre**(titre, auteur, prix, annee),

- **Livre**(Livre).

La classe Livre doit contenir des accesseurs et mutateurs pour les différents attributs. Elle doit aussi contenir une méthode toString() donnant une représentation de la classe Livre.
Ecrivez aussi une classe de testLivre afin de tester la classe Livre.

Cacher la solution.

```
/*
 * Fichier: Livre.java
 * Crée le: 13 janvier 2007.
 * Auteurs: Sébastien ESTIENNE.
 * SiteWeb: http://www.prog-info.org/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 */

package chapitre7.livre;



import java.text.DecimalFormat;



/**
 * <p>Classe représentant un livre.</p>
 * @author Sébastien ESTIENNE.
 */
public class Livre
{
   // Le titre du livre.
   private String titre;

   // L'auteur du livre.
   private String auteur;

   // Le prix du livre.
   private double prix;

   // L'année du livre.
   private int annee;


   /**
    * <p>Constructeur par défaut d'un livre.</p>
    */
   public Livre()
   {
      this("", "", 0.0, 0);
   }


   /**
    * <p>Constructeur de livre avec un titre et un auteur spécifiés.</p>
    * @param titre Le titre du livre.
    * @param auteur L'auteur du livre.
    */
   public Livre(String titre, String auteur)
   {
      this(titre, auteur, 0.0, 0);
```

```
}


/**
 * <p>Constructeur de livre avec un titre, un auteur et un prix spécifiés.</p>
 * @param titre Le titre du livre.
 * @param auteur L'auteur du livre.
 * @param prix Le prix du livre.
 */
public Livre(String titre, String auteur, double prix)
{
   this(titre, auteur, prix, 0);
}


/**
 * <p>Constructeur de livre avec un titre, un auteur, un prix et une année spécifiés.</p>
 * @param titre Le titre du livre.
 * @param auteur L'auteur du livre.
 * @param prix Le prix du livre.
 * @param annee L'année du livre.
 */
public Livre(String titre, String auteur, double prix, int annee)
{
   this.setTitre(titre);
   this.setAuteur(auteur);
   this.setPrix(prix);
   this.setAnnee(annee);
}


/**
 * <p>Constructeur de livre à partir d'un livre existant.</p>
 * @param livre Un livre.
 */
public Livre(Livre livre)
{
   this(livre.getTitre(), livre.getAuteur(), livre.getPrix(), livre.getAnnee());
}


/**
 * <p>Retourne l'année du livre.</p>
 * @return Renvoie l'année du livre.
 */
public final int getAnnee()
{
   return this.annee;
}


/**
 * <p>Modifie l'année du livre.</p>
 * @param annee L'année du livre.
 */
public void setAnnee(int annee)
{
   if(annee < 0)
      this.annee = 0;
   else
      this.annee = annee;
}


/**
 * <p>Retourne l'auteur du livre.</p>
 * @return Renvoie l'auteur du livre.
 */
public final String getAuteur()
{
   return this.auteur;
}


/**
 * <p>Modifie l'auteur du livre.</p>
 * @param auteur L'auteur du livre.
```

```java
     */
    public void setAuteur(String auteur)
    {
        this.auteur = auteur;
    }


    /**
     * <p>Retourne le prix du livre.</p>
     * @return Renvoie le prix du livre.
     */
    public final double getPrix()
    {
        return this.prix;
    }


    /**
     * <p>Modifie le prix du livre.</p>
     * @param prix Le prix du livre.
     */
    public void setPrix(double prix)
    {
        if(prix < 0)
            this.prix = 0.0;
        else
            this.prix = prix;
    }


    /**
     * <p>Retourne le titre du livre.</p>
     * @return Renvoie le titre du livre.
     */
    public final String getTitre()
    {
        return this.titre;
    }


    /**
     * <p>Modifie le titre du livre.</p>
     * @param titre Le titre du livre.
     */
    public void setTitre(String titre)
    {
        this.titre = titre;
    }


    /**
     * <p>Retourne une représentation du livre.</p>
     * @return Renvoie une représentation du livre.
     */
    @Override
    public String toString()
    {
        String resultat;
        DecimalFormat dFormat = new DecimalFormat("0.00");

        resultat = "Livre: {titre:" + this.getTitre() + "; auteur:" + this.getAuteur() + "; prix:"
              + dFormat.format(this.getPrix()) + " euros; année:" + this.getAnnee() + "}";

        return resultat;
    }
}



/*
 * Fichier: TestLivre.java
 * Crée le: 13 janvier 2007.
 * Auteurs: Sébastien ESTIENNE.
 * SiteWeb: http://www.prog-info.org/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
```

```
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 */

package chapitre7.livre;

/**
 * <p>Classe permettant de tester la classe Livre.</p>
 * @author Sébastien ESTIENNE.
 */
public class TestLivre
{
   /**
    * <p>Débute l'exécution du test.</p>
    * @param args Les paramètres de la ligne de commande.
    */
   public static void main(String[] args)
   {
      // Livre 1.
      Livre livre1 = new Livre();
      livre1.setTitre("Au coeur de Java 2 - Notions fondamentales");
      livre1.setAuteur("Cay S. Horstmann");
      livre1.setPrix(40.0);
      livre1.setAnnee(2005);
      System.out.println(livre1);

      // Livre 2.
      Livre livre2 = new Livre(livre1);
      livre2.setTitre("Au coeur de Java 2 - Fonctions avancées");
      livre2.setPrix(45.0);
      System.out.println(livre2);

      // Livre 3.
      Livre livre3 = new Livre("Pratique de .NET et C#2", "Patrick SMACCHIA", 50.0);
      livre3.setAnnee(2005);
      System.out.println(livre3);
   }
}
```

Point

Ecrivez une classe Point avec les attributs suivants:

- **x**: abscisse du point,
- **y**: ordonnée du point.

La classe Point doit disposer des constructeurs suivants:

- **Point**(): constructeur par défaut
- **Point**(x, y),
- **Point**(Point).

La classe Rectangle doit contenir des accesseurs et mutateurs pour les différents attributs. Elle doit aussi contenir les méthodes:

- **distance**(x, y),
- **distance**(Point),
- **distance**(x1, y1, x2, y2): version statique,
- **deplacer**(x, y),
- **translater**(dx, dy),
- **toString**(): donne une représentation d'un point.

Ecrivez aussi une classe testPoint afin de tester la classe Point.

⊞
⊟Cacher la solution.

```
/*
 * Fichier: Point.java
 * Crée le: 19 janvier 2007.
 * Auteurs: Sébastien ESTIENNE.
 * SiteWeb: http://www.prog-info.org/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 */

package chapitre7.point;



/**
 * <p>Classe représentant un nombre complexe.</p>
 * @author Sébastien ESTIENNE.
 */
public class Point
{
   // Abscisse du point.
   private double x;

   // Ordonnée du point.
   private double y;


   /**
    * <p>Constructeur par défaut d'un point.</p>
    */
   public Point()
   {
      this(0, 0);
   }


   /**
    * <p>Constructeur de points avec l'abscisse et l'ordonnée spécifiées.</p>
    * @param x L'abscisse du point.
    * @param y L'ordonnée du point.
    */
   public Point(double x, double y)
   {
      this.setX(x);
      this.setY(y);
   }


   /**
    * <p>Constructeur de points à partir d'un point existant.</p>
    * @param point Un point.
    */
   public Point(Point point)
   {
      this(point.x, point.y);
   }
```

```java
/**
 * <p>Calcule la distance par rapport à un autre point.</p>
 * @param x_ L'abscisse du point.
 * @param y_ L'ordonnée du point.
 * @return Renvoie la distance par rapport à un autre point.
 */
public double distance(double x_, double y_)
{
   return Math.sqrt(Math.pow(this.getX() - x_, 2) + Math.pow(this.getY() - y_, 2));
}


/**
 * <p>Calcule la distance par rapport à un autre point.</p>
 * @param point Le point.
 * @return Renvoie la distance par rapport à un autre point.
 */
public double distance(Point point)
{
   return Math.sqrt(Math.pow(this.getX() - point.getX(), 2)
         + Math.pow(this.getY() - point.getY(), 2));
}


/**
 * <p>Calcule la distance entre deux points.</p>
 * @param x1 L'abscisse du premier point.
 * @param y1 L'ordonnée du premier point.
 * @param x2 L'abscisse du second point.
 * @param y2 L'ordonnée du second point.
 * @return Renvoie la distance par rapport à un autre point.
 */
public static double distance(double x1, double y1, double x2, double y2)
{
   return Math.sqrt(Math.pow(x2 - x1, 2) + Math.pow(y2 - y1, 2));
}


/**
 * <p>Déplace le point.</p>
 * @param x_ La nouvelle abscisse du point.
 * @param y_ La nouvelle ordonnée du point.
 */
public void deplacer(double x_, double y_)
{
   this.setX(x_);
   this.setY(y_);
}


/**
 * <p>Translate le point.</p>
 * @param dx La translation suivant l'axe x.
 * @param dy La translation suivant l'axe y.
 */
public void translater(double dx, double dy)
{
   this.setX(this.getX() + dx);
   this.setY(this.getY() + dy);
}


/**
 * <p>Retourne l'abscisse du point.</p>
 * @return Renvoie l'abscisse du point.
 */
public double getX()
{
   return this.x;
}


/**
 * <p>Modifie l'abscisse du point.</p>
 * @param x L'abscisse du point.
 */
```

```java
   public void setX(double x)
   {
      this.x = x;
   }


   /**
    * <p>Retourne l'ordonnée du point.</p>
    * @return Renvoie l'ordonnée du point.
    */
   public double getY()
   {
      return this.y;
   }


   /**
    * <p>Modifie l'ordonnée du point.</p>
    * @param y L'ordonnée du point.
    */
   public void setY(double y)
   {
      this.y = y;
   }


   /**
    * <p>Retourne une représentation du point.</p>
    * @return Renvoie une représentation du point.
    */
   @Override
   public String toString()
   {
      String resultat;

      resultat = "Point(" + this.getX() + "," + this.getY() + ")";

      return resultat;
   }
}
```

```java
/*
 * Fichier: TestPoint.java
 * Crée le: 19 janvier 2007.
 * Auteurs: Sébastien ESTIENNE.
 * SiteWeb: http://www.prog-info.org/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 */

package chapitre7.point;



/**
 * <p>Classe permettant de tester la classe Point.</p>
 * @author Sébastien ESTIENNE.
 */
public class TestPoint
{
   /**
```

```
     * <p>Débute l'exécution du test.</p>
     * @param args Les paramètres de la ligne de commande.
     */
    public static void main(String[] args)
    {
        // Point 1.
        Point point1 = new Point();
        point1.setX(2.7);
        point1.setY(5.8);
        System.out.println("P1: " + point1);

        // Point 2.
        Point point2 = new Point(1.6, 6.3);
        System.out.println("P2: " + point2);

        // Point 3.
        Point point3 = new Point(point2);
        point3.setX(4.1);
        System.out.println("P3: " + point3);

        // Distance x3.
        double d1 = point3.distance(point1);
        System.out.println("Dist(P1-P3): " + d1);

        double d2 = point3.distance(point2.getX(), point2.getY());
        System.out.println("Dist(P2-P3): " + d2);

        double d3 = Point.distance(2.3, 4.2, -6.9, 3.7);
        System.out.println("Dist: " + d3);

        // Déplacer.
        point3.deplacer(-5.1, 4.7);
        System.out.println("Depl(P3): " + point3);

        // Translater.
        point3.translater(1.2, -0.7);
        System.out.println("Transl(P3): " + point3);
    }
}
```

Rationnel

Ecrivez une classe Rationnel qui définit les nombres rationnels. Les fractions doivent être stockées de manière irréductible. La classe a les attributs suivants:

- **numerateur**,
- **denominateur**.

La classe Rationnel doit disposer des constructeurs suivants:

- **Rationnel**(): constructeur par défaut,
- **Rationnel**(numerateur, denominateur),
- **Rationnel**(Rationnel).

La classe Rationnel doit contenir des accesseurs et mutateurs pour les différents attributs. Elle doit aussi contenir les méthodes:

- **ajouter**(Rationnel),
- **soustraire**(Rationnel),
- **multiplier**(Rationnel),
- **diviser**(Rationnel),
- **evaluer**(): renvoie le résultat sous la forme d'un nombre réel,
- **toString**(): donne une représentation du rationnel (a/b).

Ecrivez aussi une classe testRationnel afin de tester la classe Rationnel.


Cacher la solution.

```java
/*
 * Fichier: Rationnel.java
 * Crée le: 17 janvier 2007.
 * Auteurs: Sébastien ESTIENNE.
 * SiteWeb: http://www.prog-info.org/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 */

package chapitre7.rationnel;



/**
 * <p>Classe représentant un rationnel.</p>
 * @author Sébastien ESTIENNE.
 */
public class Rationnel
{
   // Le numérateur du nombre rationnel.
   private int numerateur;

   // Le dénominateur du nombre rationnel.
   private int denominateur;


   /**
    * <p>Constructeur par défaut d'un nombre rationnel.</p>
    */
   public Rationnel()
   {
      this(0, 1);
   }


   /**
    * <p>Constructeur de nombres rationnels à partir d'un numérateur et d'un dénominateur
    * spécifiés.</p>
    * @param numerateur Le numérateur du nombre rationnel.
    * @param denominateur Le dénominateur du nombre rationnel.
    */
   public Rationnel(int numerateur, int denominateur)
   {
      this.numerateur = numerateur;
      this.setDenominateur(denominateur);
   }


   /**
    * <p>Constructeur de nombre rationnel à partir d'un nombre rationnel existant.</p>
    * @param rationnel Un nombre rationnel.
    */
   public Rationnel(Rationnel rationnel)
   {
      this(rationnel.getNumerateur(), rationnel.getDenominateur());
   }


   /**
    * <p>Retourne le dénominateur du nombre rationnel.</p>
    * @return Renvoie le dénominateur du nombre rationnel.
    */
   public int getDenominateur()
   {
      return this.denominateur;
```

```
}


/**
 * <p>Modifie le dénominateur du nombre rationnel.</p>
 * @param denominateur Le dénominateur.
 */
public void setDenominateur(int denominateur)
{
   if(denominateur == 0)
      this.denominateur = 1;
   else
   {
      this.denominateur = denominateur;
      this.reduire();
   }
}


/**
 * <p>Retourne le numérateur du nombre rationnel.</p>
 * @return Renvoie le numérateur du nombre rationnel.
 */
public int getNumerateur()
{
   return this.numerateur;
}


/**
 * <p>Modifie le numérateur du nombre rationnel.</p>
 * @param numerateur Le numérateur.
 */
public void setNumerateur(int numerateur)
{
   this.numerateur = numerateur;
   this.reduire();
}


/**
 * <p>Déterminer le pgcd (plus grand commun diviseur) de deux nombres.</p>
 * @param x Le premier nombre.
 * @param y Le second nombre.
 * @return Le pgcd de des deux nombres.
 */
private int pgcd(int x, int y)
{
   if(x < 0)
      x = -x;
   if(y < 0)
      y = -y;

   while(x != 0 && y != 0)
   {
      if(x > y)
         x -= y;
      else
         y -= x;
   }

   if(x != 0)
      return x;

   return y;
}


/**
 * <p>Rend la fraction du nombre irréductible.</p>
 */
private void reduire()
{
   int pgcd = this.pgcd(this.getNumerateur(), this.getDenominateur());
   this.numerateur = this.getNumerateur() / pgcd;
   this.denominateur = this.getDenominateur() / pgcd;
}
```

```java
/**
 * <p>Ajoute un nombre rationnel au nombre rationnel courant.</p>
 * @param rationnel Le nombre rationnel à ajouter.
 */
public void ajouter(Rationnel rationnel)
{
   this.setNumerateur(this.getNumerateur() * rationnel.getDenominateur()
         + rationnel.getNumerateur() * this.getDenominateur());
   this.setDenominateur(this.getDenominateur() * rationnel.getDenominateur());
   this.reduire();
}


/**
 * <p>Soustrait un nombre rationnel au nombre rationnel courant.</p>
 * @param rationnel Le nombre rationnel à soustraire.
 */
public void soustraire(Rationnel rationnel)
{
   this.setNumerateur(this.getNumerateur() * rationnel.getDenominateur()
         - rationnel.getNumerateur() * this.getDenominateur());
   this.setDenominateur(this.getDenominateur() * rationnel.getDenominateur());
   this.reduire();
}


/**
 * <p>Multiplie un nombre rationnel au nombre rationnel courant.</p>
 * @param rationnel Le nombre rationnel à multiplier.
 */
public void multiplier(Rationnel rationnel)
{
   this.setNumerateur(this.getNumerateur() * rationnel.getNumerateur());
   this.setDenominateur(this.getDenominateur() * rationnel.getDenominateur());
   this.reduire();
}


/**
 * <p>Divise un nombre rationnel au nombre rationnel courant.</p>
 * @param rationnel Le nombre rationnel à diviser.
 */
public void diviser(Rationnel rationnel)
{
   this.setNumerateur(this.getNumerateur() * rationnel.getDenominateur());
   this.setDenominateur(this.getDenominateur() * rationnel.getNumerateur());
   this.reduire();
}


/**
 * <p>Evalue la valeur (nombre réel) du nombre rationnel.</p>
 * @return Retourne une évaluation du nombre rationnel.
 */
public double evaluer()
{
   return this.getNumerateur() / (double) this.getDenominateur();
}


/**
 * <p>Retourne une représentation du nombre rationnel.</p>
 * @return Renvoie une représentation du nombre rationnel.
 */
@Override
public String toString()
{
   String resultat = "";

   if(this.getNumerateur() > 0 && this.getDenominateur() < 0 ||
      this.getNumerateur() < 0 && this.getDenominateur() > 0)
      resultat += "-";
   resultat += Math.abs(this.getNumerateur()) + " / " + Math.abs(this.getDenominateur());

   return resultat;
```

```
   }
}
```

```
/*
 * Fichier: TestRationnel.java
 * Crée le: 17 janvier 2007.
 * Auteurs: Sébastien ESTIENNE.
 * SiteWeb: http://www.prog-info.org/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 */

package chapitre7.rationnel;


/**
 * <p>Classe permettant de tester la classe Rationnel.</p>
 * @author Sébastien ESTIENNE.
 */
public class TestRationnel
{
   /**
    * <p>Débute l'exécution du test.</p>
    * @param args Les paramètres de la ligne de commande.
    */
   public static void main(String[] args)
   {
      //   Gérer le signe...


      // Rationnel 1.
      Rationnel rationnel1 = new Rationnel();
      rationnel1.setNumerateur(5);
      rationnel1.setDenominateur(12);
      System.out.println("R1: " + rationnel1);

      // Rationnel 2.
      Rationnel rationnel2 = new Rationnel(-7, -15);
      System.out.println("R2: " + rationnel2);

      // Rationnel 3.
      Rationnel rationnel3 = new Rationnel(rationnel2);
      rationnel3.setNumerateur(9);
      System.out.println("R3: " + rationnel3);

      // Ajout.
      rationnel3.ajouter(rationnel1);
      System.out.println("R3+R1: " + rationnel3);

      // Soustraction.
      rationnel3.soustraire(rationnel2);
      System.out.println("R3-R2: " + rationnel3);

      // Multiplication.
      rationnel3.multiplier(rationnel1);
      System.out.println("R3*R1: " + rationnel3);

      // Division.
      rationnel3.diviser(rationnel2);
```

```
        System.out.println("R3/R2: " + rationnel3);

        // Evaluation.
        System.out.println("eval(R3): " + rationnel3.evaluer());
    }
}
```

Programmation orientée objet :

Employé

Ecrivez une classe abstraite Employe avec les attributs suivants:

- **nom**: Le nom de famille de l'employé,
- **prenom**: Le prénom de l'employé.

La classe Employe doit disposer des constructeurs suivants:

- **Employe**(),
- **Employe**(nom, prenom).

La classe Employe doit contenir des accesseurs et mutateurs pour les différents attributs et les méthodes suivantes:

- **toString**(): retourne une représentation d'un employé,
- **gains**(): retourne le salaire (abstraite).

Ecrivez une classe Patron héritant de Employé avec les attributs suivants:

- **salaire**: Le salaire mensuel.

La classe Patron doit disposer des constructeurs suivants:

- **Patron**(),
- **Patron**(nom, prenom, salaire).

La classe Patron doit contenir des accesseurs et mutateurs pour les différents attributs et les méthodes suivantes:

- **toString**(): retourne une représentation du patron,
- **gains**(): retourne le salaire du patron.

Ecrivez une classe TravailleurCommission héritant de Employé avec les attributs suivants:

- **salaire**: Le salaire mensuel de base,
- **commission**: Montant de la commission par article vendus,
- **quantite**: nombre d'articles vendus par mois.

La classe TravailleurCommission doit disposer des constructeurs suivants:

- **TravailleurCommission**(),
- **TravailleurCommission**(nom, prenom, salaire, commision).

La classe TravailleurCommission doit contenir des accesseurs et mutateurs pour les différents attributs et les méthodes suivantes:

- **toString**(): retourne une représentation du travailleur à la commission,
- **gains**(): retourne le salaire du travailleur à la commission.

Ecrivez une classe TravailleurHoraire héritant de Employé avec les attributs suivants:

- **retribution**: La rétrubution horaire,
- **heures**: Le nombre d'heures de travail par mois.

La classe TravailleurHoraire doit disposer des constructeurs suivants:

- **TravailleurHoraire**(),
- **TravailleurHoraire**(nom, prenom, retribution).

La classe TravailleurHoraire doit contenir des accesseurs et mutateurs pour les différents attributs et les méthodes suivantes:

- **toString**(): retourne une représentation du travailleur horaire,
- **gains**(): retourne le salaire du travailleur horaire.

Ecrivez aussi une classe de testEmploye afin de tester les classes.
Utilisez les propriétés du polymorphisme.

Cacher la solution.

```
/*
 * Fichier: Employe.java
 * Crée le: 30 janvier 2007.
 * Auteurs: Sébastien ESTIENNE.
 * SiteWeb: http://www.prog-info.org/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 */

package chapitre8.employe;



/**
 * <p>Classe représentant un employé.</p>
 * @author Sébastien ESTIENNE.
 */
public abstract class Employe
{
   // Le nom de l'employé.
   protected String nom;

   // Le prénom de l'employé.
   protected String prenom;


   /**
    * <p>Constructeur par défaut.</p>
    */
   public Employe()
   {
      this("", "");
   }


   /**
    * <p>Constructeur d'employé avec un nopm et un prénom spécifiés.</p>
    * @param nom Le nom de l'employé.
    * @param prenom Le prénom de l'employé.
    */
   protected Employe(String nom, String prenom)
   {
      this.setNom(nom);
```

```
      this.setPrenom(prenom);
   }


   /**
    * <p>Calcule le salaire de l'employé par mois.</p>
    * @return Retourne le salaire de l'employé par mois.
    */
   public abstract double gains();


   /**
    * <p>Retourne le nom de l'employé.</p>
    * @return Renvoie le nom de l'employé.
    */
   public String getNom()
   {
      return this.nom;
   }


   /**
    * <p>Modifie le nom de l'employé.</p>
    * @param nom Le nom de l'employé.
    */
   public void setNom(String nom)
   {
      this.nom = nom;
   }


   /**
    * <p>Retourne le prénom de l'employé.</p>
    * @return Renvoie le prénom de l'employé.
    */
   public String getPrenom()
   {
      return this.prenom;
   }


   /**
    * <p>Modifie le prénom de l'employé.</p>
    * @param prenom Le prénom de l'employé.
    */
   public void setPrenom(String prenom)
   {
      this.prenom = prenom;
   }


   /**
    * <p>Retourne une représentation de l'employé.</p>
    * @return Renvoie une représentation de l'employé.
    */
   @Override
   public String toString()
   {
      return "Employé[nom:" + this.getNom() + ", prénom:" + this.getPrenom() + "]";
   }
}




/*
 * Fichier: Patron.java
 * Crée le: 30 janvier 2007.
 * Auteurs: Sébastien ESTIENNE.
 * SiteWeb: http://www.prog-info.org/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
```

```
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 */

package chapitre8.employe;



/**
 * <p>Classe représentant un patron.</p>
 * @author Sébastien ESTIENNE.
 */
public class Patron extends Employe
{
   // Le salaire mensuel du patron.
   protected double salaire;


   /**
    * <p>Constructeur par défaut.</p>
    */
   public Patron()
   {
      this("", "", 0.0);
   }


   /**
    * <p>Constructeur de patron avec un nom, un prénom et un salaire spécifiés.</p>
    * @param nom Le nom du patron.
    * @param prenom Le prénom du patron.
    * @param salaire Le salaire du patron.
    */
   public Patron(String nom, String prenom, double salaire)
   {
      super(nom, prenom);
      this.setSalaire(salaire);
   }


   /**
    * <p>Calcule le salaire de l'employé par mois.</p>
    * @return Retourne le salaire de l'employé par mois.
    */
   @Override
   public double gains()
   {
      return this.getSalaire();
   }


   /**
    * <p>Retourne le salaire mensuel du patron.</p>
    * @return Renvoie le salaire mensuel du patron.
    */
   public double getSalaire()
   {
      return this.salaire;
   }


   /**
    * <p>Modifie le salaire mensuel du patron.</p>
    * @param salaire Le salaire mensuel du patron.
    */
   public void setSalaire(double salaire)
   {
      if(salaire < 0)
         this.salaire = 0.0;
      else
         this.salaire = salaire;
```

```
   }


   /**
    * <p>Retourne une représentation du patron.</p>
    * @return Renvoie une représentation du patron.
    */
   @Override
   public String toString()
   {
      return "Patron[nom:" + this.getNom() + ", prénom:" + this.getPrenom() + ", salaire:"
            + this.gains() + "]";
   }
}




/*
 * Fichier: TravailleurCommission.java
 * Crée le: 30 janvier 2007.
 * Auteurs: Sébastien ESTIENNE.
 * SiteWeb: http://www.prog-info.org/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 */

package chapitre8.employe;



/**
 * <p>Classe représentant un travailleur à la commision.</p>
 * @author Sébastien ESTIENNE.
 */
public class TravailleurCommission extends Employe
{
   // Le salaire mensuel de base.
   protected double salaire;

   // Le montant de la commission par article vendus.
   protected double commission;

   // Le nombre d'articles vendus par mois.
   protected int quantite;


   /**
    * <p>Constructeur par défaut.</p>
    */
   public TravailleurCommission()
   {
      this("", "", 0.0, 0.0);
   }


   /**
    * <p>Constructeur de travailleur à la commission avec un nom, un prénom, un salaire et la
    * commision spécifiés.</p>
    * @param nom Le nom du travailleur à la commission.
    * @param prenom Le prénom du travailleur à la commission.
    * @param salaire Le salaire du travailleur à la commission.
    * @param commission La commission du travailleur à la commission.
    */
```

```java
TravailleurCommission(String nom, String prenom, double salaire, double commission)
{
   super(nom, prenom);
   this.setSalaire(salaire);
   this.setCommission(commission);
}


/**
 * <p>Calcule le salaire de l'employé par mois.</p>
 * @return Retourne le salaire de l'employé par mois.
 */
@Override
public double gains()
{
   return this.getSalaire() + this.getQuantite() * this.getCommission();
}


/**
 * <p>Retourne le montant de la commission par article vendus.</p>
 * @return Renvoie le montant de la commission par article vendus.
 */
public double getCommission()
{
   return this.commission;
}


/**
 * <p>Modifie montant de la commission par article vendus.</p>
 * @param commission Le montant de la commission par article vendus.
 */
public void setCommission(double commission)
{
   if(commission < 0.0)
      this.commission = 0.0;
   else
      this.commission = commission;
}


/**
 * <p>Retourne le nombre d'articles vendus par mois.</p>
 * @return Renvoie le nombre d'articles vendus par mois.
 */
public int getQuantite()
{
   return this.quantite;
}


/**
 * <p>Modifie le nombre d'articles vendus par mois..</p>
 * @param quantite Le nombre d'articles vendus par mois.
 */
public void setQuantite(int quantite)
{
   if(quantite < 0)
      this.quantite = 0;
   else
      this.quantite = quantite;
}


/**
 * <p>Retourne le salaire mensuel de base.</p>
 * @return Renvoie le salaire mensuel de base..
 */
public double getSalaire()
{
   return this.salaire;
}


/**
 * <p>Modifie le salaire mensuel de base.</p>
```

```java
     * @param salaire Le salaire mensuel de base.
     */
    public void setSalaire(double salaire)
    {
        if(salaire < 0.0)
            this.salaire = 0.0;
        else
            this.salaire = salaire;
    }


    /**
     * <p>Retourne une représentation du travailleur à la commission.</p>
     * @return Renvoie une représentation du travailleur à la commission.
     */
    @Override
    public String toString()
    {
        return "TravailleurCommission[nom:" + this.getNom() + ", prénom:" + this.getPrenom()
              + ", salaire:" + this.gains() + "]";
    }
}
```

```java
/*
 * Fichier: TravailleurHoraire.java
 * Crée le: 30 janvier 2007.
 * Auteurs: Sébastien ESTIENNE.
 * SiteWeb: http://www.prog-info.org/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 */

package chapitre8.employe;



/**
 * <p>Classe représentant un travailleur à la commision.</p>
 * @author Sébastien ESTIENNE.
 */
public class TravailleurHoraire extends Employe
{
    // La rétrubution horaire.
    protected double retribution;

    // Le nombre d'heures de travail par mois.
    protected int heures;


    /**
     * <p>Constructeur par défaut.</p>
     */
    public TravailleurHoraire()
    {
        this("", "", 0.0);
    }


    /**
     * <p>Constructeur de travailleur horaire avec un nom, un prénom, une rétribution horaire
     * spécifiés.</p>
```

```java
     * @param nom Le nom du travailleur horaire.
     * @param prenom Le prénom du travailleur horaire.
     * @param retribution La rétrubution horaire.
     */
    TravailleurHoraire(String nom, String prenom, double retribution)
    {
       super(nom, prenom);
       this.setRetribution(retribution);
    }


    /**
     * <p>Calcule le salaire de l'employé par mois.</p>
     * @return Retourne le salaire de l'employé par mois.
     */
    @Override
    public double gains()
    {
       return this.getHeures() * this.getRetribution();
    }


    /**
     * <p>Retourne le nombre d'heures de travail par mois.</p>
     * @return Renvoie le nombre d'heures de travail par mois.
     */
    public int getHeures()
    {
       return this.heures;
    }


    /**
     * <p>Modifie le nombre d'heures de travail par mois.</p>
     * @param heures Le nombre d'heures de travail par mois.
     */
    public void setHeures(int heures)
    {
       if(heures < 0)
          this.heures = 0;
       else
          this.heures = heures;
    }


    /**
     * <p>Retourne la rétrubution horaire.</p>
     * @return Renvoie la rétrubution horaire.
     */
    public double getRetribution()
    {
       return this.retribution;
    }


    /**
     * <p>Modifie la rétrubution horaire.</p>
     * @param retribution La rétrubution horaire.
     */
    public void setRetribution(double retribution)
    {
       if(retribution < 0.0)
          this.retribution = 0.0;
       else
          this.retribution = retribution;
    }


    /**
     * <p>Retourne une représentation du travailleur horaire.</p>
     * @return Renvoie une représentation du travailleur horaire.
     */
    @Override
    public String toString()
    {
       return "TravailleurHoraire[nom:" + this.getNom() + ", prénom:" + this.getPrenom()
             + ", salaire:" + this.gains() + "]";
```

```
   }
}
```

```
/*
 * Fichier: TestEmploye.java
 * Crée le: 30 janvier 2007.
 * Auteurs: Sébastien ESTIENNE.
 * SiteWeb: http://www.prog-info.org/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 */

package chapitre8.employe;



/**
 * <p>Classe de test pour les classes Employe, Patron, TravailleurCommission, TravailleurHoraire.</p>
 * @author Sébastien ESTIENNE.
 */
public class TestEmploye
{
   /**
    * <p>Débute l'exécution du test.</p>
    * @param args Les paramètres de la ligne de commande.
    */
   public static void main(String[] args)
   {
      // Employe.
      Employe emp;

      // Patron.
      Patron p = new Patron("Paul", "MARTIN", 6500);
      emp = p;
      System.out.println(emp);

      // TravailleurCommission.
      TravailleurCommission tc = new TravailleurCommission("Jean", "DUPONT", 3000, 15);
      tc.setQuantite(60);
      emp = tc;
      System.out.println(emp);

      // TravailleurHoraire.
      TravailleurHoraire th = new TravailleurHoraire("Luc", "THOMAS", 12);
      th.setRetribution(150);
      emp = tc;
      System.out.println(th);
   }
}
```

Chaines de caractères :

Jetons

Ecrivez une classe permettant de tester la classe StringTokenizer permettant de découper une chaine de caractères en jetons.
La classe doit afficher le nombre d'éléments et la liste des jetons.
Utilisez les chaines de caractères suivantes:
- "La première chaine à découper en jetons.",
- "La|seconde|chaine|à|découper|en|jetons.".

Cacher la solution.

```
/*
 * Fichier: Jetons.java
 * Crée le: 1 février 2007.
 * Auteurs: Sébastien ESTIENNE.
 * SiteWeb: http://www.prog-info.org/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 */

package chapitre9;

import java.util.StringTokenizer;

/**
 * <p>Classe permettant de tester le découpage de chaines en jetons.</p>
 * @author Sébastien ESTIENNE.
 */
public class Jetons
{
   /**
    * <p>Débute l'exécution du programme.</p>
    * @param args Les paramètres de la ligne de commande.
    */
   public static void main(String[] args)
   {
      // Chaines à découper.
      String chaine1 = "La première chaine à découper en jetons.";
      String chaine2 = "La|seconde|chaine|à|découper|en|jetons.";

      // Jetons.
      StringTokenizer jetons1 = new StringTokenizer(chaine1);
      StringTokenizer jetons2 = new StringTokenizer(chaine2, "|");

      // Chaine 1.
      System.out.println(chaine1);
      System.out.println("Nombre de jetons: " + jetons1.countTokens());
      while(jetons1.hasMoreTokens())
         System.out.println(jetons1.nextToken());

      // Chaine 2.
      System.out.println(chaine2);
      System.out.println("Nombre de jetons: " + jetons2.countTokens());
      while(jetons2.hasMoreTokens())
         System.out.println(jetons2.nextToken());
   }
```

Fonctionnement des exceptions

Ecrivez une classe permettant d'illustrer le fonctionnement du bloc try / catch / finally.
Cette classe contient 2 méthodes, une qui lance une exception et l'autre qui ne lance pas d'exception. Ces deux méthodes contienent un bloc try / catch / finally ne faisant qu'afficher des messages de traces.

Cacher la solution.

```
/*
 * Fichier: FonctionnementException.java
 * Crée le: 23 juin 2007.
 * Auteurs: Sébastien ESTIENNE.
 * SiteWeb: http://www.prog-info.org/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
 */

package chapitre10;



import java.awt.BorderLayout;
import java.awt.Container;

import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;



/**
 * <p>Classe montrant le fonctionnement des exceptions.</p>
 * @author Sébastien ESTIENNE.
 */
public class FonctionnementException extends JFrame
{
   /**
    * <p>Serial version UID.</p>
    */
   private static final long serialVersionUID = 1L;

   // Zone de sortie.
   JTextArea zoneSortie = null;


   /**
    * <p>Construction de l'application.</p>
    */
   public FonctionnementException()
   {
      // Appel du constructeur de la classe JFrame.
      super("Fonctionnement Exceptions");

      // Obtenir le panneau de contenu et changer son layout en un BorderLayout.
      Container conteneur = this.getContentPane();
      conteneur.setLayout(new BorderLayout());

      // Initialisation des composants de l'interface graphique.
      this.zoneSortie = new JTextArea();
      conteneur.add(new JScrollPane(this.zoneSortie));
```

```
        // Test de la méthode lancerException.
        try
        {
            lancerException();
        }
        catch(Exception e)
        {
            this.zoneSortie.append("main: Exception capturée dans le bloc catch.\n");
        }

        // Test de la méthode nePaslancerException.
        this.zoneSortie.append("\n");
        nePasLancerException();

        // Modifie les propriétés de la fenêtre.
        this.setSize(400, 200);
        this.setLocation(100, 100);
        this.setVisible(true);
    }


    /**
     * <p>Méthode permettant d'illustrer le concept try/catch/finally et simulant le lancement d'une
     * exception.</p>
     * @throws Exception L'exception lancée.
     */
    public void lancerException() throws Exception
    {
        // Début de la méthode.
        this.zoneSortie.append("lancerException: Début de la méthode.\n");

        // Lance une exception.
        try
        {
            this.zoneSortie.append("lancerException: Passage dans le bloc try.\n");
            throw new Exception();
        }
        // Capture l'exception et la relance.
        catch(Exception e)
        {
            this.zoneSortie.append("lancerException: Exception capturée dans le bloc catch.\n");
            throw e;
        }
        // Code exécuté qu'une exception soit lancée ou non.
        finally
        {
            this.zoneSortie.append("lancerException: Passage dans le bloc finally.\n");
        }

        // Code inaccessible.
        // this.zoneSortie.append("lancerException: Fin de la méthode.\n");
    }


    /**
     * <p>Méthode permettant d'illustrer le concept try/catch/finally et qui ne lève pas
     * d'exception.</p>
     */
    public void nePasLancerException()
    {
        // Début de la méthode.
        this.zoneSortie.append("nePasLancerException: Début de la méthode.\n");

        // Entrée dans le bloc try.
        try
        {
            this.zoneSortie.append("nePasLancerException: Entrée dans le bloc try.\n");
        }
        // Ce bloc ne s'exécute pas car aucune exception n'est levée.
        catch(Exception e)
        {
            this.zoneSortie.append(e.toString());
        }
        // Code exécuté qu'une exception soit lancée ou non.
        finally
        {
            this.zoneSortie.append("nePasLancerException: Passage dans le bloc finally.\n");
```

```
        }

        // Fin de la méthode.
        this.zoneSortie.append("nePasLancerException: Fin de la méthode.\n");
    }


    /**
     * <p>Lance l'application.</p>
     * @param args Les paramètres de la ligne de commande.
     */
    public static void main(String[] args)
    {
        new FonctionnementException();
    }
}
```