STI 4e année
TD: Introspection et Entrées/Sorties

1 Inspection

La premiére étape de ce TD consiste à tester les fonctionnalités d'inspection de Java. Pour ce faire, nous allons utiliser usine.jar (sur Celene) qui va fournir deux méthodes dont vous n'avez pas le code source.

1.1 usine.jar

Exercice 1 Ajoutez usine.jar dans votre projet.

Exercice 2 Dans un fichier Main.java, instanciez un objet de type ObjectProvider du package usine. C'est cet objet qui va ensuite vous permettre de générer d'autres objets dont vous ne connaissez pas le type.

Exercice 3 Affichez les méthodes publiques de la classe ObjectProvider.

Exercice 4 Donnez les prototypes des méthodes nommées **giveMeIntegersAndStrings** et **storeSimilarObjects**.

En effet, La méthode **giveMeIntegersAndStrings** renvoie une collection contenant des Integer et des Strings mélangés. La méthode **storeSimilarObjects** permet de stocker des objets du méme type mais il est interdit de stocker des objets qui ont été retournés par la premiére méthode.

1.2 Inspection

Exercice 5 Récupérez la collection d'objet. Suivant le type de chaque objet Integer ou String, affichez un message différent en console (utiliser le mot clef java ***instanceof***).

On souhaite maintenant dupliquer les objets récupérés dans la collection en créant un objet du meme type que celui qui est récupéré.

Exercice 6 Pour chaque objet récupéré dans la collection, récupérez l'objet Class assocé. Créez alors une instance de cette classe. Si cette classe ne posséde pas de constructeur par défaut sans paramétre, une exception sera levée. Affichez un message en console dans ce cas lors de la récupération de l'exception. Verifiez que cela fonctionne bien pour les String mais qu'une exception est générée pour les Integer.

Exercice 7 Récupérez les méthode disponibles sur chaque objet de la collection. Testez si cet objet posséde la méthode byteValue(). Si c'est le cas affichez un message en console. Verifiez que cela fonctionne bien pour les Integer qui possédent cette méthode, mais pas pour les String.

Solution exercice 7

```
import java.lang.reflect.Method;
import java.util.Collection;
import java.util.Iterator;

import usine.ObjectProvider;
import usine.ObjectProviderInterface;


public class Main {

  public static void main(String[] args) {

    ObjectProviderInterface usine = new ObjectProvider();
    Collection<Object> c = usine.giveMeIntegersAndStrings();

    Iterator<Object> it = c.iterator();
    while (it.hasNext())
    {
      // Inspection de classes
      // ----------------------
      Object o = it.next();
      if (o instanceof Integer)
      {
        System.out.println("C'est un entier: " + (Integer)o);
      }
      if (o instanceof String)
      {
        System.out.println("C'est une string: " + (String)o);
      }

      // Genere une exception: o est deja stocke dans l'usine
      //usine.storeSimilarObjects(o);

      // Instanciation de classes similaires
      // -----------------------------------
      Class classeobjet = o.getClass();
      try {
        Object nouveau = classeobjet.newInstance();
        usine.storeSimilarObjects(nouveau);
      } catch (InstantiationException e) {
        // Pas de constructeur sans paramètre
        // On continue
        System.out.println("Impossible de creer un objet de type " + classeobjet);
      } catch (IllegalAccessException e) {
        e.printStackTrace();
      }

      // Inspection de methodes
      // ----------------------
      Method[] methodes = classeobjet.getMethods();
      boolean trouve = false;
      for (int i=0; i< methodes.length; i++)
      {
        if (methodes[i].getName().equals("byteValue"))
        {
          trouve = true;
        }
      }
      if (trouve == true)
        System.out.println("Cet objet a une méthode byteValue()");
    }
  }
}
```

Solution exercice 7

```
C'est une string: Are you not entertained! Is this not why you are here!
Ajout de l'objet: ok.
C'est un entier: 15
Impossible de creer un objet de type class java.lang.Integer
Cet objet a une methode byteValue()
C'est un entier: 22
Impossible de creer un objet de type class java.lang.Integer
Cet objet a une mÃľthode byteValue()
C'est un entier: 78
Impossible de creer un objet de type class java.lang.Integer
Cet objet a une mÃľthode byteValue()
C'est une string: May the Force be with you.
Ajout de l'objet: ok.
C'est une string: Are you not entertained! Is this not why you are here!
Ajout de l'objet: ok.
C'est un entier: 22
Impossible de creer un objet de type class java.lang.Integer
Cet objet a une mÃľthode byteValue()
```

2 Entrées/Sorties

Dans cette partie, nous allons écrire de différentes maniéres les objets de la collection provenant de ObjectProvider.

2.1 Sortie texte

Exercice 8 Récupérez chaque objet de la collection fournie par giveMeIntegersAndStrings() de l'objet ObjectProvider. Ecrivez sous forme de texte la string correspondant à cet objet dans le fichier output.txt.

Exercice 9 Si l'objet récupéré dans la collection est une string, créez un StringTokenizer : vous allez ainsi pouvoir parcourir chaque mot de la string. Ajoutez alors dans output.txt chaque mot, un par ligne. Vous obtiendrez alors :

```
La phrase de 5 mots.
La
phrase
de
5
mots.
```

2.2 Sortie serialisée

Exercice 10 Serialisez chaque objet dans le fichier output2.bin. Verifiez qu'il n'y a pas de texte lisible dans le fichier ainsi produit.

Solution exercice 10

```
public class Main2 {

    public static void main(String[] args) {
      ObjectProviderInterface usine = new ObjectProvider();
      Collection<Object> c = usine.giveMeIntegersAndStrings();
      FileOutputStream file = null;
      try {file = new FileOutputStream("output.txt");
      } catch (FileNotFoundException e) {
        System.err.println("Erreur de sortie sur output.txt");
        e.printStackTrace();
      }
      BufferedOutputStream out = new BufferedOutputStream(file);
      PrintStream print = new PrintStream(out);
      Iterator<Object> it = c.iterator();
      while (it.hasNext())
      {
        Object o = it.next();
        print.println(o);
        if (o instanceof String) {
          String str = (String)o;
          StringTokenizer s = new StringTokenizer(str);
          while (s.hasMoreTokens())
          {String bout = s.nextToken();
            print.println(bout);
          }
        }

      }
      print.close();
      FileOutputStream file2 = null;
      try {
        file2 = new FileOutputStream("output2.bin");
      } catch (FileNotFoundException e) {
        System.err.println("Erreur de sortie sur output2.bin");
        e.printStackTrace();
      }
      BufferedOutputStream out2 = new BufferedOutputStream(file2);
      ObjectOutputStream print2 = null;
      try {
        print2 = new ObjectOutputStream(out2);
      } catch (IOException e) {
        e.printStackTrace();
      }
      Iterator<Object> it2 = c.iterator();
      while (it2.hasNext())
      {Object o = it2.next();
        try {print2.writeObject(o);} catch (IOException e) {
          e.printStackTrace();}}
      try {print2.close();} catch (IOException e) {
        System.err.println("Error closing file output2.bin");
        e.printStackTrace();
      }
      // reinstancier les objets
      boolean cont = true;
      try {
        BufferedInputStream in2 = new BufferedInputStream(new FileInputStream("output2.bin"));
        ObjectInputStream get2 = new ObjectInputStream(in2);
        while(cont) {Object Oread = get2.readObject();
        if(Oread != null) {
          System.out.println("I have read this object : "+Oread.toString());}
        else {
          cont = false;
        }
        }

      } catch (FileNotFoundException e) {
        // TODO Auto-generated catch block
```

Solution exercice 10

```
Are you not entertained! Is this not why you are here!
Are
you
not
entertained!
Is
this
not
why
you
are
here!
15
22
...
```

Exercice 11 S'il vous reste encore du temps, programmez la lecture et la rï£¡instanciation de ces objets.