ENSIAS

M.2.5.1. Paradigme Objet

جامعة محمد الخامس بالرباط
Université Mohammed V de Rabat

Classes et Objets (suite)

ENSIAS

Plan de la leçon

جامعة محمد الخامس بالرباط
Université Mohammed V de Rabat

- Méthodes usuelles
- Classes internes
- ~~Ennumération~~
- Exceptions

Méthodes d'accès

Pour contrôler l'accès aux variables

Accesseurs → `getPropriété()`

Accéder en *lecture seule*

Mutateurs → `setPropriété()`

Contrôler la validité de la valeur affectée

Méthodes d'accès

```
public class Personne {
  private int age ; …

  public int getAge(){ return age; }
  public void setAge(int a) throws Exception {
    if (a < 0 || a > 120)
    throw new Exception( "Age invalide ");
    age = a ;
  }
```

```
Age = 130

Age invalide
Age = 130
```

```
  public static void main(String args[]){
    Person p = new Personne();
    p.age = 130 ;
    System.out.println("Age ="+ p.getAge());

    try { p.setAge(140); } catch(Exception e){ //erreur !! }

    System.out.println(" Age = " + p.getAge() );
  }}
```

Méthode toString()

```
public class Personne {
   private int age;            private String nom;
   private boolean marié;      private int nbEnfants;
   /*
   * Conversion d'un objet en String
   */
   public String toString() {
      String s =  nom + " " + age + "ans";
      if (marié) s += "(marié " + nbEnfants + " enfants) " ;
     return s;
   }
}
```

```
Abbad 35 ans (marié 2 enfants)
```

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

     Person person = new Person("Abbad",35,true,2);
     System.out.println(person);
  }
}
```

Méthode equals()

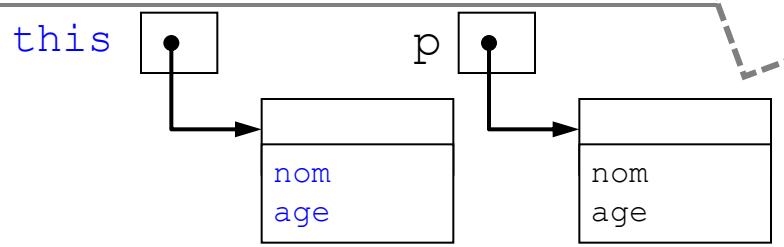
Person.java

```
public class Personne {
  private String nom;  private int age;

   public boolean equals(Personne p) {
       return (
                  nom.equals(p.nom)
                        &&
                  age == p.age
                );
    }
}
```

equals(Personne)

equals(Employé) ?

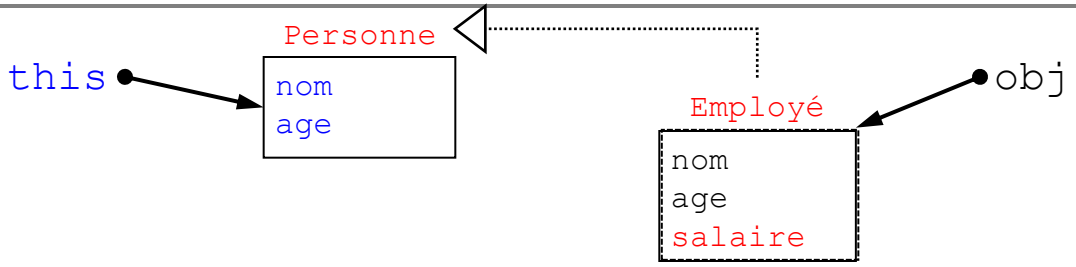
equals(Object)



Méthode equals()

Person.java

```
public class Person {
  private String nom;  private int age;
    public boolean equals(Object obj) {

      if (obj == null) return false;
      if (this == obj) return true;
      if(!(obj instanceof Personne)) return false;

      Personne p = (Personne) obj;
      return (nom.equals(p.nom)   && age == p.age);
    }
}
```



Passage de paramètres

```
public class App{
    public static void raz(int n) {
        n = 0 ; }
    public static void raz(A obj) {
        obj.setValue(0); }

    public static void main (String[] args){
        int i=10;                raz(i);
        System.out.println("i= "+i);
        //
        A a = new A(10) ; raz(a);
        System.out.println("a= "+ a.getValue());
        return;
    }
}
```

type primitif

Par valeur

i = 10

type Objet

Par référence

a = 0

```
class A{
    private int value ;
    public A(int value){ this.value=value;}
    public int getValue(){ return(value); }
    public void setValue(int value){
            this.value=value; }
}
```

ENSIAS

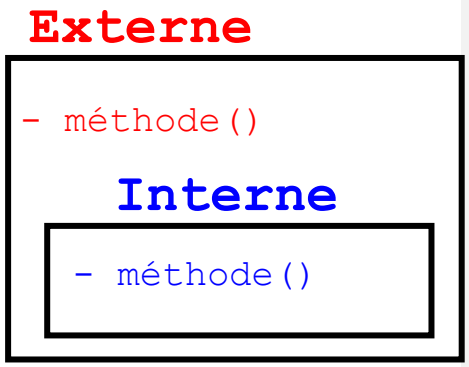
جامعة محمد الخامس بالرباط
Université Mohammed V de Rabat

Classes internes

Classes internes

→ **Classes définies à l'intérieur d'autres classes**

```
public class Externe{
  //variables et méthodes
  class Interne{
    //variables et méthodes
  }
}
```



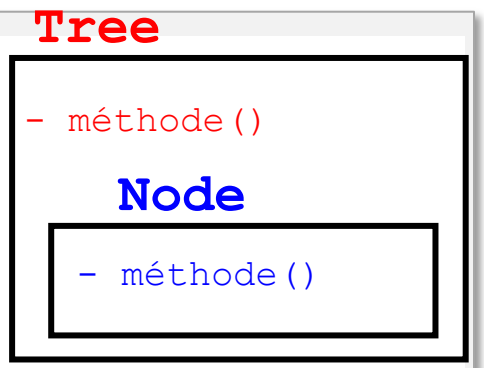
→ **Avantages :**

1. **Plus d'encapsulation**
2. **Les deux classes ont accès chacune aux membres privées de l'autre**

Classes internes

→ Exemples

```
public class Tree{
  private class Node{
    public int data;
    public Node left;
    public Node right;
    //…
  }
  private Node root;
  //…
}
```

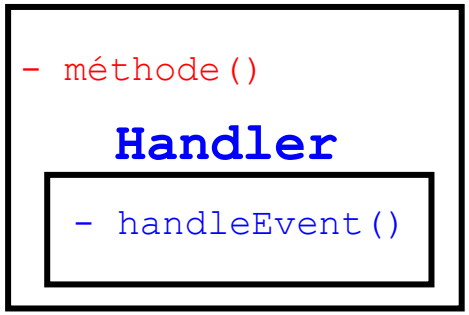


Classes internes

→ **Exemples** Applications de messagerie instantanée

- La classe ChatSession "gère" la session de chat

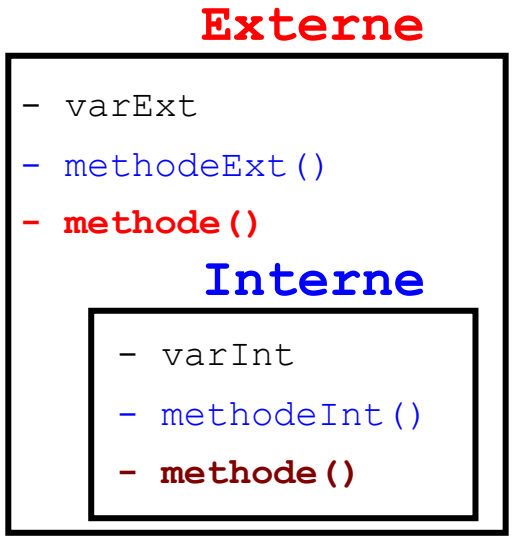
- Des *listeners (handlers)* sont utilisés **uniquement** par la classe ChatSession (à chaque envoi/réception d'un *message)*



Classes internes

Classe interne simple

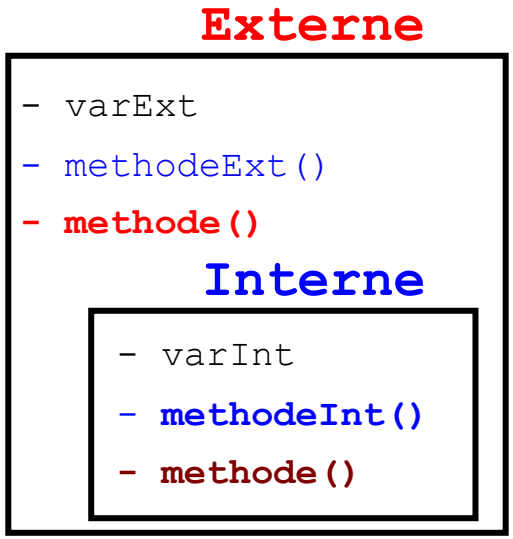
- Ne peut pas contenir des déclarations **statiques** ni de méthode main
- Le compilateur génère deux fichiers indépendants



Externe.class	Externe$Interne.class

Classes internes

Classe interne simple



→ Peut accéder aux attributs (externes)

```
public void methodeInt(){
   system.out.println( methodeExt() );
}
```

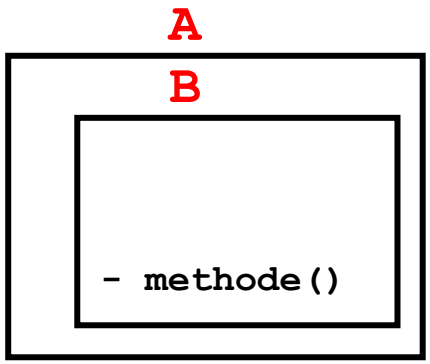
→ Possède une référence sur le **this** de sa classe englobante

```
public void methodeInt(){
   system.out.println(
         Externe.this.methode() );
}
```

«**this** de la classe **Externe** »

Classes internes

Classe interne simple



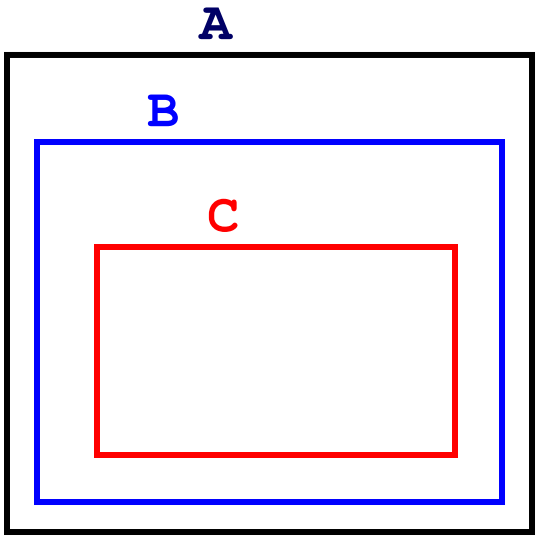

```
public class Accessor {
  public static void main(String... args){
    A.B b = new A().new B();
    b.methode(); //
  }
}
```

– Lors de la construction, une classe interne doit être **construite par un objet de la classe englobante**

Classes internes

→ Les classes peuvent être imbriquées

```
A a = new A() ;
A.B b = a.new B() ;
A.B.C c = b.new C() ;
```




```
A.B.C c = new A().new B().new C() ;
```

ENSIAS

Classes internes

جامعة محمد الخامس بالرباط
Université Mohammed V de Rabat

```
public class ShadowTest {

   public int x = 0;

   class FirstLevel {

      public int x = 1;

      void methodInFirstLevel(int x) {
         System.out.println("x = " + x);
         System.out.println("this.x = " + this.x);
         System.out.println("ShadowTest.this.x = " +
                            ShadowTest.this.x);
         }

   }

   public static void main(String... args) {
      ShadowTest st = new ShadowTest();
      ShadowTest.FirstLevel fl = st.new FirstLevel();
      fl.methodInFirstLevel(23);
   }
}
```

```
x = 23

this.x = 1

ShadowTest.this.x = 0
```

Classes anonymes

→ **Pour créer un objet sans nommer sa classe**

```
public abstract class
   private Point centr
   public int getCente
   public abstract dou
}
```

```
public class Rectangle extends Figure{
   private double long,larg;
   public double surface(){return (long*larg);}
   //…
}
```

```
Rectangle r = new Rectangle(); //"éliminer" Rectangle ???
```

✖ Pas de se

```
Figure r = new Rectangle(); //ou encore :
```

```
Figure r = new Figure(){
   private double long,larg;
   public double surface(){return (long*larg);}
```

Classes internes anonymes

```
public abstract class Operation{
   int a, b ; //visibilité par défaut
   public Operation (int a, int b){
      this.a =a ; this.b = b;
   }
   abstract int eval();
}
public class OperationFactory{
   public static Operation plus(int x, int y) {
      // return (new Operation(x,y)) ; //incorrecte
      return  new Operation(x,y) {
         @Override
         int eval() { return (x+y); }
      };
   }
}
```

Classes internes aux méthodes

```
public class A {
   private int a = 10;
   void methode(final int x) {
      final int y = 10;
      class B {
         public void print(int z) {
            System.out.println("somme= " + (a+x+y+z));
         }
      }

      B b = new B();
      // juste après la définition de la classe
      b.print(5);
   }
}
```

Mécanisme des exceptions

Exceptions : problématique

Traitement des erreurs par test des valeurs de retour

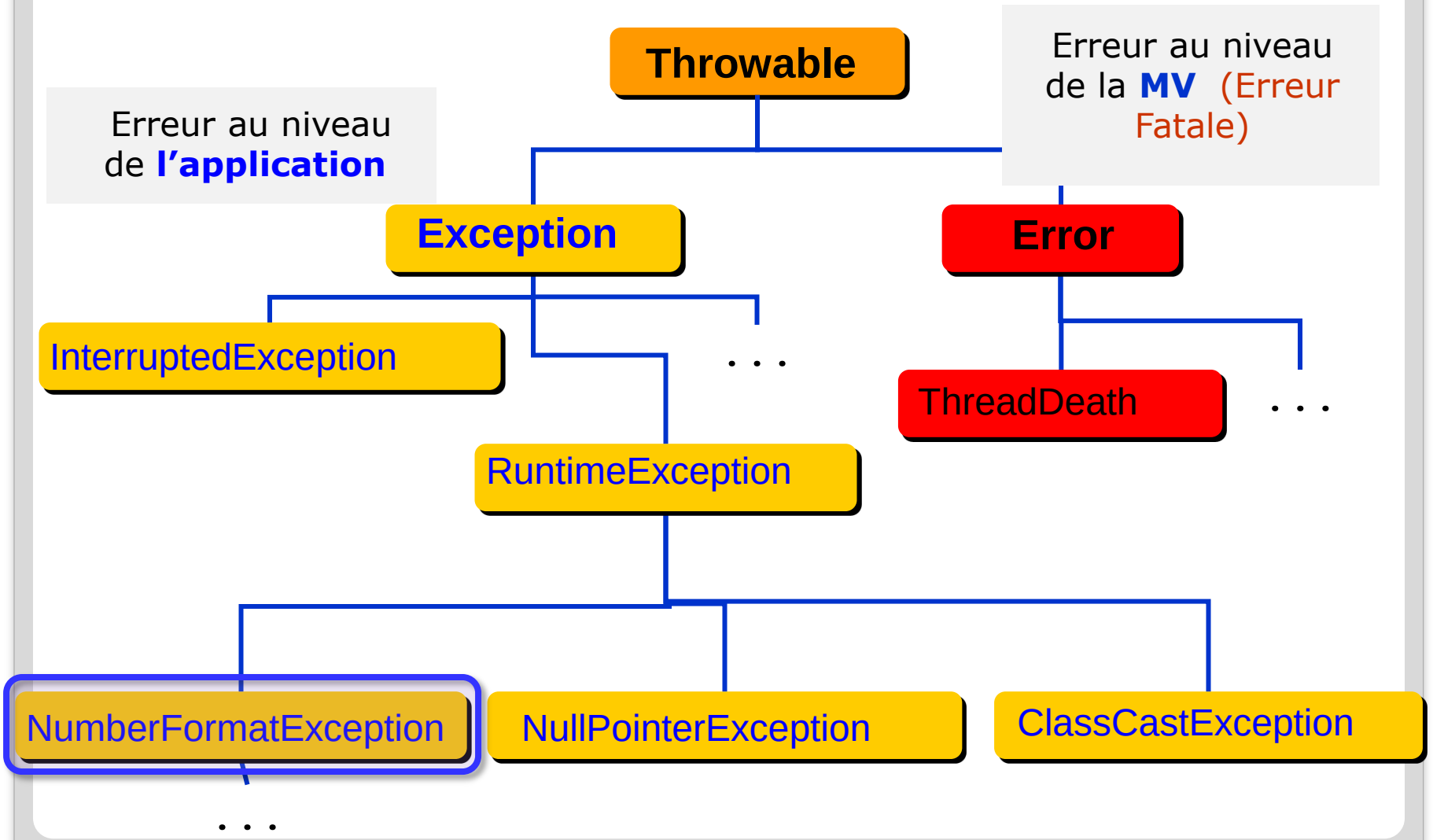
→ La valeur de retour doit servir **à retourner des résultats**

→ On risque d'oublier le **traitement des valeurs de retour**

Exception ❓

Objet *contenant l'information relative au problème*

Exceptions : java.lang.Throwable



Exceptions : exemple

NumberFormatException

```
int moyenne(String[] lst)
{
   int somme=0,entier,nbNotes=0;
   for (int i=0;i<lst.length ;i++){
      entier=Integer.parseInt(lst[i]);
```

12 19 8 14 16 10

API **java.lang.Integer**

public static int **parseInt**(String s) **throws** NumberFormatException

- *Parses the string argument as a signed decimal integer.*
- ***Throws**: NumberFormatException - if the string does not contain a parsable integer.*

Exceptions : exemple

NumberFormatException

1. Ne traite pas l'exception

```
int moyenne(String[] lst) throws NumberFormatException
{
   int somme=0,entier,nbNotes=0;
   for (int i=0;i<lst.length ;i++){
      entier=Integer.parseInt(lst[i]);
      somme+=entier;     nbNotes++;
   }
   return somme/nbNotes;
}
```

```
12 19 8 14 16 10
```

Exceptions : exemple

NumberFormatException

2. Traite l'exception : try/ catch

```
int moyenne(String[] lst)
{
   int somme=0,entier,nbNotes=0;
   for (int i=0;i<lst.length ;i++){
      try {
         entier=Integer.parseInt(lst[i]);
      } catch (NumberFormatException e) {
         System.out.println("Valeur non entiere ");
      }
      somme+=entier;     nbNotes++;
   }
   return somme/nbNotes;
}
```

12 19 8 14 16 10

Si la liste est vide ?

Exceptions : exemple

```
int moyenne(String[] lst) throws DivZeroException
{
  //…
  if (nbNotes == 0) throw new DivZeroException();
  return somme/nbNotes;
}
```

La méthode indique au compilateur qu'elle peut remonter une exception

La méthode lève une exception d'une manière explicite

➔ Une méthode() qui appelle **`moyenne()`** doit :

1. *la remonter à son tour : clause* **`throws`** *dans sa déclaration*

ou bien

2. *intercepter et traîter l'exception : bloc* **`try /catch`**

Exceptions : exemple

```
int moyenne(String[] lst) throws DivZeroException
{
  //…
  if (nbNotes == 0) throw new DivZeroException();
  return somme/nbNotes;
}
```

```
public void methode1(String[] a) throws DivZeroException
{
  int m = moyenne(a) ;   //…
}
```

Exceptions : exemple

```
int moyenne(String[] lst) throws DivZeroException
{
  //…
  if (nbNotes == 0) throw new DivZeroException();
  return somme/nbNotes;
}
```

```
public void methode2(String[] a) {
  try {
     int m = moyenne(a) ;
  }catch (DivZeroException e){ System.out.println(e);}}
```

Exceptions : exemple

```
int moyenne(String[] lst) throws DivZeroException
{
  //…
  if (nbNotes == 0) throw new DivZeroException();
  return somme/nbNotes;
}
```

```
class DivZeroException extends Exception{
    public String toString() {
      return "Aucune note n'est valide";
    }
    // Constructeurs, getters
}
```

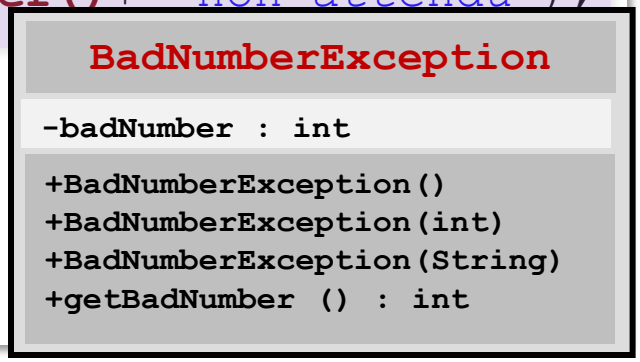
Exceptions : clause try/catch

```
try {
   // code pouvant lever une exception
   // délimite la portion de code à surveiller
}
catch (MonException e) {
   // Code à exécuter si MonException est levée
   // Intercepte et traite une exception
}
catch (Exception e) {
   System.out.println(e);
}
finally {
   //Code toujours exécuté, exception levée ou non
}
```

Exceptions : exemple

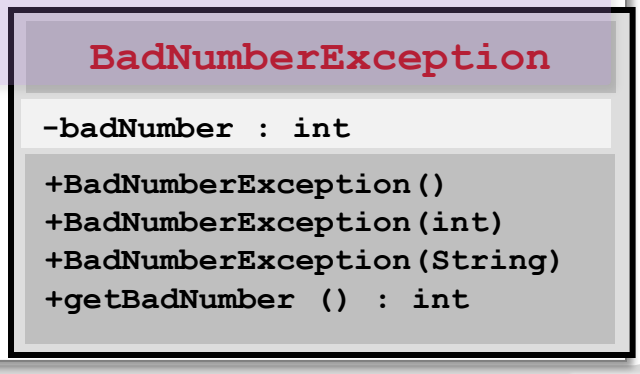
```
public class BadNumber {

  public static void main(String[] args) {

    System.out.println("Entrez un nombre:");

    try {
      Scanner kbd = new Scanner(System.in);
      int i = kbd.nextInt();

      if(i != 10) throw new BadNumberException(i);

    } catch (BadNumberException e) {

      System.out.println(e.getBadNumber()+ "non attendu");

    }
    System.out.println("FIN");
  }
}
```

```
5
5 non attendu
FIN !!!!
```



Exceptions : exemple

```
public class BadNumberException extends Exception {
   private int badNumber;

   public BadNumberException() {
      super("Exception BadNumberException");
   }
   public BadNumberException(String message) {
      super(message);
   }
   public BadNumberException(int badNumber) {
      super();
      this.badNumber = badNumber;
   }
   public int getBadNumber() {
      return badNumber;
   }
}
```



ENSIAS

جامعة محمد الخامس بالرباط
Université Mohammed V de Rabat

Classes et Objets (suite)