Année universitaire 2014/2015

NOM :..... PRENOM :.....

Consignes relatives au déroulement de l'épreuve

Date : 23 Février 2015

Contrôle de : Module 4ETI de Programmation Objet / Java (2ème session)

Durée: 2 heures

Professeurs responsables : Martine BREDA/Françoise PERRIN

Documents : autorisés ☒

Tout document papier, support de cours ou de TP (livres interdits).

LES CALCULATRICES, TELEPHONES PORTABLES ET AUTRES APPAREILS DE STOCKAGE DE DONNEES NUMERIQUES NE SONT PAS AUTORISES.

Les téléphones portables doivent être éteints pendant toute la durée de l'épreuve et rangés dans les cartables.

Les oreilles des candidats doivent être dégagées.

Rappels importants sur la discipline lors des examens

La présence à tous les examens est strictement obligatoire ; tout élève présent à une épreuve doit rendre une copie, même blanche, portant son nom, son prénom et la nature de l'épreuve.
Une absence non justifiée à un examen invalide automatiquement le module concerné.
Toute suspicion sur la régularité et le caractère équitable d'une épreuve est signalée à la direction des études qui pourra décider l'annulation de l'épreuve; tous les élèves concernés par l'épreuve sont alors convoqués à une épreuve de remplacement à une date fixée par le responsable d'année.
Toute fraude ou tentative de fraude est portée à la connaissance de la direction des études qui pourra réunir le Conseil de Discipline. Les sanctions prises peuvent aller jusqu'à l'exclusion définitive du (des) élève(s) mis en cause.

Consignes générales :

- Lisez SOIGNEUSEMENT tout l'énoncé avant de commencer l'exercice.
- Lisez SOIGNEUSEMENT les annexes.
- Répondre sur ce document.

1° exercice : 4 points

Soit les classes suivantes :

```
class Abeilles {
      Miel [] abeilleMA;
}
class RatonLaveur {
      Kit k;
      Miel rm;
}
class Kit {
      Miel km;
}
class Ours {
      Miel miam;
}
class Miel {
}
public class TestMiel {

      public static void main(String [] args) {
            Miel potDeMiel = new Miel();
            Miel [] ha = {potDeMiel, potDeMiel, potDeMiel, potDeMiel};
            Abeilles a1 = new Abeilles();
            a1.abeilleMA = ha;
            Ours [] oa = new Ours[5];
            for (int x=0; x < 5; x++) {
                  oa[x] = new Ours();
                  oa[x].miam = potDeMiel;
            }
            Kit k = new Kit();
            k.km = potDeMiel;
            RatonLaveur r = new RatonLaveur();
            r.rm = potDeMiel;
            r.k = k;
            k = null;
      }
}
```

1. **Combien d'instances de la classe `Miel` sont créées à la fin de la fonction `main()` ? Expliquez.**

2. **Listez les variables qui référencent l'objet créé par l'instruction**
   `Miel potDeMiel = new Miel();`
   Combien en comptez-vous ?

2° exercice : 6 points

Lire soigneusement le programme donné page suivante.

1. La collection qui stocke les différentes montagnes est une liste (objet qui implémente l'interface List).
   Pourquoi aurait-elle pu être un ensemble (objet qui implémente l'interface Set) **?**

   Pourquoi ne peut-on pas l'instancier comme ci-dessous sans lever une exception ?
   Set<Montagne> set = new TreeSet<Montagne>(mtgn);

2. **Complétez-le code de manière à ce que la trace d'exécution donne le résultat ci-dessous :**

```
Affichage dans l'ordre d'insertion:
[Cervin 4482, MontBlanc 4807, JungFrau 4168, Pelvoux 3955]
Tri par nom :
[Cervin 4482, JungFrau 4168, MontBlanc 4807, Pelvoux 3955]
Tri par hauteur :
[MontBlanc 4807, Cervin 4482, JungFrau 4168, Pelvoux 3955]
```

```
import java.util.Collections;
import java.util.Comparator;
import java.util.LinkedList;
import java.util.List;

public class TriMontagnes {
     public static void main(String [] args) {
          List____________________ mtgn;

          mtgn = new LinkedList____________________();

          mtgn.add(new Montagne("Cervin", 4482));
          mtgn.add(new Montagne("MontBlanc", 4807));
          mtgn.add(new Montagne("JungFrau", 4168));
          mtgn.add(new Montagne("Pelvoux", 3955));
          System.out.println("Affichage par ordre d'insertion :\n" + mtgn);
          ComparateurNoms cn = new ComparateurNoms();

          ____________________________________________________________;

          System.out.println("Tri par nom :\n" + mtgn);
          ComparateurHauteurs ch = new ComparateurHauteurs();

          ____________________________________________________________;

          System.out.println("Tri par hauteur :\n" + mtgn);
     }

     class ComparateurNoms ____________________________________________ {
          public int compare(Montagne un, Montagne deux) {









          }
     }
     class ComparateurHauteurs ____________________________________________ {
          public int compare(Montagne un, Montagne deux) {









     }
}
```

```
class Montagne {
      String nom;
      int hauteur;
// On admet l'existence de getter et setter sur les 2 attributs : INUTILE DE LES ÉCRIRE
//Insérer votre réponse ici
```

3° exercice : 6 points

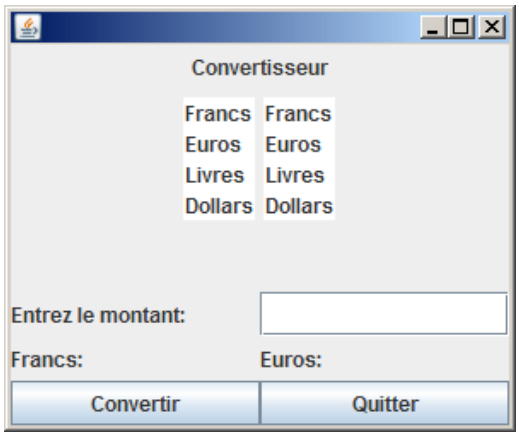
Le programme "convertisseur" permet :

- de saisir un montant,
- de sélectionner la monnaie de départ et la monnaie d'arrivée,
- puis, à la demande, il affiche ce montant dans les 2 monnaies choisies.

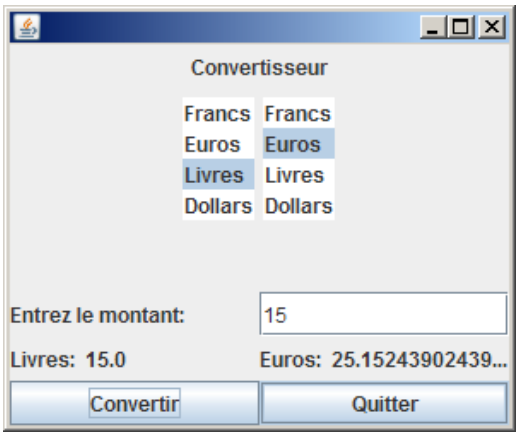
Le tableau `double[] changes = {1,6.56,11,6};` donne les différents taux de conversion utiles.

Exemple de conversion de Livres en Euros :

- La livre apparait en 3ème position :
  le taux de conversion `c1` de Livres en Francs est le 3ème: `c1 =11`
- le taux de conversion `c2` d'Euros en Francs est donné en 2ème position : `c2 = 6.56`
- le taux de conversion `c` de Livres en Euros est le quotient des 2 : `c = c1/c2`.



En début d'exécution



Après 1 exécution

Complétez le code donné page suivante :

```
// On suppose que les "import" nécessaires sont faits.
public class Convertisseur extends JFrame
{
    public static void main(String[] args) {
        Convertisseur maFenetre = new Convertisseur();
        maFenetre.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        maFenetre.setBounds(100,100,300,250);
        maFenetre.setVisible(true);
    }

    JLabel titre, info, monnaie1, monnaie2;
    JButton convertButton, quitterButton;
    Container cont, contTitre, contAppli, change;
    JTextField tf = new JTextField(10);
    String[] moneys = {"Francs", "Euros", "Livres", "Dollars"};
    double[] changes = {1,6.56,11,6.32};
    JList change1, change2;

    public void quitter(){
        System.exit(0);
    }

    private void afficherFenetre()
    {
        contTitre.setLayout(new FlowLayout());
        contTitre.add(titre);
        contAppli.setLayout(new GridLayout(3,2));
        contAppli.add(info);
        contAppli.add(tf);
        contAppli.add(monnaie1);
        contAppli.add(monnaie2);
        contAppli.add(convertButton);
        contAppli.add(quitterButton);
        change.setLayout(new FlowLayout());
        change.add(change1,BorderLayout.WEST);
        change.add(change2,BorderLayout.EAST);
        cont.add(contTitre,BorderLayout.NORTH);
        cont.add(contAppli,BorderLayout.SOUTH);
        cont.add(change,BorderLayout.CENTER);
    }

    public Convertisseur() {
        titre = new JLabel("Convertisseur");
        info = new JLabel("Entrez le montant:");
        monnaie1 = new JLabel("Francs:");
        monnaie2 = new JLabel("Euros:");

        convertButton = new JButton("Convertir");
        quitterButton = new JButton("Quitter");
        cont = getContentPane();
        contTitre = new Container();
        contAppli = new Container();
        change = new Container();
        tf = new JTextField(10);
        change1 = new JList(moneys);
        change2 = new JList(moneys);

        afficherFenetre();
```

//Insérer votre réponse ici

4° exercice : 4 points

On donne les classes Ville et Capitale :

```
public class Ville {
  String nomVille;
  String nomPays;
  int nbreHabitants;
// On admet l'existence des constructeurs, getters et setters.
}

class Capitale extends Ville {
  private String monument;

  public Capitale(){
    super();
    monument = "aucun";
  }
// On admet l'existence des constructeurs, getters et setters.
}
```

Dans un programme de test :

- On déclare le tableau `Ville[] tableau = new Ville[6];`
- On y instancie des villes et des capitales.
- On veut parcourir tout le tableau en affichant le descriptif de chaque ville (`nomVille, nomPays, nbreHabitants` et le nom du monument des capitales).

```
for(Object v : tableau)
{
  System.out.println(v.decrisToi()+"\n");
}
```

1. **Pourquoi ce code ne fonctionne-t-il pas?**

2. **Comment faut-il implémenter la méthode `String decrisToi()`?**

Annexes :

java.lang

Interface Comparable<T>

Modifier and Type	Method and Description
int	**compareTo**(T o) Compares this object with the specified object for order.

java.util

Interface Comparator<T>

Modifier and Type	Method and Description
int	**compare**(T o1, T o2) Compares its two arguments for order.
boolean	**equals**(Object obj) Indicates whether some other object is "equal to" this comparator.

Class Collections

java.lang.Object

java.util.Collections

Modifier and Type	Method and Description
static <T extends Comparable<? super T>> void	**sort**(List<T> list) Sorts the specified list into ascending order, according to the natural ordering of its elements.
static <T> void	**sort**(List<T> list, Comparator<? super T> c) Sorts the specified list according to the order induced by the specified comparator.

Class TreeSet<E>

java.lang.Object

java.util.AbstractCollection<E>

java.util.AbstractSet<E>

java.util.TreeSet<E>

Constructor and Description
TreeSet() Constructs a new, empty tree set, sorted according to the natural ordering of its elements.
TreeSet(Collection<? extends E> c) Constructs a new tree set containing the elements in the specified collection, sorted according to the *natural ordering* of its elements.
TreeSet(Comparator<? super E> comparator) Constructs a new, empty tree set, sorted according to the specified comparator.
TreeSet(SortedSet<E> s) Constructs a new tree set containing the same elements and using the same ordering as the specified sorted set.

javax.swing

Class JList<E>

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JList<E>

int	**getSelectedIndex**() Returns the smallest selected cell index; *the selection* when only a single item is selected in the list.
E	**getSelectedValue**() Returns the value for the smallest selected cell index; *the selected value* when only a single item is selected in the list.

Class JTextField

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.text.JTextComponent
          javax.swing.JTextField

String	**getText**() Returns the text contained in this TextComponent.
void	**setText**(String t) Sets the text of this TextComponent to the specified text.

java.lang

Class Double

java.lang.Object
  java.lang.Number
    java.lang.Double

static double	**parseDouble**(String s) Returns a new double initialized to the value represented by the specified String, as performed by the valueOf method of class Double.

java.awt.event

Interface ActionListener

void	**actionPerformed**(ActionEvent e) Invoked when an action occurs.