

Université Sultan Moulay Slimane

Beni-Mellal



École nationale des sciences appliquées

Khouribga



COMPTE RENDU :

TP7 JAVA POO

Filière : Informatique et Ingénierie des données (iid1)

Réalisé par :

- LAHMAMA Fatima-Zahraa
- EL FATHI Zakaria

Encadré par :

- Mr. GHERABI Noredline

EXERCICE 1 :

Gestion des exceptions créées par la division par zéro

1. Pour ce faire, on commence d'abord par créer une classe appelée « *ENTIER* » qui comprend un attribut *A* de type *int*, on ajoute aussi un constructeur qui initialise cet attribut :

Code :

```
package tp7;

public class ENTIER {
    int A;
    // constructeur avec parametre
    public ENTIER(int A) {
        super();
        this.A=A;
    }
}
```

2. Dans la même classe on crée la méthode « division » qui prend en paramètre la variable « diviseur » de type « ENTIER », et qui permet de retourner le résultat, de type double, de la division de « A » par « diviseur » :

Code :

```
double division(ENTIER diviseur) {
    double div

    div=this.A/diviseur.A;
    return div;
}
```

3. Pour tester la méthode de division on passe à une méthode principale pour ce faire et on instancie des objets de type « ENTIER » :

Code :

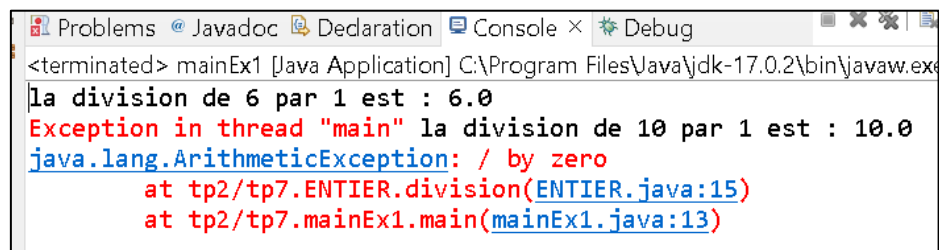
```
package tp7;
public class mainEx1 {
    public static void main(String[] args) {
```

TRAVAUX PRATIQUES 7

– JAVA POO

```
ENTIER n1= new ENTIER(6);
ENTIER n2= new ENTIER(10);
ENTIER d= new ENTIER(1);
System.out.println("la division de "+n1.A +" par "+d.A+" est :
"+(n1.division(d)));
System.out.println("la division de "+n2.A +" par "+d.A+" est :
"+(n2.division(d)));
d= new ENTIER(0);
System.out.println("la division de "+n2.A +" par "+d.A+" est :
"+(n2.division(d)));
}}
```

Exécution :



On remarque que la division par zéro cause un problème au niveau de l'exécution

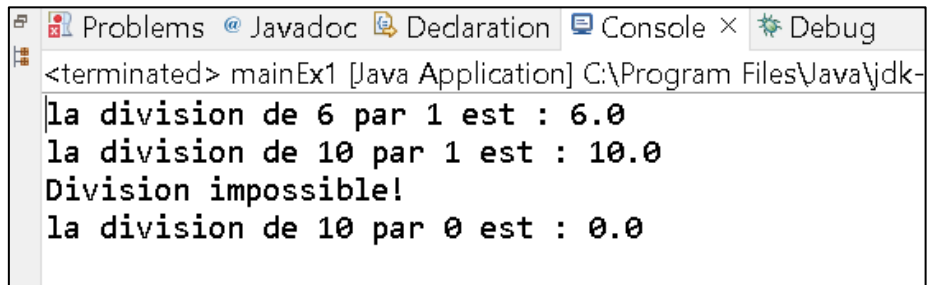
4. Donc pour remédier à ce problème d'exécution on modifie le code de la classe « ENTIER » de telle sorte que si une ArithmeticException est détectée on affiche 'DIVISION IMPOSSIBLE !'

Code :

```
double division(ENTIER diviseur) {
    double div=0;
    try {
        div=this.A/diviseur.A;
    }catch(ArithmeticException e) {
        System.out.println("Division impossible!");
    }
    return div;
}
```

Maintenant revenons a la classe principale pour exécuter le même code que précédemment, on remarque que le problème est bien résolu !

Exécution :



```
<terminated> mainEx1 [Java Application] C:\Program Files\Java\jdk-  
la division de 6 par 1 est : 6.0  
la division de 10 par 1 est : 10.0  
Division impossible!  
la division de 10 par 0 est : 0.0
```

EXERCICE 2 :

1. Un code JAVA qui calcule le factoriel d'un entier.

Dans la classe principale :

```
static public int factoriel(int n) {  
    if(n==0 || n==1) return 1;  
    return (n)*factoriel(n-1);  
}
```

2. On essaie des entrées non entières :

Tout d'abord on crée deux classes de gestion d'exceptions comme suit :

```
package tp7;  
  
public class negativeException extends Exception {  
    public negativeException() {  
        System.out.println("----Vous avez entre un nombre negatif----");  
    }  
}
```

La deuxième est :

```
public class intException extends Exception{  
  
    public intException(int n) {  
        if (n<0) System.out.println("-----nombre inferieur a 0-----");  
    }  
}
```

TRAVAUX PRATIQUES 7

– JAVA POO

Après on procède comme suivant pour mieux gérer les exceptions de cet exercice :

```
public static void main(String[] args) throws negativeException,intException {
    //Q1
    int n;
    do{
        System.out.println("entrer : " );
        Scanner sc=new Scanner(System.in);
        n=sc.nextInt();
    }while(n<0);
    System.out.println("la factoriel : "+ factoriel(n) );

    //Q2 Q3
    try{
        int n;
        do {
            System.out.println("entrer : " );
            Scanner sc=new Scanner(System.in);
            n=sc.nextInt();
            if(n<0) throw new negativeException();
        }while(n < 0);
        System.out.println("la factoriel : "+ factoriel(n) );
    }catch(Exception e1) {
        System.out.println("vous avez entre un Non-entier !" );
        e1.printStackTrace() ;
    }

    // Q4

    System.out.println("entrer : " );
    Scanner sc=new Scanner(System.in);
    n=sc.nextInt();
    if (n<0) throw new negativeException();
    System.out.println("la factoriel : "+ factoriel((int)n) );
}
```

EXERCICE 3 :

on crée deux classes d'exceptions :

```
public class tailleDepassee extends Exception {
```

TRAVAUX PRATIQUES 7
– JAVA POO

```
public tailleDepassee() {  
    System.out.println("valeur maximale depassee");  
}
```

La deuxième est :

```
package tp7;  
  
public class infA1 extends Exception{  
  
    public infA1() {  
        System.out.println("-----entree inferieur a 1-----");  
    }  
}
```

Maintenant on passe à la méthode principale :

Code :

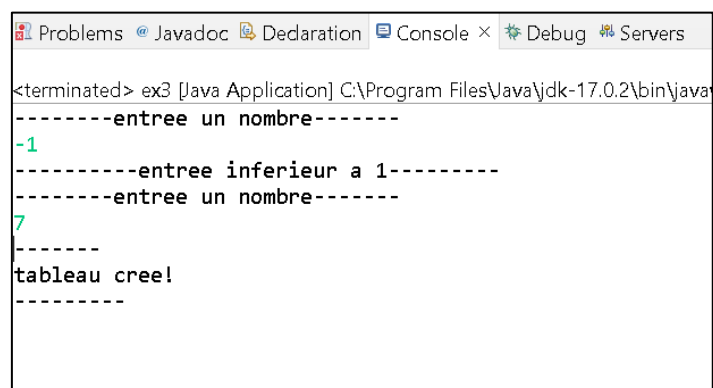
```
package tp7;  
  
import java.util.Scanner;  
  
// Q1  
public class Ex3 {  
    static public int fct(int n) throws infA1, tailleDepassee {  
        System.out.println("-----entree un nombre-----");  
        int a = 0;  
        Scanner scan=new Scanner(System.in);  
  
        a = Integer.parseInt(scan.nextLine());  
        if (a<=1) throw new infA1();  
        else if(a > n) throw new tailleDepassee();  
        return a;  
    }  
  
    static public void tabInitialise(int n) throws nullTableau,infA1,  
tailleDepassee{  
  
        int [] t= new int[fct(n)];  
        System.out.println("-----\ntableau cree!\n-----");  
        if(t==null) throw new nullTableau();  
    }  
}
```

TRAVAUX PRATIQUES 7

– JAVA POO

```
}  
public static void main(String[] args) {  
  
    try {  
        System.out.println("----- L'ENTREE EST :"+fct(20)+"-----  
-");  
    } catch (infA1 | tailleDepassee e) {  
        //e.printStackTrace();  
    }  
  
    // tableau  
    try {  
        tabInitialise(17);  
    } catch (nullTableau | infA1 | tailleDepassee e) {  
        // TODO Auto-generated catch block  
        //e.printStackTrace();  
    }  
  
}
```

Exécution :



```
<terminated> ex3 [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\java  
-----entree un nombre-----  
-1  
-----entree inferieur a 1-----  
-----entree un nombre-----  
7  
-----  
tableau cree!  
-----
```

EXERCICE 4 :

Pour accéder à la base de données Oracle, il faut saisir le login et le mot de passe système. Notre objectif est de gérer les problèmes d'accès en séparant les exceptions pour une meilleure gestion.

1. Un programme qui demande en boucle un nom d'utilisateur (login) et un mot de passe (pwd) jusqu'à recevoir un login/pwd correct.

Code :

```
package tp7;

import java.util.Scanner;

public class ex4Main {

    static public void fLogin() {

        Scanner sc=new Scanner(System.in);

        String user, pwd;
        // entree du login
        System.out.println("-----\nlogin: ");
        user=sc.next();
        System.out.println("-----\npwd: ");
        pwd=sc.next();
        while(!user.equals("scott") && !pwd.equals("tiger")) {

            System.out.println("*****\nlogin incorrect!
\n\n\n entrer a nouveau!\n");

            // entree du login
            System.out.println("-----\nlogin: ");
            user=sc.next();
            System.out.println("-----\npwd: ");
            pwd=sc.next();

        };
        System.out.println("---connection reussie---");

    }

    static public void fLoginAvecException() throws LoginException,
    pwdException,InputLength {

        Scanner sc=new Scanner(System.in);

        String user, pwd;

        do {
            // entree du login
            System.out.println("-----\nlogin: ");
            user=sc.next();
            if (!user.equals("scott")) {
```


TRAVAUX PRATIQUES 7
– JAVA POO

```
        throw new LoginException();
    }

    System.out.println("-----\npwd: ");
    pwd=sc.next();
    if (!pwd.equals("tiger")) {
        throw new pwdException();
    }
    if(user.length()>10 || pwd.length()>10) {
        throw new InputLength();
    }
    if(!user.equals("scott") && !pwd.equals("tiger"))
        System.out.println("-----entrer a nouveau!-----");
    }while(!user.equals("scott") && !pwd.equals("tiger"));
    System.out.println("---connection reussie---");
}

public static void main(String[] args) {
    // TODO Auto-generated method stub

    try {
        fLoginAvecException();
    } catch (LoginException |pwdException|InputLength e) {
        // TODO Auto-generated catch block
        //e.printStackTrace();
    }
}
}
```

2. Implémenter les exceptions suivantes:

- LoginException qui se produit lorsque l'utilisateur saisit un login inexistant
- PwdException lorsque le mot de passe est erroné
- InputLength lorsque le login où le pwd saisi dépasse 10 caractères.

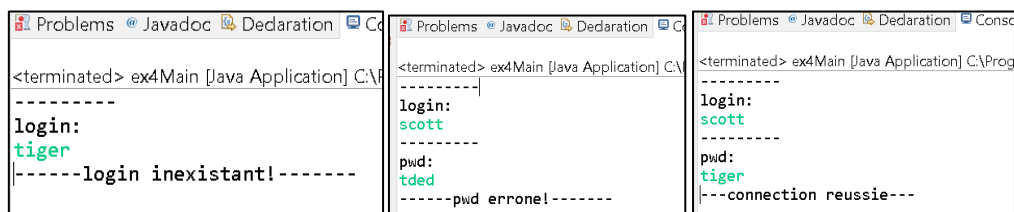
```
public class LoginException extends Exception{
    public LoginException() {
        System.out.println("-----login inexistant!-----");
    }
}
```

TRAVAUX PRATIQUES 7
– JAVA POO

```
public class InputLength extends Exception {  
  
    public InputLength() {  
        System.out.println("-----login ou pwd depasse 10 caracteres!-----");  
    }  
}
```

```
public class pwdException extends Exception{  
  
    public pwdException() {  
        System.out.println("-----pwd errone!-----");  
    }  
}
```

Exécution :



FIN !!