Module POA 2nde partie : Exercices

Exercices sur les exceptions, la généricité, le clonage et la sérialisation.

Exo A : Méthodes génériques

- Écrire une méthode générique qui additionne les nombres d'une liste et retourne la somme en format double.

Écrire un programme de test :

```
$ java SommeListeNombre 56.6 7.8 -45
somme = -37.2
```

- Écrire une méthode générique qui trie un tableau d'éléments quelconques (mais comparable).

Écrire un programme de test sur un tableau de mots :

```
$ java Tri voici est un test
arguments tries :
est test un voici
```

L'algoritme de tri bulle est :

```
pour i de tab.length-1 à 0 pas -1 faire
  pour  j de 0 à i-1 pas +1 faire
    si tab[j+1] < tab[j]) alors
      temp = tab[j];
      tab[j] = tab[j+1];
      tab[j+1] = temp;
    finsi
  fait
fait
```

Exo B : Une classe Pile

Fichier Pile1.java

```
1  import java.util.*;
2  public class Pile1 {
3    private ArrayList liste;
4    public Pile1() {
5      liste = new ArrayList();
6    }
7    public Pile1(Collection collection) {
8      // suppose que collection != null)
9      int taille = collection.size();
10       liste = new ArrayList(taille);
11       int i=0;
12       for (Object element : collection)
13         liste.add(i++, element);
14   }
15   public void empiler(Object element) {
16     liste.add(element);
17   }
18   public Object sommet() {
19     int taille = liste.size();
20     if (taille == 0)
21       return null;
22     else
23       return liste.get(taille - 1);
24   }
```

```
25  public Object dépiler() {
26    int taille = liste.size();
27    if (taille == 0)
28      return null;
29    else {
30      Object element = liste.get(taille - 1);
31      liste.remove(taille - 1);
32      return element;
33    }
34  }
35  public boolean estVide(){
36    return (liste.size() == 0);
37  }
38  public String toString() {
39    StringBuffer result = new StringBuffer("[[");
40    int max = liste.size() - 1;
41    for ( int i=0; i < max; i++)
42      result.append(liste.get(i).toString()+",");
43    if (max >= 0)
44      result.append(liste.get(max).toString());
45    return result.toString() + "<-";
46  }
47 }
```

Ce programme est assez mauvais et incomplet.

1. Proposer une meilleure solution Pile2.java qui prenne en compte les exceptions, en particulier lève des EmptyStackException, et qui soit générique pour avoir une pile de même type d'objet.

2. Modifier le programme UsePile1.java pour l'adapter à la meilleure solution Pile2.java

```
$ java UsePile1
q(uitter), a(fficher), s(ommet), e(mpiler) mot, d(epiler), v(ide)
a
[[ ceci,est,une,pile,de,mots <-
s
mots
d
a
[[ ceci,est,une,pile,de <-
e string
a
[[ ceci,est,une,pile,de,string <-
q
```

```
1  import java.util.*;
2  public class UsePile1  {
3    public static void main(String args[]) {
4      Collection collection = Arrays.asList("ceci", "est", "une", "pile", "de", "mots");
5      Pile1 pile = new Pile1(collection);
6      System.out.println("q(uitter), a(fficher), s(ommet), e(mpiler) mot, d(epiler), v(ide)");
7      Scanner sc = new Scanner(System.in);
8      String rep;
9      do {
10       rep = sc.next().trim();
11       if (rep.equals("a"))
12         System.out.println(pile.toString());
13       else if (rep.equals("s") && !estVide(pile)) {
14         String mot = (String) pile.sommet();
15         System.out.println(mot);
16       } else if (rep.equals("v"))
17         System.out.println(pile.estVide()?"vide":"non vide");
18       else if (rep.equals("d") && !estVide(pile))
```

```
19          pile.depiler();
20        else if (rep.equals("e")) {
21          rep = sc.next().trim();
22          pile.empiler(rep);
23          rep = "";
24        }
25      } while (! rep.equals("q")) ;
26    }
27  }
```

3. Comment rendre Pile1 clonable "en profondeur" ? La solution se nommera Pile3.java

4. Comment rendre Pile2<T> sérialisable ? La solution se nommera Pile4.java

5. Rendre Pile4<T> clonable "en profondeur" en utilisant la sérialisation dans une zone mémoire tampon utilisé comme un flot outputStream puis InputStream :

```
ByteArrayOutputStream bout = new ByteArrayOutputStream();
      ....
ByteArrayInputStream bin = new ByteArrayInputStream(bout.toByteArray());
```

Exercices sur les threads

Exo C : Election : Piège à … programmeur

Fichier Vote1.java

```
1  import java.util.*;
2
3  public class Vote1  {
4    public int scoreNationalFraiseDB = 0, scoreNationalNabo = 0;
5    public static void main(String args[]) {
6      new Vote1();
7    }
8    public Vote1() {
9      Thread commElec = new Thread(new CommissionElectorale());
10       commElec.start();
11       Thread bureau;
12       for (int i=0; i<5 ; i++) {
13         bureau = new Thread(new BureauDeVote());
14         bureau.start();
15       }
16   }
17   class BureauDeVote implements Runnable  {
18     public void run() {
19       System.out.println("BureauDeVote thread = "  + Thread.currentThread().getName());
20       // votation
21       try {
22         int scoreFraiseDB = 0, scoreNabo = 0;
23         Thread.sleep((int)(Math.random()*5000));
24         scoreFraiseDB = (int)(Math.random()*1000);
25         scoreNabo = (int)(Math.random()*1000);
```

```
26       // "transmettre" à la commision électorale
27       scoreNationalFraiseDB += scoreFraiseDB ;
28       scoreNationalNabo += scoreNabo ;
29       /* System.out.println("BureauDeVote thread = "  + Thread.currentThread().getName()
30         + " - résultats : scoreFraiseDB = "+ scoreFraiseDB + " scoreNabo = " + scoreNabo); */
31     } catch (InterruptedException ie) {
32       System.out.println("interruption ... thread =" + Thread.currentThread().getName());
33     }
34    }
35  }
36  class CommissionElectorale implements Runnable  {
37    public void run() {
38      System.out.println("CommissionElectorale : attente des résultats ");
39      // attente
40      Scanner sc = new Scanner(System.in);
41      String rep;
42      do {
43        System.out.println("CommissionElectorale : résultats partiels"
44        + " - résultats : scoreNationalFraiseDB = "+ scoreNationalFraiseDB
45        + " scoreNationalNabo = " + scoreNationalNabo + "/n continuer (o/n) ?") ;
46        rep = sc.nextLine().trim();
47      } while (rep.equals("o")) ;
48      // publication
49      System.out.println("CommissionElectorale : résultats finaux "
50        + " - résultats : scoreNationalFraiseDB = "+ scoreNationalFraiseDB
51        + " scoreNationalNabo = " + scoreNationalNabo);
52    }
53  }
54 }
```

Le fichier permet à la commission électorale de rassembler les résultats des différents bureaux de vote et de publier les résultats définitifs.
La solution programmée est une catastrophe :

- les variables partagées peuvent être manipulées conjointement par les différents threads,
- la commission électorale a une boucle d'attente active pour attendre les résultats et n'est même pas sure si c'est réellement fini.

1. Proposer une meilleure solution avec synchronized.

2. Proposer une meilleure solution avec AtomicNumber.

3. Proposer une solution sans attente active avec AtomicNumber et un synchonisateur "barrière"

4. Proposer une solution selon le pattern producteur-consommateur

5. Proposer une solution avec Future, Callable, Executor

Exo D : un widget compte à rebours

Dans l'exemple ci-dessous, on voudrait un décompte de 5 secondes après enfoncement du bouton ARMEMENT.
Le décompte des secondes doit s'afficher.

Au bout du décompte, l'action à faire est déclenchée : ici le message Mise à feu ! Est affichée en console.

Avant le décompte total, il doit être possible d'annuler.

Malheureusement la solution programmée dans le fichier CompteARebours1.java ci-après ne fonctionne pas.
Le thread Event Dispatcheur de traitement graphique est monopolisé pendant les 5 secondes par le décompte !!
Proposer une solution correcte.

```
1  import javax.swing.*;
2  import java.awt.event.*;
3  import java.awt.*;
4
5  public class CompteARebours1 extends JPanel implements ActionListener
6  {
7    private int secondes;
8    private Runnable aFaire;
9    private boolean rebours;
10   private JButton boutonArmer;
11   private JLabel decompte;
12   private JButton boutonArret;
13
14   public CompteARebours1(int secondes, Runnable aFaire) {
15     super(new GridLayout(3,1));
16     this.secondes = secondes;
17     this.aFaire = aFaire;
18     this.rebours = false;
19     boutonArmer = new JButton("ARMEMENT");
20     boutonArmer.addActionListener(this);
21     this.add(boutonArmer);
22     decompte = new JLabel(""+secondes);
23     decompte.setHorizontalTextPosition(SwingConstants.CENTER);
24     this.add(decompte);
25     boutonArret = new JButton("ANNULATION");
26     boutonArret.addActionListener(this);
27     boutonArret.setEnabled(false);
28     this.add(boutonArret);
29   }
30
31   public void actionPerformed(ActionEvent ae) {
32     if (ae.getActionCommand().equals("ARMEMENT")) {
33       this.rebours = true;
34       boutonArret.setEnabled(true);
35       boutonArmer.setEnabled(false);
36       decompte.setForeground(Color.RED);
37       try {
38         for (int i=this.secondes; rebours && (i > 0) ; i--) {
39           Thread.sleep(1000);
40           decompte.setText(""+i);
41         }
42       } catch (InterruptedException ie) {
43       }
44       if (rebours) {
45         decompte.setText("0");
46         boutonArret.setEnabled(false);
47         Thread tacheAFaire = new Thread(aFaire);
48         tacheAFaire.start();
49       }
50     } else if (ae.getActionCommand().equals("ANNULATION")) {
51       boutonArret.setEnabled(false);
52       decompte.setText("ANNULE");
53       rebours = false;
```

```
54    }
55   }
56
57   private static void createAndShowGUI() {
58    JFrame frame = new JFrame("Compte à Rebours");
59    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
60    Runnable afficheMessage = new Runnable() {
61     public void run() {
62      System.out.println("mise a feu !");
63     }
64    };
65    CompteARebours1 newContentPane = new CompteARebours1(5, afficheMessage);
66    newContentPane.setOpaque(true);
67    frame.setContentPane(newContentPane);
68    frame.pack();
69    frame.setVisible(true);
70   }
71
72   public static void main(String[] args) {
73    javax.swing.SwingUtilities.invokeLater(new Runnable() {
74      public void run() {
75       createAndShowGUI();
76      }
77    });
78   }
79  }
```

Exo E : Mise en Cache

La mise en cache permet d'optimiser les temps d'exécution. Par exemple : l'accès à des fichiers, la génération de fichier HTML via PHP, le calcul de fonctions …
Voici un fichier d'interface modélisant les taches à résultat réutilisable :

```
1  public interface TacheAResultatReutilisable<T,R> {
2   /* la tache à résultat réutilisable possède
3     une méthode calculer, coûteuse en temps d'exécution,
4     qui travaille sur des données de type T,
5     fournit des résultats de type R, et est interruptible.
6   */
7   R calculer(T arguments) throws InterruptedException;
8  }
```

Voici une classe implémentant l'interface :

```
1  public class CalculComplique implements TacheAResultatReutilisable<Integer, Integer> {
2   public Integer calculer(Integer x) {
3    int y = x;
4    for (int i=0; i< 100000; i++)
5     for (int j=0; j< 100000; j++)
6      y +=  5;
7    return y;
8   }
9  }
```

Quand le calcul a été fait pour la valeur 1515, il suffit de mémoriser le résultat calculé pour le ressortir s'il est demandé une nouvelle fois.
Voici une première implémentation CalculerEtCacher1.java :

```
1  import java.util.*;
2  public class CalculerEtCacher1<T,R> implements TacheAResultatReutilisable<T,R> {
```

```
3   // @GuardedBy("this")
4   private final Map<T,R> cache = new HashMap<T,R>();
5   private TacheAResultatReutilisable<T,R> maTache = null;
6   public CalculerEtCacher1(TacheAResultatReutilisable<T,R> tache) {
7     this.maTache = tache;
8   }
9   public synchronized R calculer(T arguments) throws InterruptedException {
10     R resultatPrecedent = cache.get(arguments);
11     if (resultatPrecedent != null)
12       return resultatPrecedent;
13     R resultat = maTache.calculer(arguments);
14     cache.put(arguments, resultat);
15     return resultat;
16   }
17  }
```

La solution programmée est une catastrophe :

1. pourquoi ?

2. Proposer une meilleure solution en changeant un peu son code

3. Écrire un programme qui teste la solution sur CalculCompliqué.

Le programme lira les nombres entiers produit au clavier par l'utilisateur tandis que 2 threads "consommeront" ces arguments en calculant selon CalculCompliqué ou en utilisant le cache.

```
$ java TestTachesEtCache1

3 4 5 6 7 8 3 5 9 1

calcul(4) = -1539607548

calcul(3) = -1539607549

calcul(6) = -1539607546

calcul(7) = -1539607545

calcul(8) = -1539607544

calcul(3) = -1539607549

calcul(5) = -1539607547

calcul(9) = -1539607543

calcul(1) = -1539607551

calcul(5) = -1539607547
```

4. Critiquer la meilleure solution ci-dessus au point 2 et donnez les pistes de la solution "définitive".