EF1
Bases de la Programmation Orientée Objet

- Valeur et référence
- Exceptions
- Evaluation (objectifs)

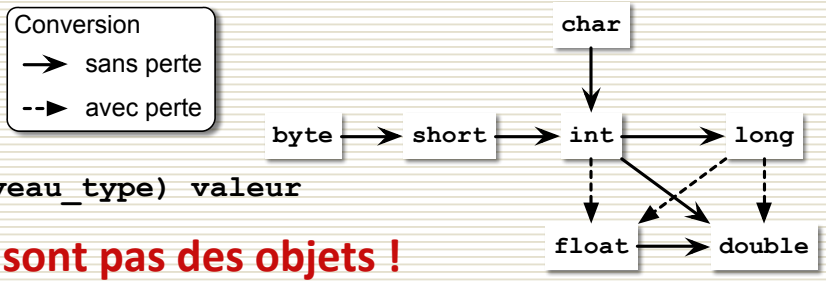
VALEUR ET RÉFÉRENCE EN JAVA

Que contient une variable ?

- ✓ Le contenu d'une variable est différent selon que …
  - … c'est une variable de **type primitif**
  - … c'est une variable de **type Classe** (d'objet)
- ✓ Il est important de faire la différence…

Les types primitifs…

- ✓ Qui sont-ils ?
  - Booléen
    - **`boolean`** (valeurs true et false)
  - Caractère
    - **`char`** (unicode de \u0000 à \uffff, 128 premiers : codes ASCII)
  - Types entiers (… taille représentation)
    - **`byte`** (1 octet), **`short`** (2 octets), **`int`** (4 octets), **`long`** (8 octets)
  - Types flottants
    - **`float`** (4 octets), **`double`** (8 octets)
- ✓ Les conversions possibles avec l'opérateur *cast*



  - `(nouveau_type) valeur`
- ✓ **Ce ne sont pas des objets !**

Contenu d'une variable…

… de Type primitif

- ✓ déclaration-initialisation

```
int i = 12;
```

- ✓ état de la mémoire

variable	adresse	contenu mémoire
i	$ff34ef24	12

- ✓ À RETENIR !
  - **une variable de type primitif contient une valeur**

… de Type Classe d'Objet

- ✓ déclaration-initialisation

```
String s = new String("ABC");
```

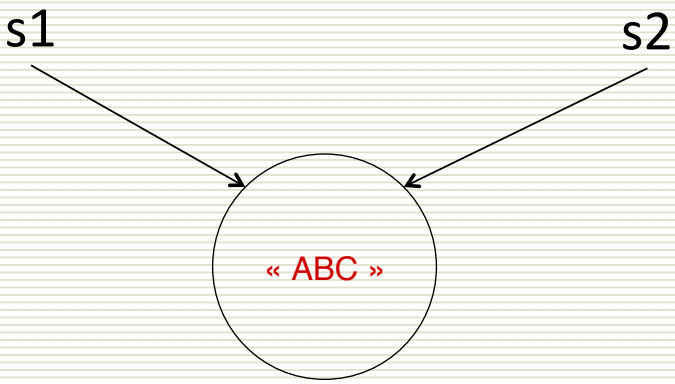
- ✓ état de la mémoire

variable	adresse	contenu mémoire
s	$ff34ef68	$ff34effa
	$ff34effa	"ABC"

- ✓ À RETENIR !
  - **une variable de type Classe contient une référence** à un objet (un pointeur vers un objet, l'adresse d'un objet)

… Donc variables de même référence

- ✓ 2 variables peuvent donc contenir la même référence et « pointer » vers le même objet



Exemple : géométrie (1/2)

- ✓ **Attention aux références**

- ✓ Exemple avec l'origine d'une forme

```
private Point origine;

public Point getOrigine() {
      return origine;
}
```


Exemple : géométrie (2/2)

```
Forme forme = new Cercle(0, 0, 10);
Point origine = forme.getOrigine();
origine.deplaceDe(1, 1);
```

- **PROBLÈME** : La forme a été déplacée **SANS UTILISATION** de la méthode déplacée de la forme

Interface cloneable

✓ **SOLUTION** : Rendre un clone de l'objet

```
public class Point implements Cloneable {
        …
        @Override
        protected Point clone() {
         Point point = null;
         try {
             point = (Point) super.clone();
         } catch (CloneNotSupportedException cnse) {
             cnse.printStackTrace(System.err);
         }
         return point;
    }
```

✓ **FONCTIONNE si on n'a pas besoin de cloner en profondeur**

Attention au passage de paramètres

✓ Si paramètre de **type primitif**

- Passage par **valeur**
- S'il est modifié à l'intérieur de la méthode, la modification ne sera que locale.

✓ Si paramètre de **type classe d'objet**

- Passage par **référence**
- S'il est modifié à l'intérieur de la méthode, la modification sera « globale »

Exemple : géométrie

- Une forme est composée d'un point
- Quand on change l'origine d'une forme, la forme doit donc **conserver un clone** du point passé au setter.

```
public void setOrigine(Point origine) {
        this.origine = origine.clone();
    }
```


EXCEPTIONS EN JAVA

Exception : introduction (1/2)

- ✓ Un programme peut rencontrer une erreur ou un événement anormal
  - Erreur technique
    - *Fichier non présent, plus de mémoire, etc.*
  - Erreur métier
    - *Arguments invalides, etc.*
- ✓ Prévoir un **traitement d'erreur** sur les instructions susceptibles de les provoquer
- ✓ En Java, ce traitement est intégré dans le langage : traitement des **exceptions**

Exception : principe

- ✓ Une exception est créée :
  - soit par la JVM [environnement d'exécution] (erreur interne/technique)
  - soit par une levée d'exception du programmeur
- ✓ Deux solutions
  - Soit on **attrape (intercepte)** l'exception immédiatement pour la traiter
  - Soit on **relance (propage)** l'exception à la méthode qui a déclenché le traitement erroné. MAIS on devra l'attraper et la traiter à un moment de la « remontée »

Attraper une exception (1/2)

```
try {
   instructions; // Instructions à contrôler

} catch( MonException e) {
   instructions; // Traitement de l'exception e

} [catch( AutreException e) {…}]
```


Attraper une exception (2/2)

```
Scanner sc = new Scanner(System.in)
System.out.println("Donner un entier :");

try {
      // Donner un entier
      int i = sc.nextInt();
      ...

} catch (Exception e) {
   e.printStackTrace();
}
```

Exception : bloc `finally`

```
try {
   instructions;   // Instructions à contrôler

} catch( MonException e) {
   instructions4; // Traitement de l'exception
   e

} finally {
   instructions;   // Instructions exécutées
                   // dans tous les cas
}
```


Relancer une exception

- ✓ Ajouter dans la signature de la méthode le mot-clef **`throws`** suivi du type d'exception

```
public static int division(int c, int d)
                    throws ArithmeticException
   {

…
}
```

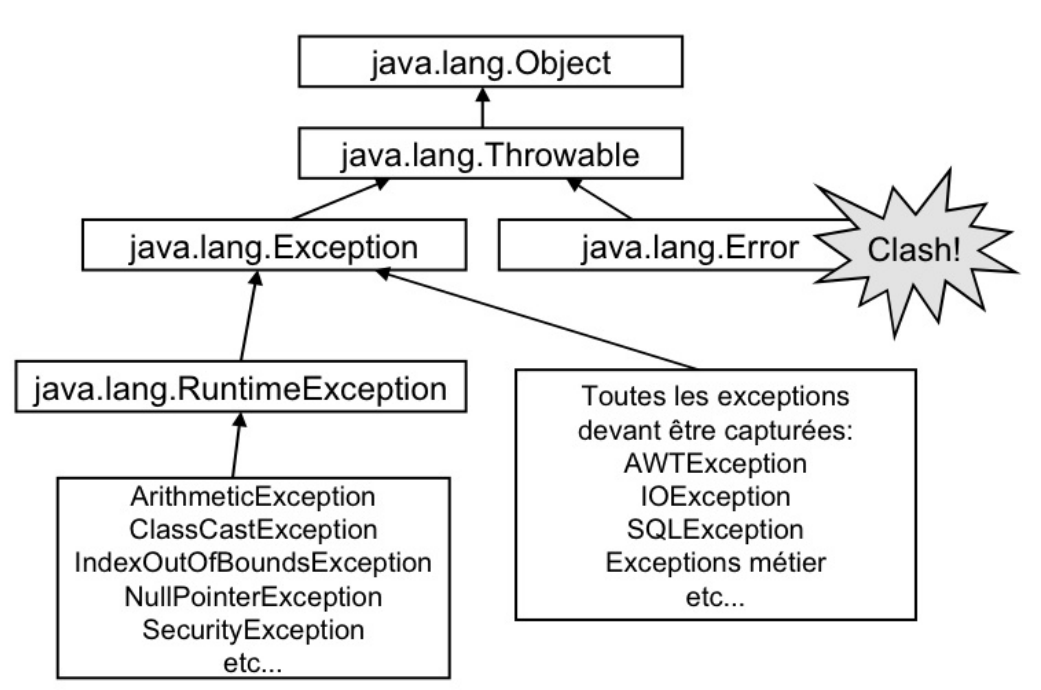
ATTENTION : on devra attraper l'exception et la traiter à un moment de la « remontée »

Lever une exception

```
throw new Exception();
```

- ✓ Autres types d'exceptions possibles :
  - IOException,ParseException,etc.
- ✓ Vous pouvez aussi créer vos propres exceptions. En héritant de la classe Exception !

Hiérarchie des exceptions



Particularité : les RunTimeException n'ont pas à être obligatoirement traitées

Exception

```
class Classe2 {
        Classe1 objet1 = new Classe1();
        // ...

        void methodeX() {
                try {
                        objet1.methode1()
                        // ...

                } catch (Exception exception1) {
                        // Traitement exception

                } finally {
                        // ...
                }
        }

        void methodeY() throws Exception {
                objet1.methode1();
                // ...
        }
}
```

```
class Classe1 {
        // ...

        void methode1() throws Exception {
                boolean erreur = false;
                // erreur = ...
                // En cas d'erreur
                // déclenchement d'une exception
                if (erreur)
                        throw new Exception();
                // ...
        }
}
```


Exception

```
class Classe2 {
        Classe1 objet1 = new Classe1();
        // ...

        void methodeX() {
                try {
                        objet1.methode1()
                        // ...

                } catch (Exception exception1) {
                        // Traitement exception

                } finally {
                        // ...
                }
        }

        void methodeY() throws Exception {
                objet1.methode1();
                // ...
        }
}
```

```
class Classe1 {
        // ...

        void methode1() throws Exception {
                boolean erreur = false;
                // erreur = ...
                // En cas d'erreur
                // déclenchement d'une exception
                if (erreur)
                        throw new Exception();
                // ...
        }
}
```

Exécution normale

Exception

```
class Classe2 {
        Classe1 objet1 = new Classe1();
        // ...

        void methodeX() {
                try {
                        objet1.methode1()
                        // ...

                } catch (Exception exception1) {
                        // Traitement exception

                } finally {
                        // ...
                }
        }

        void methodeY() throws Exception {
                objet1.methode1();
                // ...
        }
}
```

```
class Classe1 {
        // ...

        void methode1() throws Exception {
                boolean erreur =  true;
                // erreur = ...
                // En cas d'erreur
                // déclenchement d'une exception
                if (erreur)
                        throw new Exception();
                // ...
        }
}
```

Exception levée

Exception

```
class Classe2 {
        Classe1 objet1 = new Classe1();
        // ...

        void methodeX() {
                try {
                        objet1.methode1()
                        // ...

                } catch (Exception exception1) {
                        // Traitement exception

                } finally {
                        // ...
                }
        }

        void methodeY() throws Exception {
                objet1.methode1();
                // ...
        }
}
```

```
class Classe1 {
        // ...

        void methode1() throws Exception {
                boolean erreur = false;
                // erreur = ...
                // En cas d'erreur
                // déclenchement d'une exception
                if (erreur)
                        throw new Exception();
                // ...
        }
}
```

Exécution normale

Exception

```
class Classe2 {
        Classe1 objet1 = new Classe1();
        // ...

        void methodeX() {
                try {
                        objet1.methode1()
                        // ...

                } catch (Exception exception1) {
                        // Traitement exception

                } finally {
                        // ...
                }
        }

        void methodeY() throws Exception {
                objet1.methode1();
                // ...
        }
}
```

```
class Classe1 {
        // ...

        void methode1() throws Exception {
                boolean erreur = true;
                // erreur = ...
                // En cas d'erreur
                // déclenchement d'une exception
                if (erreur)
                        throw new Exception();
                // ...
        }
}
```

Exception levée

Discussion sur Faërun

- Lors d'un coût critique (dégât donné maximum), tous les guerriers ennemis sur le carreau meurent.
  - exemple: nain bleu réalise un coût critique [dégat donné = 30] sur un nain rouge. Tous les rouges sur le carreau meurent.

1. Où lever l'exception ?
2. Comment la définir ?
3. Où la récupérer ? Que faire ?