Date : Novembre 2020
Enseignant : Marie-Laure Nivet
Diplôme : Licence 3 SFA Informatique
Nom de l'UE : Programmation orientée objet, langage Java
Type de document : Enoncé de TD sur les interfaces et les exceptions

Interfaces et exceptions...

Exercice 1 : Interface et implémentation (version avec exception)

Question 1 : Ecrire le code d'une interface `AStackV2` contenant les méthodes classiques des Piles :

- `isEmpty` qui teste si la pile est vide ;
- `push` qui ajoute un objet au sommet de la pile. (Attention si la pile est pleine, dans le cas d'une pile à contenance finie, il y a levée d'une exception de type `FullStackException` fille de la classe `Exception` que vous devez créer pour l'occasion).
- `peek` qui retourne l'objet du sommet de la pile, sans toute fois l'enlever de la pile (Attention si la pile est vide il y a levée d'une exception de type `EmptyStackException` fille de la classe `RuntimeException` existant dans le JDK).
- `pop` qui retourne et retire l'objet qui est au sommet de la pile (Attention si la pile est vide il y a levée d'une exception de type `EmptyStackException`).

Question 2 : On vous modifier votre classe `ConcreteStackArray` qui implémente cette interface `AStackV2`.
Question 3 : Testez cette classe dans un main en créant une nouvelle pile, empilez quelques éléments de votre choix, dépilez… bref écrivez au moins une fois un appel à toutes les méthodes que vous avez écrites précédemment en traitant comme il se doit les éventuelles exceptions.

Exercice 2 : QCM sur les Exceptions

1- Consider these classes, defined in separate source files:

```
1.   public class Test1 {
2.   public float aMethod(float a, float b)        throws IOException {
3.   }
4.   }
```

```
1. public class Test2 extends Test1 { 2.
3. }
```

Which of the following methods would be legal (individually) at line 2 in class Test2 ?

A. `float aMethod(float a, float b) { }`

B. `public int aMethod(int a, int b) throws Exception{ }`

MCF informatique : Marie-Laure Nivet - ☎ : +33 (0)4 95 45 02 25 - 🖷 : +33 (0)4 95 61 05 51 - 🖃: marie-laure.nivet@univ-corse.fr

C. `public float aMethod(float a, float b) throws _Exception { }`

D. `public float aMethod(float p, float q) { }`

---

2 – Quelle sera la sortie si vous compilez et exécutez ce code, sachant qu'il n'existe pas de fichier nommé Hello.txt dans le répertoire courant ?

```
import java.io.*;
public class Mine {
   public int amethod() {
            try {
                  FileInputStream dis=new FileInputStream("Hello.txt");
            }catch (FileNotFoundException fne) {
              System.out.println("Fichier non trouvé, ");
   return -1;
            }catch(IOException ioe) {
            } finally{
                  System.out.println("Exécution de finally, ");
            }
            return 0;
      }
      public static void main(String argv[]){
            Mine m=new Mine();
            System.out.println(m.amethod());
      }
}
```

A. `Fichier non trouvé,`

B. `Fichier non trouvé, -1`

C. `Fichier non trouvé, Exécution de finally, -1`

D. `0`

---

3 - Soit le code suivant :

```
import java.io.*;

public class Test{
   public static void main(String argv[]){
         Test t = new Test();
   t.amethod();
      }
   public void amethod(){
   try{
                  ioCall();
            }catch(IOException ioe){}
      }
}
```

Quel pourrait être le code de la méthode ioCall ?

A.
```
public void ioCall() throws IOException{
        DataInputStream din = new DataInputStream(System.in);
   din.readChar();
}
```

B.
```
public void ioCall() throw IOException{
        DataInputStream din = new DataInputStream(System.in);
   din.readChar();
}
```

C.
```
public void ioCall(){
        DataInputStream din = new DataInputStream(System.in);
   din.readChar();
}
```

---

4 - Soit le code suivant :

```
class ThreeException extends Exception{}

public class FinallyWorks{
   static int count=0;
   public static void main(String argv[]){
         while(true){            try{
                              if (count++==0) throw new ThreeException();
                              System.out.println("No Exception");
                        }catch(ThreeException e){
                        System.err.println("ThreeException");
                        }finally{
                              System.err.println("In finally clause");
                              if (count == 2) break;
                        }
               }
         }
}
```

Quelle sera la sortie écran ?

---

5 - Cette construction de code est elle correcte ?

```
try {
...
} finally {
...
}
```

6 - Quelles sont les types d'exception qui peuvent être attrapées par ce `catch` ?

```
catch (Exception e) {
...
}
Est-ce judicieux d'écrire ce genre de catch ?
```

7 - Quelles sont les exceptions qui peuvent être interceptées par le code suivant ?

```
} catch (Exception e) {
    ...
} catch (ArithmeticException a) {
    ...
}
Ce code est-il valide ? Va-t-il compiler ?
```

8 - Given the following method body:

```
{  if (atest()) {

      unsafe();

      }else {

      safe();

      }

}
```

La méthode "`unsafe`" peut propager une exception `AWTException` (qui n'est pas une fille de `RunTimeException`). Quelle sera parmi les déclarations suivantes la déclaration pouvant correspondre à la méthode dont le code est donné ci-dessus ?

```
a public AWTException methodName()   b public
void methodName()   c public void methodName()
throw AWTException   d public void methodName()
throws AWTException   e public void methodName()
throws Exception
```

9 - Quel est le résultat de la compilation et exécution du code suivant (une ou plusieurs réponse(s))?

```
import java.io.*;

class MyExp {
      void myMethod() throws IOException, EOFException {
            //............//
```

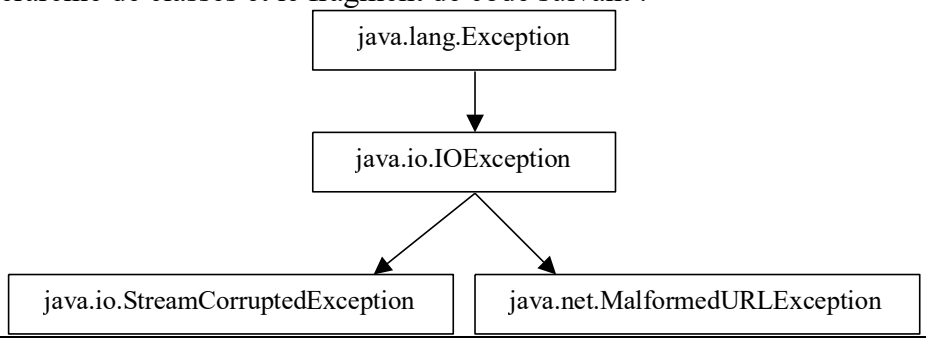
```
      }   }

class MyExp1 extends MyExp {
void myMethod() {
//..........//
       }   }

public class MyExp2 extends MyExp1 {
void myMethod() throws IOException {
             //.........//
        }
}
```

---

a – Erreur de compilation b –
Pas d'erreur de compilation c –
Erreur à l'exécution
d – `MyMethod()` ne peut propager une `IOException` dans `MyExpr2` e – `MyMethod()` doit propager les `IOExeption` et les `EOFException` dans `MyExpr1`

---

10 - Soit la hiérarchie de classes et le fragment de code suivant :




```
1.            try{
2.            // on suppose que s est précédemment défini
3.            URL u = new URL(s);
4.            // in est un ObjectInputStream
5.            Object o = in.readObject();
6.            System.out.println("Réussite");
7.            }
8.            catch(MalformedURLException e){
9.            System.out.println("Mauvaise URL");
10.           }
11.           catch(StreamCorruptedException e){
12.           System.out.println("Contenu de fichier incorrect");
13.           }
14.           catch(Exception e){
15.           System.out.println("Exception générale");
16.           }finally{
17.           System.out.println("Exécution de la partie finally");
18.           }
19.           System.out.println("On continue…");
```

Quels seront le ou les messages affichés en sortie écran si une `MalformedURLException est levée lors de la tentative de construction de l'URL à la ligne 3`

a. `Réussite` b. `Mauvaise URL` c. `Contenu de fichier incorrect`

d. `Exécution de la partie finally` e. `On continue…`

En reprenant le même code et la même hiérarchie de classe que la question précédente, quels seront le ou les messages affichés en sortie écran si tout s'exécute normalement sans lever aucune exception ?

a. `Réussite` b. `Mauvaise URL` c. `Contenu de fichier incorrect`

d. `Exécution de la partie finally` e. `On continue…`

11 - Quel est le résultat de la compilation et exécution du code

```
public class Foo {
    public static void main(String[] args){
try {return;}
        finally {System.out.println( "Finally" );}
}}
```

a – Affichage de « Finally » b – Erreur de compilation

c – Erreur à l'exécution d – Le code s'exécute sans affichage de sortie

12 - Quel est le résultat de la compilation et exécution du code ci-dessous ?

```
public class X {
    public static void main(String [] args){
try{
            badMethod();
System.out.print("A");
        }
        catch (Exception ex){
System.out.print("B");
        }
finally{
            System.out.print("C");
        }
        System.out.print("D");
    }
    public static void badMethod()
    {
        throw new Error(); /* Line 22 */
    }
}
```

a – Affichage de « ABCD » b – Erreur de compilation

c – Affichage de « C » avant sortie sur une erreur d'exécution d – Affichage de « BC » avant sortie sur erreur d'exécution

13 - Quel est le résultat de la compilation et exécution du code ci-dessous ?

```
public class X{
    public static void main(String [] args){
try {
```

```
            badMethod();
            System.out.print("A");
        }catch (RuntimeException ex){ /* Line 10 */
            System.out.print("B");
        }catch (Exception ex1){
            System.out.print("C");
        }finally {
            System.out.print("D");
        }
        System.out.print("E");
    }
    public static void badMethod()
    {
        throw new RuntimeException();
    }  }
```

a – Affichage de « BD »
b – Affichage de « BCD »
c – Affichage de « BDE
d – Affichage de « BCDE »