10 Exceptions

Séance 10
Objectif : exception
53, 54, (55), 56, 58

Rappel : Les exceptions sont un mécanisme de gestion des erreurs. Il existe 3 catégories d'exceptions : les exceptions qui étendent la classe `Exception` qui doivent obligatoirement être gérées par le programme, les exceptions qui étendent la classe `RunTimeException` qui peuvent être gérées par le programme, et les erreurs critiques qui étendent la classe `Error` qui ne sont pas censées être gérées en temps normal.
Toute instance de la classe `Exception` doit obligatoirement être capturée ou bien signalée comme étant propagée par toute méthode susceptible de la lever.

— Pour capturer une exception :

```
try {
    instructions qui peuvent lever une exception
} catch (MonException me) {
    System.out.println(me.toString());
} catch (AutreException ae) {
    System.out.println(ae.getMessage());
} finally {
    instructions toujours executees
}
```

— Pour signaler une erreur, on va lever / lancer une exception, pour cela il faut créer un nouvel objet : `throw new MonException();`
— Pour définir un nouveau type d'exception, il faut écrire une classe qui hérite de la classe `Exception` : `public class MonException extends Exception {...}`
— Pour déléguer / transmettre / propager une exception pour qu'elle soit capturée par une autre méthode : `public void maMethode () throws MonException {...}`

Exercice 53 – Capture dans le main d'une exception prédéfinie (`try catch`)

Q 53.1 Soit classe `TestAttrapePas0` ci-dessous. Que se passe-t-il lors de l'exécution ?

```
public class TestAttrapePas0{
    public static void main(String[] args){
        int [] tab= {1,2,3,4,5};
        for (int i=0; i<15; i++)
            System.out.print(tab[i] + "␣");
    }
}
```

Une exception de dépassement des bornes du tableau n'est pas attrapée par le programmeur et est donc levée directement par java après l'affichage des 5 valeurs du tableau.

```
// EXECUTION :
java TestAttrapePas0
1 2 3 4 5
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
at TestAttrapePas0.main(TestAttrapePas0.java:8)
```

Q 53.2 La méthode `getMessage()` de l'exception `ArrayIndexOutOfBoundsException` retourne la position dans le tableau à laquelle l'erreur s'est produite. Modifier la classe `TestAttrapePas0` pour capturer cette exception et afficher le texte : `"Exception : depassement des bornes a la position 5"` quand l'exception se produit.

```
1 public class TestAttrapePas0 {
2     public static void main(String[] args){
3         int[] tab= {1,2,3,4,5};
4         try {
5             for (int i=0; i<15; i++)
6                 System.out.print(tab[i] + "␣");
7         } catch(ArrayIndexOutOfBoundsException e) {
8             System.out.println("Exception␣:␣depassement␣des␣bornes␣position␣"+e.
                  getMessage());
9         }
10    }
11 }

// EXECUTION /
> java TestAttrapeDepassTab
1 2 3 4 5
Exception : depassement des bornes position 5

Si le try... catch est de part et d'autre du System.out, on aura 10 fois le message
```

Exercice 54 – `Try, catch, throw, throws`, création d'une exception utilisateur

Q 54.1 Écrire une classe `TestAttrapePas1` dans laquelle on définira une méthode de classe `moyenne(String[] tab)` qui, étant donné un tableau de chaînes de caractères représentant des notes (entiers entre 0 et 20), rend la moyenne entière de ces notes. Testez cette méthode dans un main, en affichant la moyenne des notes passées en argument sur la ligne de commande, sans capturer l'exception éventuellement levée.
Indications :

- Utiliser la méthode `Integer.parseInt` qui transforrme une chaîne de caractères en entier et lève une exception `NumberFormatException` si la chaîne n'est pas un entier.
- Les arguments qui sont passés en ligne de commande sont récupérables par le tableau `String[] args` passé en paramètre de la méthode main.

```
1  /** Levee des exceptions predefinies NumberFormatException ou  ArithmeticException,
       qui ne sont pas capturees */
2  public class TestAttrapePas1 {
3    public static int moyenne(String [] tab) {
4      int note,somme=0,n=0;
5      for (int i=0;i<tab.length; i++) {
6        note=Integer.parseInt(tab[i]);   //exception non capturee
7        somme=somme+note;
8        n=n+1;
9      }
10     return (somme/n);     // exception non capturee
11   }
12   public static void main(String[] args) {
13     System.out.println("La␣moyenne␣est␣:␣"+moyenne1(args));
14   }
15 }
```

Q 54.2 Que donnent les exécutions suivantes :

1. javaTestAttrapePas1 10 12 16 18
2. javaTestAttrapePas1 12 1j 10 13 15
3. javaTestAttrapePas1

```
1)  EXECUTION QUAND TOUT VA BIEN :
    java AttrapePas1 10 12 16 18
    La moyenne est : 14
2) EXECUTION QUAND TOUT VA MAL :
    java TestAttrapePas1 12 1j 10 13 15
    Exception in thread "main" java.lang.NumberFormatException:
     For input string: "1j"
       at java.lang.NumberFormatException.forInputString(Unknown Source)
         at java.lang.Integer.parseInt(Unknown Source)
           at java.lang.Integer.parseInt(Unknown Source)
            at TestAttrapePas1.moyenne1(TestAttrapePas1.java:10)
             at TestAttrapePas1.main(TestAttrapePas1.java:18)

3) DEUXIEME EXECUTION QUAND TOUT VA MAL, MAIS AUTREMENT :
ON A OUBLIE DE PASSER LA LISTE DES NOTES
   >java TestAttrapePas1
    Exception in thread "main" java.lang.ArithmeticException: / by zero
      at TestAttrapePas1.moyenne1(TestAttrapePas1.java:14)
         at TestAttrapePas1.main(TestAttrapePas1.java:18)
```

Q 54.3 Dans une classe `TestAttrape2`, reécrire une méthode `moyenne(String[] tab)`qui calcule la moyenne des notes de tab, mais capture cette fois l'exception levée si une note n'est pas entière et la traite en affichant le message « la note n'est pas entière ».

1. Où peut-on attraper l'exception `NumberFormatException` ?
2. Que se passe-t-il si aucune des notes n'est pas entière ou s'il n'y a aucune note ?

1. 1ere version en attrapant l'exception dans le main :

```
1  /**   TestAttrape2.java :  l'exception est levee dans la boucle,
2  * mais attrapee dans le main, ce qui interrompt
3  *  le programme des qu'une note n'est pas entiere  */
4  public class TestAttrape2 {
5    // inchange par raport a la question 1
6    public static int moyenne(String [] liste) {
7      int note,somme=0,n=0;
8      for (int i=0;i<liste.length; i++) {
9        note=Integer.parseInt(liste[i]);  // levee de l'exception
10       somme=somme+note;
11       n=n+1;
12     }
13     return (somme/n);   // exception non capturee
14   }
15   public static void main(String [] args) {
16     try {
17       System.out.println("La moyenne est : "+moyenne(args));
18     } catch(NumberFormatException e) {
```

```
19          System.out.println("la note n'est pas entiere");
20        }
21      }
22    }

EXECUTION :
java TestAttrape2_0 10  12 lm 63 54 pp
la note n'est pas entiere
```

2. 2eme version en attrapant l'exception dans la boucle :

```
1 /* TestAttrape2.java : capture, A L'INTERIEUR de la BOUCLE,
2 * d'une exception predefinie. Le programme n'est ainsi
3 * donc pas interrompue.*/
4 public class TestAttrape2 {
5   public static int moyenne(String [] liste) {
6     int note,somme=0,n=0;
7     for (int i=0;i<liste.length; i++) {
8       try {
9         note=Integer.parseInt(liste[i]);
10        somme=somme+note;
11        n=n+1;
12      } catch(NumberFormatException e) {
13        System.out.println("la "+(i+1)+" eme note n'est pas entiere\textbackslashn
              ");
14      }
15    }
16    return (somme/n);  // exception non capturee
17  }
18  // inchange par rapport a la qestion1
19  public static void main(String [] args) {
20    System.out.println("La moyenne est :"+moyenne1(args));
21  }
22 }

EXECUTION :
  java TestAttrape2 11 lm 16 1e pp 18
  la 2 eme note n'est pas entiere
  la 4 eme note n'est pas entiere
  la 5 eme note n'est pas entiere
  La moyenne est : 15
Execution :
  java TestAttrape2 mm reg 6r c5 mm
  la 1 eme note n'est pas entiere
  la 2 eme note n'est pas entiere
  la 3 eme note n'est pas entiere
  la 4 eme note n'est pas entiere
  la 5 eme note n'est pas entiere
  Exception in thread "main" java.lang.ArithmeticException:  by zero
Il y a levée de l'exception "ArithmeticException" car division par 0
```

Q 54.4 Écrire une classe `AucuneNoteEntiereException` dérivée de la classe `Exception`. Dans une classe `TestAttrape3` réécrire la méthode moyenne qui lancera une instance de la classe `AucuneNoteEntiereException` lorsque ce cas se présentera. Cette exception sera capturée dans le main.

```
1  public class AucuneNoteEntiereException extends Exception {
2    public AucuneNoteEntiereException (String s) {
3      super(s);
4    }
5    public String toString() {
6      return "aucune␣note␣n'est␣valide\n";
7    }
8  }

1 /* Lancement et capture d'une exception creee par l'utilisateur */
2    public class TestAttrape3 {
3      public static int moyenne(String [] liste)
4      throws AucuneNoteEntiereException {
5        int note,somme=0,n=0;
6        for (int i=0;i<liste.length; i++) {
7          try {
8            note=Integer.parseInt(liste[i]);
9            somme=somme+note;
10           n=n+1;
11         } catch(NumberFormatException e) {
12           System.out.println("la␣"+(i+1)+"␣eme␣note␣n'est␣pas␣entiere\n");
13         }
14       }
15       if (n==0) throw new AucuneNoteEntiereException("Lancement␣de␣
             AucuneNoteEntiereException:");
16       return (somme/n);
17     }
18     public static void main(String [] args) {
19       try {
20         System.out.println("La␣moyenne␣est␣:␣"+moyenne(args));
21       }   catch(AucuneNoteEntiereException e) {
22         System.out.println(""+ e.getMessage()+ e);
23       }
24     }
25   }
```

Q 54.5 Que donne l'exécution de la commande `java`TestAttrape3 mm reg 6r c5 mm ?

```
java TestAttrape3 mm reg 6r c5 mm
  la 1 eme note n'est pas entiere
  la 2 eme note n'est pas entiere
  la 3 eme note n'est pas entiere
  la 4 eme note n'est pas entiere
  la 5 eme note n'est pas entiere
Lancement de AucuneNoteEntiereException : aucune note n'est valide
```

Q 54.6 Créer de même une classe `PasEntre0et20Exception` qui servira à traiter les cas où une note serait négative ou strictement supérieure à 20. Où faut-il capturer cette nouvelle exception ? Modifier le programme dans une classe `TestIntervalle` pour qu'il lève et capture aussi cette exception. Que donne l'exécution de la commande `java`TestIntervalle -10 -3 45 -78 -6 21 ?

```
1  /* On capture plusieurs type d'exceptions d'ou
2  * plusieurs blocs catch */
3  public class PasEntre0et20Exception extends Exception {
4    public PasEntre0et20Exception(String s){super(s);}
5  }
6  public class TestIntervalle {
7    public static int moyenne(String [] liste) throws AucuneNoteEntiereException  {
8      int note,somme=0,n=0;
9      for (int i=0;i<liste.length; i++) {
10       try {
11         note=Integer.parseInt(liste[i]);
12         if (note < 0)
13               throw new PasEntre0et20Exception("negative");
14         if (note > 20)
15               throw new PasEntre0et20Exception("superieure a 20");
16         somme=somme+note;
17         n=n+1;
18       } catch(PasEntre0et20Exception e) {
19         System.out.println("la " + (i+1)+" eme note est "+e.getMessage());
20       } catch(NumberFormatException e) {
21         System.out.println("la "+(i+1)+ "eme note n'est pas entiere\n");
22       }
23     }
24     if (n==0)
25       throw new AucuneNoteEntiereException("Lancement de AucuneNoteEntiereException :
             ");
26     return (somme/n);
27   }
28   public static void main(String [] args) {
29     try {
30       System.out.println("La moyenne est : "+moyenne(args));
31     } catch(AucuneNoteEntiereException e) {
32       System.out.println(""+ e.getMessage()+ e);
33     }
34   }
35 }
```

```
java TestIntervalle -10 -3 45 -78 -6 21
la 1 eme note est negative
la 2 eme note est negative
la 3 eme note est superieure a 20
la 4 eme note est negative
la 5 eme note est negative
la 6 eme note est superieure a 20
Lancement de AucuneNoteEntiereException : aucune note n'est valide
```

Exercice 55 – EntierBorne (throw,throws)

Le but de l'exercice est de définir une classe `EntierBorne` qui représente tous les entiers entre -10 000 et +10 000 et se prémunisse des dépassements de ces bornes. On testera au fur et à mesure les méthodes écrites. Note : toutes les exceptions seront capturées dans le main.

Q 55.1 Écrire dans une classe `TestEntierBorne` la méthode `main` qui saisit une valeur entière. On utilisera obli-

gatoirement la méthode `saisirLigne` de la classe `Clavier` non standard qui affiche un message et lit un `String`, puis la méthode `parseInt` de la classe `Integer` (voir la documentation en ligne pour cette méthode) pour transformer la chaîne saisie en entier. Dans le cas où la saisie n'est pas un entier, cette méthode peut lever l'exception `NumberFormatException`.
Que se passe-t-il à l'exécution si la saisie n'est pas entière ? Expliquez.

Une erreur `NumberFormatException` est levée

Q 55.2 Traiter maintenant l'exception levée dans le main. Ajouter les instructions pour que le main s'endorme pendant n secondes en utilisant la méthode `sleep`de la classe `Thread` qui lève une exception de type `InterruptedException`.

```
try {
  n=Integer.parseInt(Clavier.lireLigne("Entrez un nb secondes (entier) : "));
  System.out.println("entier lu : "+n);
  n1000=1000*n;
  System.out.println("debut de l'arret de "+n+" secondes");
  Thread.sleep(n1000);
} catch(NumberFormatException e) {
  System.out.println("la saisie n'est pas un entier");
}catch(InterruptedException e) {
  System.out.println("pb ds sleep(n)");
}
```

Q 55.3 Écrire la classe `EntierBorne` qui est une classe « enveloppe » du type simple int, i.e. qui "enveloppe" une variable d'instance de type int dans un objet de cette classe. Écrire le constructeur à un paramètre de type `int`qui peut lever l'exception `HorsBornesException` si la valeur qui est passée en paramètre est plus grande que 10000 ou plus petite que -10000, et la méthode `toString()`. On définira pour cela la classe `HorsBornesException`.

```
    class HorsBornesException extends Exception {
      public HorsBornesException(String s) {super(s);}
    }
    class EntierBorne { //les bornes pour les el de la classe :
      static int maxEntier=10000,minEntier=-10000;
      int valeur;// la valeur de l'Entier
      public EntierBorne(int i) throws HorsBornesException {
        if (i<minEntier)
           throw new HorsBornesException("Constructeur:tu es trop petit");
        if (i>maxEntier)
           throw new HorsBornesException("Constructeur:tu es trop grand");
        valeur=i;
      }
      public int getValeur() {return valeur;}
      public String toString() { return ""+valeur+" "; }
    }
```

Q 55.4 Définir la méthode `EntierBorne somme(EntierBorne i)` qui rend un objet `EntierBorne` dont la valeur est la somme de cet élément et du paramètre. Elle pourra lever sans la capturer l'exception `HorsBornesException` si la somme est trop grande.

```
1  public EntierBorne somme(EntierBorne i) throws HorsBornesException {
2    // rend un objet EntierBorne somme de cet el et de i
3    int val=this.valeur+i.valeur;
4    if (val<minEntier)
5       throw new HorsBornesException("somme:je suis trop petit ");
6    if (val>maxEntier)
7       throw new HorsBornesException("somme:je suis trop grand ");
8    return new EntierBorne(val);
9  }
```

Q 55.5 Définir la méthode `EntierBorne divPar(EntierBorne i)` qui rend un objet EntierBorne dont la valeur est la division entière de cet élément par le paramètre i. Elle pourra lever l'exception `HorsBornesException` ou l'exception `DivisionParZeroException`.

```
1  public EntierBorne divPar(EntierBorne i) throws ArithmeticException,
2  HorsBornesException {
3    // rend un objet EntierBorne division de cet el par i
4    if (i.valeur == 0) throw new ArithmeticException("division par zero");
5    int val=(int)(this.valeur/i.valeur);
6    return new EntierBorne(val);
7  }
```

Q 55.6 On définira ensuite la méthode `EntierBorne factorielle()` qui calcule la factorielle de cet élément. Elle pourra, en plus de l'exception `HorsBornesException`, lever l'exception `IllegalArgumentException` dans le cas où n serait négatif.

```
1  public EntierBorne factorielle() throws Exception {
2    if (this.getValeur() <0)
3       throw new IllegalArgumentException("x doit etre >= 0");
4    int fact=1;
5    for (int i=2; i<=getValeur(); i++)
6       fact *= i;
7    return new EntierBorne(fact);
8  }
```

Q 55.7 Créer un jeu de tests pour ce programme, en réfléchissant aux différents cas possibles et les tester dans le main.

```
1  public class TestEntierBorne {
2    public static void main(String [] args) {
3      int n=0; // nbbre saisi de secondes
4      int n1000=0; // n en milisecondes
5      int m=0; //
6      EntierBorne eb=null,eb1=null,eb2=null;
7      try {
8        n=Integer.parseInt(Clavier.saisirLigne("Entrez un entier (nb de secondes d'
             arret): "));
9        System.out.println("entier lu : "+n);
```

```
10          n1000=1000*n;
11          System.out.println("debut de l'arret de "+n+" secondes");
12          Thread.sleep(n1000);
13          throw new Exception("Tout va bien");
14        } catch(NumberFormatException e) {
15          System.out.println("la saisie n'est pas un entier");
16        }catch(InterruptedException e) {
17          System.out.println("pb ds sleep(n)");
18        }
19        catch(Exception e) {
20          System.out.println("Fin de l'arret de "+n+" secondes");
21        }
22        System.out.println();
23        System.out.println("TEST SOMME : ");
24        try {
25          n=Integer.parseInt(Clavier.saisirLigne("Entrez un entier : "));
26          System.out.println(" "+n);
27          eb=new EntierBorne(n); // peut lever exception HorsBornes
28          m=Integer.parseInt(Clavier.saisirLigne("Entrez un autre entier : "));
29          System.out.println(" "+m);
30          eb1=new EntierBorne(m); // peut lever exception HorsBornes
31          eb2=eb.somme(eb1); //leve exception HorsBornes si somme trop grande
32          throw new Exception("Tout va bien");
33        } catch(HorsBornesException e) {
34          System.out.println(e.getMessage());
35        } catch(NumberFormatException e) {
36          System.out.println("la saisie n'est pas entiere");
37        } catch(Exception e) {
38          System.out.println("EntierBornes : "+eb+" "+eb1);
39          System.out.println("somme des EntierBorne : "+eb2);
40        }
41        /*-------------------------------------------------*/
42        System.out.println();
43        System.out.println("TEST DIVISION : ");
44        try {
45          n=Integer.parseInt(Clavier.saisirLigne("Entrez un entier : "));
46          System.out.println(" "+n);
47          eb=new EntierBorne(n);
48          m=Integer.parseInt(Clavier.saisirLigne("Entrez un autre entier : "));
49          System.out.println(" "+m);
50          eb1=new EntierBorne(m); // peut lever HorsBornesException
51          eb2=eb.divPar(eb1);
52          throw new Exception("Tout va bien");
53        } catch(NumberFormatException e) {
54          System.out.println("la saisie n'est pas entiere");
55        } catch(HorsBornesException e) {
56          System.out.println(e.getMessage());
57        } catch(ArithmeticException e){
58          System.out.println("Entiers Bornes : " + eb + eb1);
59          System.out.println(e.getMessage());
60        } catch(Exception e){
61          System.out.println("EntierBornes : " + eb + eb1);
62          System.out.println("division des EntierBorne : "+eb2);
63        }
64        //-------------------------------------------------
65        System.out.println();
66        System.out.println("TEST FACTORIELLE : ");
67        try {
68          n=Integer.parseInt(Clavier.saisirLigne("Entrez un entier : "));
```

```
69          System.out.println("␣"+n);
70          eb=new EntierBorne(n); // peut lever exception HorsBornes
71          //leve HorsBornes si factorielle trop grande
72          // ou IllegalArgument si negatif
73          eb2=eb.factorielle();
74          throw new Exception("Tout␣va␣bien");
75        } catch(HorsBornesException e) {
76          System.out.println(e.getMessage());
77        } catch(NumberFormatException e) {
78          System.out.println("la␣saisie␣n'est␣pas␣entiere");
79        } catch(Exception e) {
80          System.out.println("EntierBorne␣:␣"+eb);
81          System.out.println("factorielle␣de␣EntierBorne␣:␣"+eb2);
82        }
83      }
84    }
```

```
EXECUTION 1:
Entrez un entier(nb de secondes d'arret) :
3
3
debut de l'arret de 3 secondes
Fin de l'arret de 3 secondes
TEST SOMME :
Entrez un entier :
5
5
Entrez un autre entier :
10000
10000
somme:je suis trop grand
TEST DIVISION :
Entrez un entier :
5000
5000
Entrez un autre entier :
10
10
EntierBornes : 5000 10
division des EntierBorne : 500
TEST FACTORIELLE :
Entrez un entier :
9
9
Constructeur:tu es trop grand
**********************************
EXECUTION 2 :
Entrez un entier(nb de secondes d'arret) :
2
2
debut de l'arret de 2 secondes
Fin de l'arret de 2 secondes
TEST SOMME :
Entrez un entier :
78
```

```
78
Entrez un autre entier :
-789
-789
EntierBornes : 78  -789
somme des EntierBorne : -711
TEST DIVISION :
Entrez un entier :
987
987
Entrez un autre entier :
0
0
Entiers Bornes : 987 0
division par zero
TEST FACTORIELLE :
Entrez un entier :
6
6
EntierBorne : 6
factorielle de EntierBorne : 720
```

Exercice 56 – throw, throws, finally

Q 56.1 Donnez l'affichage produit par le programme ci-après. Expliquez les résultats.

Cet exercice a pour but d'expliquer le fonctionnement de finally. Regarder l'affichage obtenue. Chaque méthode test() permet de comprendre un peu mieux le fonctionnement des exceptions et de finally.

```
1 public class MonException extends Exception {
2    public MonException(String s) {
3       super(s);
4       System.out.println("\n MonException : constructeur");
5    }
6 }
7
8 public class TestFinally {
9    /** Exception deleguee a la methode appelante (ici main).*/
10   public static void test1() throws MonException {
11      if (true) throw new MonException("lancee dans test1");
12      System.out.println("test1 : fin de la methode");
13   }
14
15 /** Exception capturee (et pas deleguee) dans la methode test2 */
16 public static void test2() {
17   try {
18      if (true) throw new MonException("lancee dans test2");
19   } catch (MonException e) {
20      System.out.println("test2 : capture de l'exception : "+e);
21   }
22   System.out.println("test2 : fin de la methode");
23 }
24
```

```
25 /** Exception capturee (et pas deleguee) dans la methode test3 avec finally */
26 public static void test3() {
27    try {
28       if (true) throw new MonException("lancee dans test3");
29    } catch (MonException e) {
30       System.out.println("test3 : capture de l'exception : "+e);
31    } finally {
32       System.out.println("test3 : finally est effectue");
33    }
34    System.out.println("test3 : fin de la methode");
35 }
36
37 /** Exception deleguee a la methode appelante (ici main) avec finally */
38 public static void test4() throws MonException {
39    try {
40       if (true)
41          throw new MonException("lancee dans test4");
42    } finally {
43       System.out.println("test4 : finally est effectue");
44    }
45    System.out.println("test4 : fin de la methode");
46 }
47
48 /** Meme cas que le test4, mais ici l'exception n'est pas levee */
49 public static void test5() throws MonException {
50    try {
51       if (false) throw new MonException("lancee dans test5");
52    } finally {
53       System.out.println("test5 : finally est effectue");
54    }
55    System.out.println("test5 : fin de la methode");
56 }
57
58 public static void main(String [] args){
59    try {
60       test1();
61    } catch (MonException e) {
62       System.out.println("main : test1 : capture de l'exception "+e);
63    }
64    test2();
65    test3();
66    try {
67       test4();
68    } catch (MonException e) {
69       System.out.println("main : test4 : capture de l'exception "+e);
70    }
71    System.out.println();
72    try {
73       test5();
74    } catch (MonException e) {
75       System.out.println("main : test5 : capture de l'exception "+e);
76    }
77    System.out.println("Fin du programme");
78 }
79 }
```

Attention : finally est TOUJOURS EFFECTUE qu'il y ait une exception ou non (voir test5).

```
Attention : comme le montre le test4,  finally ne capture pas l'exception.

Mais si l'exception est capturée, alors le programme continue avec l'instruction
suivant le finally, sinon le bloc finally est exécutée, puis l'exception est
déléguée à la méthode appelante.


 VOICI L'AFFICHAGE OBTENUE
MonException : constructeur
main : test1 : capture de l'exception MonException: lancee dans test1
MonException : constructeur
test2 : capture de l'exception : MonException: lancee dans test2
test2 : fin de la methode
MonException : constructeur
test3 : capture de l'exception : MonException: lancee dans test3
test3 : finally est effectue
test3 : fin de la methode
MonException : constructeur
test4 : finally est effectue
main : test4 : capture de l'exception MonException: lancee dans test4
test5 : finally est effectue
test5 : fin de la methode
Fin du programme
```

Exercice 57 – MonTableau

Le but de l'exercice est de définir une classe `MonTableau`, gérant des « tableaux » ayant une longueur maximum fixée pour tous les éléments de la classe, et qui se prémunisse des dépassements de capacité de ses objets.

Q 57.1 Définir une classe `MonTableau` qui possède les variables `tab` (tableau d'entiers) et `lgReelle` (entier) donnant le nombre de cases de `tab` réellement utilisées dans le tableau. Au départ, `lgReelle` vaut 0. Ecrire un constructeur prenant en paramètre la taille du tableau, et une méthode `ajouter(int n)` qui ajoute la valeur n à la suite du tableau sans vérifier s'il reste de la place.

```
1 public class MonTableau {
2     private int [] tab;
3     private int lgReelle;
4
5     public MonTableau(int max) {
6         tab=new int[max];
7         lgReelle=0;
8     }
9
10    public void ajouter(int i) {
11        tab[lgReelle]= i;
12        lgReelle++;
13    }
14 }
```

Q 57.2 Ecrire la méthode `main` qui crée un objet `MonTableau` de 3 cases et y ajoute 10 entiers. Exécutez le programme. Que se passe-t-il ?

```
public static void main(String[] args){
    MonTableau t=new MonTableau(3);
    for(int i=1;i<=10;i++) {
        t.ajouter(i);
    }
}

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 3
        at MonTableau.ajouter(MonTableau.java:11)
        at MonTableau.main(MonTableau.java:19)
```

Q 57.3 Capturer dans la méthode `main` l'exception précédemment levée, et afficher le texte `"Depassement des bornes a la position 3"` en utilisant la méthode `getMessage()` de l'exception levée.

```
public static void main(String[] args){
    MonTableau t=new MonTableau(3);
    try {
        for(int i=1;i<=10;i++) {
            t.ajouter(i);
        }
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Depassement des bornes position : "+e.getMessage());
    }
}
```

Q 57.4 Définir un nouveau type d'exception appelée `TabPleinException`.

```
public class TabPleinException extends Exception {
    public TabPleinException(String s) {
        super(s);
    }
}
```

Q 57.5 Modifier la méthode `ajouter` pour lever cette exception quand le tableau est plein. Capturer cette exception dans la méthode `main`. Que retourne les méthodes `getMessage()` et `toString()` de cette exception ?

```
    public void ajouter(int i) throws TabPleinException {
        if (lgReelle < tab.length) {
            tab[lgReelle]= i;
            lgReelle++;
        } else {
            throw new TabPleinException("Le tableau est plein");
        }
    }

public static void main(String[] args){
    MonTableau t=new MonTableau(3);
```

```
12      try {
13          for(int i=1;i<=10;i++) {
14              t.ajouter2(i);
15          }
16      } catch (ArrayIndexOutOfBoundsException e) {
17          System.out.println("Depassement des bornes position : "+e.getMessage());
18      } catch (TabPleinException tpe) {
19          System.out.println("Depassement : "+tpe.getMessage());
20          System.out.println("Depassement : "+tpe.toString());
21      }
22  }
```

La méthode `getMessage()` retourne le texte passée dans le constructeur. La méthode `toString()` de `Exception` retourne NomDeLaClasse+le texte passée dans le constructeur.

Exercice 58 – Extrait de l'examen de 2007-2008 S1

On veut écrire une classe `Etudiant` dont les instances décrivent un étudiant ayant un nom et une liste de notes entières (au maximum 5 notes) implantée par un tableau.
Rappel de cours : toute instance de la classe `Exception` doit obligatoirement être attrapée ou signalée comme étant propagée par toute méthode susceptible de la lever.

Q 58.1 Écrire la classe `Etudiant` correspondant à la description ci-dessus avec un constructeur à un paramètre, le nom. La méthode `toString()` rend le nom de l'étudiant suivi de ses notes.

```
1 public class Etudiant {
2     private String nom;
3     private int [] tabNotes;
4     private int nbNotes;
5     public Etudiant(String n) {
6         nom=n;
7         tabNotes=new int[5];
8         nbNotes=0;
9     }
10    public String toString() { // rend le nom et les notes
11        String s= "";
12        for (int i=0; i<nbNotes; i++) {
13            s += tabNotes[i] + " ";
14        }
15        return nom + " " + s ;
16    }
17 }
```

Q 58.2 Ajouter la méthode `void entrerNote(int note)` qui entre la note dans la liste des notes de cet étudiant. Elle lèvera une exception `TabNotesPleinException` (à définir) dans le cas où le tableau de notes de cet étudiant serait plein. Cette exception sera capturée dans le `main`.

```
1 public class TabNotesPleinException extends Exception {
2     public TabNotesPleinException(String s) {
3         super(s);
4     }
5 }
```

```
1     public void entrerNote(int note) throws TabNotesPleinException {
2         if (nbNotes<tabNotes.length) {
3             tabNotes[nbNotes]=note;
4             nbNotes++;
5         } else {
6             throw new TabNotesPleinException("le␣tableau␣de␣notes␣de␣l'etudiant␣" +
                  this.nom+ "␣est␣plein");
7         }
8     }
```

Q 58.3 En supposant que la classe qui contient le `main` s'appelle `TestEtudiants`, on veut passer sur la ligne de commande une liste d'étudiants avec leurs notes, par exemple :

```
java TestEtudiants Anna 12 13 7 15 Tom Arthur 9 12 15 0 13 12 Karim 15 8 11 12
10 Melissa 12 6 18 10 12 6
```

On supposera que chaque donnée est correcte (pas de mélange entre lettres et chiffres), et que la première donnée est un nom.
Ces données sont de deux types : chaîne de caractères et entier. On va utiliser le fait qu'un entier ne fait pas lever d'exception à la méthode `Integer.parseInt` alors qu'une chaîne de caractères lui fait lever l'exception `NumberFormatException`.
Rappel : la méthode `int Integer.parseInt(String s)` rend l'entier représenté par la chaîne `s`, ou bien lève une exception `NumberFormatException` si la chaîne `s` ne repésente pas un entier.
Ecrire le code du `main` qui récupère les données et affiche pour chacune "c'est une note" ou bien "c'est un nom" suivant le cas. On utilisera obligatoirement le mécanisme d'exception pour ce faire.
Voici une exécution possible :

```
>java TestEtudiants Anna 12 13 7 15 Tom Arthur 9 12 15 0 13 12
Anna c'est un nom,
12 c'est une note, 13 c'est une note, 7 c'est une note, 15 c'est une note,
Tom c'est un nom,
Arthur c'est un nom,
9 c'est une note, 12 c'est une note, 15 c'est une note, 0 c'est une note, 13 c'est une note, 12 c'est une no
```

```
1 public class TestEtudiants {
2     public static void main(String [] args) {
3       int note;
4       // lecture des donnees sur ligne de commande :
5       for (int i=0;i<args.length; i++) {
6         try {
7           note=Integer.parseInt(args[i]);
8           System.out.print(args[i]+"␣c'est␣une␣note,␣");
9         }
10        catch(NumberFormatException e) {
11          System.out.println("\n" + args[i]+"␣c'est␣un␣nom,␣");
12        }
13      }
14    }
15 }
```

Q 58.4 On souhaite gérer dans la classe `Etudiant` une liste au sens `ArrayList` d'étudiants. Une liste d'étudiants ne dépend pas d'un étudiant en particulier. Qu'en concluez-vous sur le type de variables que doit être la liste d'étudiants ? Ajouter les instructions nécessaires dans la classe `Etudiant`.

La liste d'étudiants doit être statique.

```
1 class Etudiant {
2     private static ArrayList<Etudiant> vEtu=new ArrayList<Etudiant>();
3     public Etudiant(String n) {
4         ...
5         vEtu.add(this);
6     }
7     public static int getNbEtudiants() {
8         return vEtu.size();
9     }
10    public static String listeEtudiants() {
11        return vEtu.toString();
12    }
13 }
```

Q 58.5 Enrichir/modifier le code précédent pour qu'il traite les données de la façon suivante :

- si c'est une chaîne de caractères, il crée une nouvelle instance d'étudiant portant ce nom.
- si c'est une note, il ajoute cette note à la liste des notes de l'étudiant créé précédemment, puis affiche la liste des étudiants. On pensera à traiter les différentes exceptions levées (on rappelle qu'un étudiant a au maximum 5 notes).

Voici une exécution possible :

```
>java TestEtudiants Anna 12 13 7 15 Tom Arthur 9 12 15 0 13 12 Karim 15 8 11 12 10
Melissa 12 6 18 10 12 6
le tableau de notes de l'etudiant Arthur est plein
le tableau de notes de l'etudiant Melissa est plein
les 5 etudiants :
[Anna 12 13 7 15, Tom, Arthur 9 12 15 0 13, Karim 15 8 11 12 10, Melissa 12 6 18 10 12 ]
```

```
1  public class TestEtudiants {
2    public static void main(String [] args) {
3      int note;
4      Etudiant eCourant=null;
5      // lecture et stockage des donnees de la ligne de commande :
6      for (int i=0;i<args.length; i++) {
7        try {
8            note=Integer.parseInt(args[i]);
9            eCourant.entrerNote(note);
10       } catch(TabNotesPleinException e) {
11           System.out.println(e.getMessage());
12       } catch(NumberFormatException e) {
13           eCourant = new Etudiant(args[i]);
14       }
15     }
16     //affichage des etudiants :
17     System.out.println("les␣"+ Etudiant.getNbEtudiants() + "etudiants␣:␣\n" +
           Etudiant.listeEtudiants());
18   }
19 }
```