ILLUSTRATION DES EXCEPTIONS

Exemple 1: Capture d'une exception prédéfinie

```
package exception1;

public class PrintArgs {
     public static void main ( String[ ] args ) {
          System.out.println ( " Affichage des paramètres de la
ligne de commande " + " ( il y en a " + args.length + " ) " );

          for(String s: args)
               System.out.println(s);

          try {
               System.out.println ( args [ 0 ] );
          } catch ( ArrayIndexOutOfBoundsException e ) {
               System.out.println ("il n’ y a pas de paramètres.");
          } finally {
               System.out.println ( "Au revoir \n " );
          }
     }
}
```

Exécution:

```
il n’ y a pas de paramètres.
Au revoir
```

Exemple 2: Propagation d'une exception prédéfinie

```
package exception1;

public class PrintArgsBis {
     public static void main ( String[ ] args )
               throws ArrayIndexOutOfBoundsException {
          System.out.println ( " Affichage des paramètres de la
ligne de commande " + " ( il y en a " + args.length + " ) " ) ;

          for(String s: args)
               System.out.println(s);

          System.out.println ( args [ 0 ] ) ;
     }
}
```

Exécution:

```
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 0
     at Exception1.PrintArgsBis.main(PrintArgsBis.java:12)
 Affichage des paramètres de la ligne de commande  ( il y en a 0 )
```

Exemple 2bis: exception prédéfinie non gérée

```
package exception1;

public class PrintArgsBis {
     public static void main ( String[ ] args ) {
          System.out.println ( " Affichage des paramètres de la
ligne de commande " + " ( il y en a " + args.length + " ) " ) ;

          for(String s: args)
               System.out.println(s);

          System.out.println ( args [ 0 ] ) ;
     }
}
```

Exécution:

```
 Affichage des paramètres de la ligne de commande  ( il y en a 0 )
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 0
     at exception1.PrintArgsBis.main(PrintArgsBis.java:11)
```

Exemple 3: RuntimeException

```
package exception1;
```

```
java.lang.Object
  └java.lang.Throwable
      └java.lang.Exception
          └java.lang.RuntimeException
              └java.lang.NegativeArraySizeException
```

```
package exception1;

public class GestionTableau {
      public static void construitTableau ( int a ) {
            int tab [ ] = new int [a]; // NegativeArraySizeException

            for ( int i =0; i < 10; i++) {
                  tab [i] = i;
            }

            for ( int i =0; i < 10; i ++) {
                  System.out.print( tab[i] + " ");
            }
            System.out.println();
      }


      public static void main ( String [ ] args ) {
            int n = -5; // NegativeArraySizeException
            // int n = 8 ArrayIndexOutOfBoundsException
            try {
                  construitTableau( n ) ;
            } catch ( ArrayIndexOutOfBoundsException e) {
                  System.out.println( "indice de tableau hors de la plage
                                          légitime " + e.getMessage());
            } catch ( Exception e ) {
                  e .printStackTrace (System.err);
            } finally {
                  System.out.println( "Au revoir " ) ;
            }
      }
}
```

Exécutions:

Pour n = - 5

```
java.lang.NegativeArraySizeException
      at exception1.GestionTableau1.construitTableau(GestionTableau1.java:5)
      at exception1.GestionTableau1.main(GestionTableau1.java:21)
Au revoir
```

Pour n = 8

```
indice de tableau hors de la plage légitime 8
Au revoir
```

Exemple 4: création de sa propre classe d'exception

```
public class Age0ver extends Exception {
        public Age0ver (String msg) { super (msg); }
}

public class Demo {
    public static void main(String arg[]) {
        DataInputStream s = new DataputStream(System.in);
        do {
            System.out.print("age: ");
            System.out.flush ();
            try {
                String str = s.readLine();
                int age = Integer.parseInt(str);
                if(age> 130)
                    throw new AgeOver(age + " ans trop grand!");
            } catch (IOException ioc) {
                System.out.printin ("erreur d'E/S");
                System.exit(1);
            } catch (AgeOver a) {
                System.out.printin (a.getMessage());
                continue;
            }
            catch (NumberFormatException nbe) {
                System.out.printin (" Il faut saisir une valeur entière!" );
                continue;
            } finally{
                System.out.printin ("Saisie effectuée" );
            }
        }while (true);
    }
}
```

Exemple 5: propagation d'exception

```
package exception1;

public class AgeOver extends Exception {

    private static final long serialVersionUID = 1695368480301843359L;

    public AgeOver(String msg) {
        super(msg);
    }

    @Override
    public String getMessage() {
        return ("Trop vieux" + super.getMessage());
    }
}
```

```
package exception1;

public class AgeLess extends Exception {

      private static final long serialVersionUID = 1695368480301843359L;

    public AgeLess(String msg) {
        super(msg);
    }

    @Override
    public String getMessage() {
        return ("Pas encore né" + super.getMessage());
    }
}
```

```
package exception1;

import java.io.*;

public class Demo {

    public static void main(String arg[])
    {
      try
      {
            saisie();
      }
      catch (IOException ioc) {
            System.out.println(ioc.getMessage() + "\n erreur E/S ");
            System.exit(1);
      }
    }


    public static void saisie() throws IOException {
        BufferedReader s = new BufferedReader(new InputStreamReader(System.in));
        do {
            System.out.print(" age: ");
            System.out.flush();
            try {
                String str = s.readLine();
                int age = Integer.parseInt(str);
                if (age > 130) {
                    throw new AgeOver(age + " ans trop grand ");
                }
                if (age == 0) {
                    throw new AgeLess(" pas né ");
                }
                if (age < 0) {
                    throw new AgeLess("nombre négatif ");
                }
             } catch (AgeOver a) {
                System.out.println(a.getMessage());
                continue;
            } catch (AgeLess a) {
                System.out.println(a.getMessage());
            } catch (NumberFormatException nbe) {
                System.out.println(nbe.getMessage() + " il faut saisir une
valeur entière ! ");
                continue;
            } finally {
                System.out.println(" Saisie effectuée ");
            }
        } while (true);
    }
}
```

Exemple 6: Division par zero

```
public class divisionParZeroException extends Exception
{
}

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

public class divMul
{
        private static int execOperation(int operande1,String operation,int operande2)
                        throws divisionParZeroException, operationInconnueException
        {
                if ("*".equals(operation))
                        return operande1*operande2;
                if ("/".equals(operation))
                {
                        if (operande2 == 0)
                                throw new divisionParZeroException();
                        return operande1/operande2 ;
                }
                else throw new operationInconnueException(operation);
    }
```

```
public static void main (String args[])
{
        System.out.println("exemple de traitement d'exceptions");

        int    operande1[] = {3,  56, 33, 25};
        String operation[] = {"*","/","+","/"};
        int    operande2[] = {5,  4,  5,  0};

        for (int n = 0;n < operande1.length;n++)
        {
                try
                {
                        System.out.println("Calcul de "+ operande1[n] + operation[n] +
                                                                operande2[n]);
                        System.out.println("Resultat"  + execOperation
                                        ( operande1[n], operation[n], operande2[n]));
                }
                catch (divisionParZeroException e)
                { System.out.println("Division par zero."); }
                catch (operationInconnueException e)
                {System.out.println("Operation inconnue " + e.getMessage() ); }
                catch (Exception e)
                { System.out.println("une autre exception est survenue.");}
                 finally
                {  System.out.println("clause finally appellee"); }
        }
   }
}
```

Exemple de l'ouvrage Java in Nutshell de chez O'Reilly

Le meilleur que je connaisse concernant la propagation des exceptions sur la Pile des appels.

```
// Here we define some exception types of our own.
// Exception classes generally have constructors but no data or
// other methods.  All these do is to call their superclass constructors.

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

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

class MySubException extends MyException
{
   public MySubException() { super(); }
   public MySubException(String s) { super(s); }
}

// This class demonstrates defining, throwing and handling exceptions.
// Try invoking it in the following ways and try to understand the
// output:
//    java throwtest
//    java throwtest one
//    java throwtest 0
//    java throwtest 1
```

```
//    java throwtest 99
//    java throwtest 2
//    java throwtest 3

public class throwtest
{
   // This is the main() method.  Note that it uses two
   // catch clauses to handle two standard Java exceptions.
   public static void main(String argv[])
   {
      int i;
      // First, convert our argument to an integer
      // Make sure we have an argument and that it is convertible.
      try
      {
         i = Integer.parseInt(argv[0]);
      }
      catch (ArrayIndexOutOfBoundsException e)
      { // argv is empty
         System.out.println("Must specify an argument");
         return;
      }
      catch (NumberFormatException e)
      { // argv[0] isn't an integer
         System.out.println("Must specify an integer argument.");
         return;
      }

      // Now, pass that integer to method a().
      a(i);
   }
```

```
// This method invokes b(), which is declared to throw
// one type of exception.  We handle that one exception.

public static void a(int i)
{
        try
        {
                b(i);
        }

        // A PROGRAMMER AVEC DEUX CLAUSES catch
        catch (MyException e)
        {                       // Point 1.
        // Here we handle MyException and
        // its subclass MySubException
        if (e instanceof MySubException)
           System.out.print("MySubException: ");
        else
           System.out.print("MyException: ");

        System.out.println(e.getMessage());
        System.out.println("Handled at point 1");
    }
}
```

```
  // This method invokes c(), and handles one of the
  // two exception types that that method can throw.  The other
  // exception type is not handled, and is propagated up
  // and declared in this method's throws clause.
  // This method also has a finally clause to finish up
  // the work of its try clause.  Note that the finally clause
  // is executed after a local catch clause, but before a
  // a containing catch clause or one in an invoking procedure.

  public static void b(int i) throws MyException
        // et à fortiori MySubException
{
    int result;
    try
    {
      System.out.print("i = " + i);
      result = c(i);
      System.out.print(" c(i) = " + result);
    }
    catch (MyOtherException e)
    {                    // Point 2
      // Handle MyOtherException exceptions:
      System.out.println("MyOtherException: " + e.getMessage());
      System.out.println("Handled at point 2");
    }
    finally
    {
      // Terminate the output we printed above with a newline.
      System.out.print("\n");
    }
  }
```

```
    // This method computes a value or throws an exception.
    // The throws clause only lists two exceptions, because
    // one of the exceptions thrown is a subclass of another.

   public static int c(int i) throws MyException, MyOtherException
            // et à fortiori MySubException
{
       switch (i) {
          case 0: // processing resumes at point 1 above
             throw new MyException("input too low");
          case 1: // processing resumes at point 1 above
             throw new MySubException("input still too low");
          case 99:// processing resumes at point 2 above
             throw new MyOtherException("input too high");
          default:
             return i*i;
       }
    }
}
```

Exécution

```
exemple de traitement d'exceptions
Calcul de 3*5
Resultat = 15
clause finally appellee
Calcul de 56/4
Resultat = 14
clause finally appellee
Calcul de 33+5
Operation inconnue : +
clause finally appellee
Calcul de 25/0
Division par zero.
clause finally appellee
```