Module POA 2nde partie : Exercices
sur les exceptions, la généricité, le clonage et la sérialisation.
Didier FERMENT – 2015-16

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
  ```

Exo B : Persistance d'un GUI de choix d'une couleur
Sérialisation et Exception

Complétez le fichier ci-dessous pour la couleur choisie soit persistante d'une exécution à l'autre.
Fichier fourni ChoixCouleur.java en archive zip de projet Eclipse.

```
package df.choixcouleur1;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;
import java.util.*;

public class ChoixCouleur extends JFrame {
	ModeleColor modele;

	public ChoixCouleur(final ModeleColor modelInitial) {
		if (modelInitial == null)
			this.modele = new ModeleColor(255,255,255);
		else
			this.modele = modelInitial;

		this.addWindowListener(new WindowAdapter() {
			public void windowClosing(WindowEvent e) {
					File fichierPersistant = new File(".choixcouleur");
			  // sauvegarder la couleur actuelle dans le fichier .choixcouleur
			}
		});

		Box box = new Box(BoxLayout.Y_AXIS);
		ControleNuancier choix = new ControleNuancier(modele);
```

```
            box.add(choix);
            PanelColor panelColor = new PanelColor(modele.getColor());
            modele.addObserver(panelColor);
            box.add(panelColor);
            Color initiale = modele.getColor();
            PanelRVB panelRVB = new PanelRVB(initiale.getRed(),initiale.getGreen(),
                                             initiale.getBlue());
            modele.addObserver(panelRVB);
            box.add(panelRVB);
            this.getContentPane().add(box);
            this.pack();
            this.setVisible(true);
      }

      public static void main(String[] args) {
            javax.swing.SwingUtilities.invokeLater(new Runnable() {
                  public void run() {
                        File fichierPersistant = new File(".choixcouleur");
                        if ...
                          // récupérer la dernière couleur dans le fichier .choixcouleur
                        } else
                              new ChoixCouleur(null);
                  }
            });
      }
}

class ModeleColor extends Observable  {
      private int rouge, vert, bleu;
      public ModeleColor(int r, int v, int b) {
            rouge = r; vert = v; bleu = b;
      }
      public Color getColor() {
            return new Color(rouge, vert, bleu);
      }
      public void setColor(int r, int v, int b) {
            rouge = r; vert = v; bleu = b;
            this.setChanged();
            this.notifyObservers(new Color(rouge, vert, bleu));
      }
}
class ControleNuancier extends JPanel {
      private ModeleColor modele;
      public ControleNuancier(ModeleColor m) {
            super();
            this.setPreferredSize(new Dimension(100,100));
            this.setLayout(new GridLayout(5,5));
            modele = m;
            for (int i=0; i <25; i++) {
                  JButton nuance = new JButton();
                  nuance.setPreferredSize(new Dimension(20,20));
                  final int rouge = (int)(Math.random()*255);
                  final int vert = (int)(Math.random()*255);
                  final int bleu = (int)(Math.random()*255);
                  nuance.setBackground(new Color(rouge, vert, bleu));
                  this.add(nuance);
                  nuance.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent ae) {
```

```
                              modele.setColor(rouge, vert, bleu);
                        }
                  });
            }
      }
}

class PanelColor extends JPanel implements Observer {
      private Color couleur;
      public PanelColor(Color c) {
            super();
            this.setPreferredSize(new Dimension(100,100));
            couleur = c;
            this.setBackground(couleur);
      }
      public void paint(Graphics g) {
            super.paintComponent(g);
      }
      public void update(Observable o, Object arg) {
            if (arg instanceof Color) {
                  couleur = (Color)arg;
                  setBackground(couleur);
                  this.repaint();
            }
      }
}

class PanelRVB extends JPanel implements Observer {
      private  int rouge, vert, bleu;
      private JTextField fieldR, fieldV, fieldB;
      public PanelRVB(int r, int v, int b) {
            super(new GridLayout(3,2));
            rouge = r; vert = v; bleu = b;
            fieldR = new JTextField(3);
            fieldR.setEditable(false);
            fieldV = new JTextField(3);
            fieldV.setEditable(false);
            fieldB = new JTextField(3);
            fieldB.setEditable(false);
            this.add(new JLabel("R = "));
            this.add(fieldR);
            this.add(new JLabel("V = "));
            this.add(fieldV);
            this.add(new JLabel("B = "));
            this.add(fieldB);
      }
      public void setR(int r) {
            rouge = r;
            fieldR.setText(""+r);
      }
      public void setV(int v) {
            vert = v;
            fieldV.setText(""+v);
      }
      public void setB(int b) {
            bleu = b;
            fieldB.setText(""+b);
      }
```

```
      public void update(Observable o, Object arg) {
            if (arg instanceof Color) {
                  Color couleur = (Color)arg;
                  setR(couleur.getRed());
                  setV(couleur.getGreen());
                  setB(couleur.getBlue());
            }
      }
}
```

Exo C : Une classe Pile
avec des Exceptions, de la généricité, du clonage

Fichier fourni 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<T> 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 fourni UsePile1.java pour obtenir UsePile2.java qui est adapté à la meilleure solution Pile2.java

```
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 }
```

Voici une exécution :

```
$ 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
```

3. Comment rendre Pile1 clonable ? La solution se nommera Pile3.java
   Le programme d'exécution UsePile3.java fourni est enrichi de quelques lignes :

```
…
Pile3 copiePile = null;
System.out.println("n(numero de la pile), q(uitter), a(fficher), s(ommet),
                    e(mpiler) mot, d(epiler), v(ide), c(opier), r(estaurer)");
...
else if (rep.equals("c"))
     copiePile = (Pile3) pile.clone();
else if (rep.equals("r") && (copiePile != null))
```

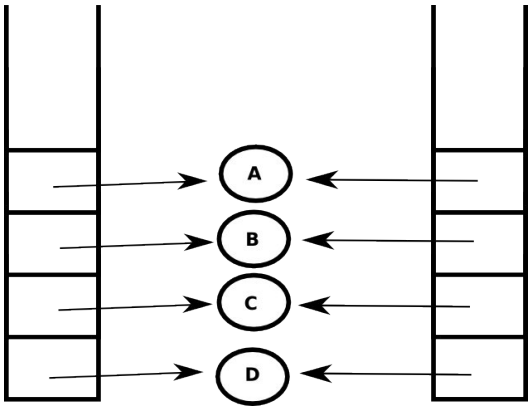
```
            pile = copiePile;
        ...
```

4. Comment rendre Pile2<T> sérialisable ? La solution se nommera Pile4<T>
   Le programme d'exécution UsePile4.java fourni est enrichi de quelques lignes :

```
...
System.out.println("q(uitter), a(fficher), s(ommet), e(mpiler) mot, d(epiler),
                    v(ide), w(rite) fichier, r(ead) fichier");
...
else if (rep.equals("w")) {
     fichier = sc.next().trim();
     File f = new File(fichier);
     ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(f));
     oos.writeObject(pile);
     oos.flush();
     oos.close();
} else if (rep.equals("r")) {
     fichier = sc.next().trim();
     File f = new File(fichier);
     ObjectInputStream ois = new ObjectInputStream(new FileInputStream(f));
     pile = (Pile4<String>) ois.readObject();
     System.out.println(pile.toString());
}
...
```

5. (hors TD) Rendre Pile4<T> clonable "en profondeur" pour éviter qu'après clonage d'une pile comme ci-contre, on puisse modifier le contenu de l'objet B à partir de la pile de droite comme à partir de la pile de gauche.

   La solution se nommera Pile5<T>

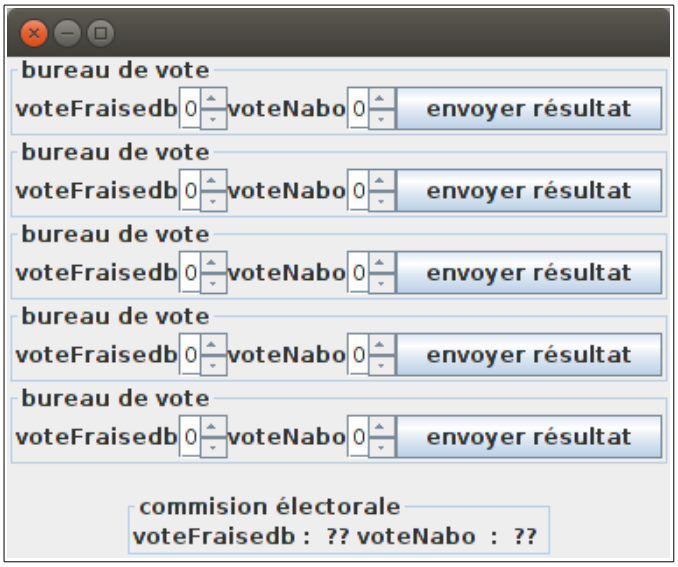


Exercices sur les threads

Exo D : Election : Piège à … programmeur des threads, de la synchronisation, de l'accès en exclusion mutuelle, des taches ….

Le fichier permet à la commission électorale de rassembler les résultats des différents bureaux de vote et de publier les résultats définitifs. Voici le déroulement souhaité :

pour l'élection entre les 2 candidats Fraisedb et Nabo, les bureaux de vote transmettent leur résultats à la commission électorale qui attends tous les résultats partiels pour publier le résultat définitif.



Finalement 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.

Améliorez le projet Eclipse fourni ElectionPiegeAC0
Fichier ElectionPiegeAC0.java fourni en archive zip de projet Eclipse.

```
package df.election0;

import java.awt.BorderLayout;
import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JSpinner;
import javax.swing.SwingUtilities;
import javax.swing.border.TitledBorder;

public class ElectionPiegeAC0 {
	private int scoreNationalFraisedb = 0, scoreNationalNabo = 0;
	private int compte = 0;

	public class CommissionElectorale extends JPanel implements Runnable {
		private JLabel lblVoteFraisedb;
		private JLabel lblVoteNabo;
		private BureauDeVote[] bureauxDeVotes =  null;

		public CommissionElectorale() {
			this.setBorder(new TitledBorder(null, "commision électorale",
					TitledBorder.LEADING, TitledBorder.TOP, null, null));
			this.setLayout(new BoxLayout(this, BoxLayout.X_AXIS));

			lblVoteFraisedb = new JLabel("voteFraisedb :  ?? ");
			this.add(lblVoteFraisedb);
```

```
		lblVoteNabo = new JLabel("voteNabo  :  ?? ");
		this.add(lblVoteNabo);
	}

	public void setVoteFraisedb(final int voteFraisedb) {
		SwingUtilities.invokeLater(new Runnable() {
		    public void run() {
			lblVoteFraisedb.setText("voteFraisedb : "+voteFraisedb+" ");
		    }
		});
	}

	public void setVoteNabo(final int voteNabo) {
		SwingUtilities.invokeLater(new Runnable() {
		    public void run() {
			lblVoteNabo.setText("voteNabo : "+voteNabo+" ");
		    }
		});
	}

	public void setBureauxDeVote(BureauDeVote[] bureauxDeVotes) {
		this.bureauxDeVotes = bureauxDeVotes;
	}

	public void run() {
		// attendre pour publier Resultat
		boolean fini = false;
		do {
			try {
				Thread.sleep(1000);
				if (compte >= 5) {   // attente active !
					setVoteFraisedb(scoreNationalFraisedb);
					setVoteNabo(scoreNationalNabo);
					fini = true;
				}
			} catch (InterruptedException ie) {
				System.out.println("interruption de la commission electorale");
			}
		} while (! fini);

	}
}

public class BureauDeVote extends JPanel  {
	private CommissionElectorale commissionElectorale;
	private JSpinner spinnerVoteFraisedb;
	private JSpinner spinnerVoteNabo;
	private JButton btnEnvoyerResultat;

	public BureauDeVote(CommissionElectorale commissionElectorale)  {
		this.commissionElectorale = commissionElectorale;
		this.setBorder(new TitledBorder(null, "bureau de vote", TitledBorder.LEADING,
							TitledBorder.TOP, null, null));
		this.setLayout(new BoxLayout(this, BoxLayout.X_AXIS));

		JLabel lblVoteFraisedb = new JLabel("voteFraisedb");
		this.add(lblVoteFraisedb);

		spinnerVoteFraisedb = new JSpinner();
		this.add(spinnerVoteFraisedb);

		JLabel lblVoteNabo = new JLabel("voteNabo");
		this.add(lblVoteNabo);
```

```
			spinnerVoteNabo = new JSpinner();
			this.add(spinnerVoteNabo);

			btnEnvoyerResultat = new JButton("envoyer résultat");
			btnEnvoyerResultat.addActionListener(new ActionListener() {
				public void actionPerformed(ActionEvent arg0) {
					spinnerVoteFraisedb.setEnabled(false);
					spinnerVoteNabo.setEnabled(false);
					btnEnvoyerResultat.setEnabled(false);
					// fini donc transmettre Resultat
					scoreNationalFraisedb += getVoteFraisedb() ;
					scoreNationalNabo += getVoteFNabo() ;
					compte++;
				}
			});
			this.add(btnEnvoyerResultat);
		}

		int getVoteFraisedb() {
			return (Integer) spinnerVoteFraisedb.getValue();
		}

		int getVoteFNabo() {
			return (Integer) spinnerVoteNabo.getValue();
		}
	}

	private JFrame frame;

	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					ElectionPiegeAC0 window = new ElectionPiegeAC0();
					window.frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	public ElectionPiegeAC0() {
		initialize();
	}

	private void initialize() {
		frame = new JFrame();
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		JPanel panel = new JPanel();
		frame.getContentPane().add(panel, BorderLayout.CENTER);
		panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));

		CommissionElectorale commissionElectorale = new CommissionElectorale();
		BureauDeVote[] bureauxDeVote = new BureauDeVote[5];
		BureauDeVote BureauDeVote;
		for (int i=0; i<5; ++i) {
		  BureauDeVote = new BureauDeVote(commissionElectorale);
		  panel.add(BureauDeVote);
		  bureauxDeVote[i] = BureauDeVote;
		}

	 panel.add(new JLabel("  "));
```

```
        panel.add(commissionElectorale);
        commissionElectorale.setBureauxDeVote(bureauxDeVote);

        frame.pack();

        Thread threadCommissionElectorale = new Thread(commissionElectorale);
        threadCommissionElectorale.start();
      }

}
```

1. Proposer une meilleure solution avec synchronized mais sans résoudre l'attente active.
2. Proposer une solution sans attente active avec AtomicNumber et un synchonisateur "barrière"
3. (hors TD) Proposer une solution selon le pattern producteur-consommateur

Exo E : un widget compte à rebours
multi-theading et GUI

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 : dans notre exemple c'est seulement 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 fourni 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 F : tache longue asynchrone
Callable, Future

Voici une première solution UsePile1etAsyncTask

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.Socket;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Arrays;
import java.util.Collection;
import java.util.Scanner;

public class UsePile1etAsyncTask  {
  public static void main(String args[]) {
    Collection collection = Arrays.asList("ceci", "est", "une", "pile", "de", "mots");
    Pile1 pile = new Pile1(collection);
    System.out.println("q(uitter), a(fficher), s(ommet), e(mpiler) mot, d(epiler), v(ide), et
c(harger), r(esultat)");
    Scanner sc = new Scanner(System.in);
    String rep;
    StringBuffer tampon = null;
    boolean charger = false;
    do { c
      rep = sc.next().trim();
      if (rep.equals("a"))
        System.out.println(pile.toString());
      else if (rep.equals("s") && !pile.estVide()) {
        String mot = (String) pile.sommet();
        System.out.println(mot);
      } else if (rep.equals("v"))
      System.out.println(pile.estVide()?"vide":"non vide");
      else if (rep.equals("d") && !pile.estVide())
        pile.depiler();
      else if (rep.equals("e")) {
        rep = sc.next().trim();
        pile.empiler(rep);
        rep = "";
      } else if (rep.equals("c")) {
       charger = false;
       Socket soc = null;
        try {
          soc = new Socket("localhost", 7777);
          tampon = new StringBuffer();
          BufferedReader buffReader = new BufferedReader(
            new InputStreamReader(soc.getInputStream()));
          String ligne;
          while ((ligne = buffReader.readLine()) !=  null)
            tampon.append(ligne+"\n");
          charger = true;
        }
        catch (MalformedURLException e) {}
        catch (IOException e) {}
        finally {
          if (soc != null)
          try {
            soc.close();
        } catch (IOException e) {}
        }
```

```
      } else if (rep.equals("r")) {
         if (! charger)
           System.out.println("chargement = rien");
         else {
              System.out.println("chargement = \n"+tampon.toString());
              charger = false;
         }
      }
    } while (! rep.equals("q")) ;
  }
}
```

Voici une première exécution : commencez par lancer le ServiceLent

```
$ java ServiceLent &
$ java UsePile1etAsyncTask
q(uitter), a(fficher), s(ommet), e(mpiler) mot, d(epiler), v(ide), et  c(harger), r(esultat)

a
[[ ceci,est,une,pile,de,mots <-
c
s
mots
r
chargement =
chargement tres lent
chargement tres tres lent
chargement tres tres tres lent
chargement tres tres tres tres lent
chargement tres tres tres tres tres lent
chargement tres tres tres tres tres tres lent
chargement tres tres tres tres tres tres tres lent
chargement tres tres tres tres tres tres tres tres lent
chargement tres tres tres tres tres tres tres tres tres lent
chargement tres tres tres tres tres tres tres tres tres tres lent
a
[[ ceci,est,une,pile,de,mots <-
r
chargement = rien
```

Cette première solution n'est pas terrible : elle bloque le fonctionnement de la pile tant que le chargement lent n'est pas terminé.
Écrire une solution où la commande "c" lance le chargement et la commande "r" donne le résultat si le chargement est terminé correctement.

Exo G : Mise en Cache
Tache à exécuter, Executor, Future

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 …
Le serveur aura à effectuer des "calculs compliqués" prenant beaucoup de temps d'exécution.
Quand le calcul a été fait pour la valeur 1515, il est préférable de mémoriser le résultat calculé pour le ressortir s'il est demandé une nouvelle fois.
Voici une première solution CacheEtCalcul1

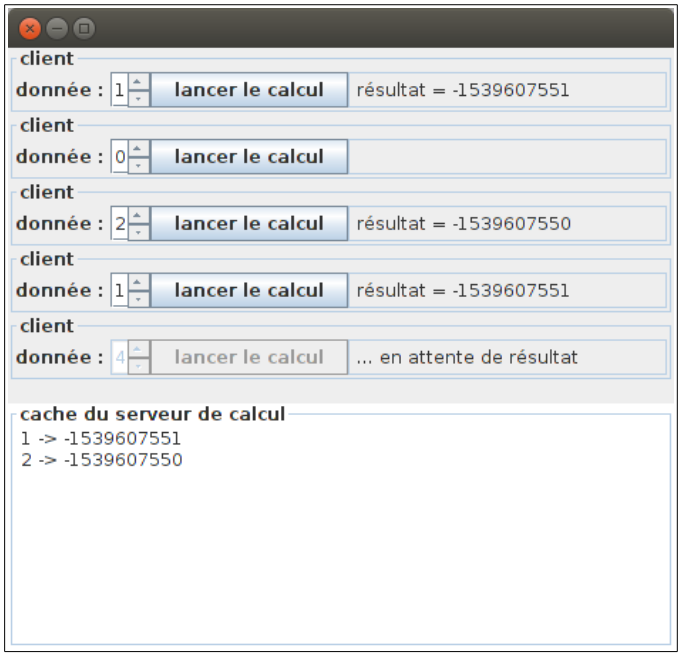
```
package df.cache1;
```

```
import java.util.HashMap;
import java.util.Map;

public class CacheEtCalcul1  {
  // @GuardedBy("this")
  private final Map<Integer,Integer> cache = new HashMap<Integer,Integer>();
  public synchronized Integer cacherEtCalculer(Integer donnee) throws InterruptedException {
         Integer resultatPrecedent = cache.get(donnee);
    if (resultatPrecedent != null)
      return resultatPrecedent;
    Integer resultat = faireCalculComplique(donnee);
    cache.put(donnee, resultat);
    return resultat;
  }

  public Integer faireCalculComplique(Integer x) throws InterruptedException {
          int y = x;
          for (int i=0; i< 100000; i++)
            for (int j=0; j< 100000; j++)
              y +=  5;
          Thread.sleep((int)(5000*(1+Math.random())));
          return y;
        }
}
```

1. La solution programmée de CacheEtCalcul1 est une catastrophe : pourquoi ?
Essayer l'implémentation "graphique" fournie dans un projet Eclipse Cache1Gui.
Ecrire une meilleure solution CacheEtCalcul2.




```
package df.cache1;

import java.awt.BorderLayout;
import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JSpinner;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.SwingUtilities;
import javax.swing.border.TitledBorder;

public class Cache1Gui {

	private JFrame frame;

	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Cache1Gui window = new Cache1Gui();
					window.frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
```

```
	});
}

public Cache1Gui() {
	initialize();
}

private void initialize() {
	frame = new JFrame();
	frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	JPanel panel = new JPanel();
	frame.getContentPane().add(panel, BorderLayout.CENTER);
	panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
	CacheEtCalcul1Gui serveurCalculateur = new CacheEtCalcul1Gui(new CacheEtCalcul1());
	Client[] clients = new Client[5];
	Client Client;
	for (int i=0; i<5; ++i) {
		Client = new Client(serveurCalculateur);
		panel.add(Client);
		clients[i] = Client;
	}
	panel.add(new JLabel(" "));
	panel.add(serveurCalculateur);
	frame.pack();
}

public class Client extends JPanel  {
	private CacheEtCalcul1Gui serveurCalculateur;
	private JSpinner donnee;
	private JButton btnCalculer;
	private JTextField labelResultat;

	public Client(CacheEtCalcul1Gui serveurCalculateur)  {
		this.serveurCalculateur = serveurCalculateur;
		this.setBorder(new TitledBorder(null, "client", TitledBorder.LEADING,
                                                TitledBorder.TOP, null, null));
		this.setLayout(new BoxLayout(this, BoxLayout.X_AXIS));

		this.add(new JLabel("donnée : "));
		donnee = new JSpinner();
		this.add(donnee);

		btnCalculer = new JButton("lancer le calcul");
		btnCalculer.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent arg0) {
				donnee.setEnabled(false);
				btnCalculer.setEnabled(false);
				labelResultat.setText(" ... en attente de résultat ");
				Thread threadCalcul = new Thread(new Runnable() {
				   public void run() {
				       Client.this.serveurCalculateur.lancerCalcul(
                                        (int)donnee.getValue(), Client.this);
				   }
				});
				threadCalcul.start();
			}
		});
		this.add(btnCalculer);
		labelResultat = new JTextField (20);
		labelResultat.setEditable(false);
		this.add(labelResultat);
	}

	public void setResultat(final int result) {
```

```
                SwingUtilities.invokeLater(new Runnable() {
                        public void run() {
                                labelResultat.setText(" résultat = "+result);
                                donnee.setEnabled(true);
                                btnCalculer.setEnabled(true);
                        }
                });
        }
    }

    public class CacheEtCalcul1Gui extends JTextArea {
        private CacheEtCalcul1 calculateur = null;

        public CacheEtCalcul1Gui(CacheEtCalcul1 calculateur) {
                this.setRows(10);
                this.calculateur = calculateur;
                this.setBorder(new TitledBorder(null, "cache du serveur de calcul",
                                        TitledBorder.LEADING, TitledBorder.TOP, null, null));
        }

        public void lancerCalcul(Integer data, Client client) {
                try {
                        Integer resultat = calculateur.cacherEtCalculer(data);
                        client.setResultat(resultat);
                        if (! this.getText().contains(" "+data+" -> "))
                             this.append(" "+data+" -> "+resultat+"\n");
                } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }
    }
}
```

2. Notre 1er serveur lance autant de thread que nécessaire, donc plein de "calcul compliqué" en parallèle qui "épuisent" les processeurs. Limiter le nombre de calculs simultanés du serveur de calcul : 2 calculs en parallèle maximum !
Indications : FixedThreadPool, Future, Callable

3. (hors TD) La solution est encore améliorable : si un calcul compliqué pour la valeur 1515 commence et qu'il n'a jamais encore été effectué, et qu'un second est appelé avant que le premier se termine, alors 2 calculs identiques vont s'effectuer quasi-simultanément !
Indication : mettre le Future dans le cache au lieu du résultat.