Le langage JAVA

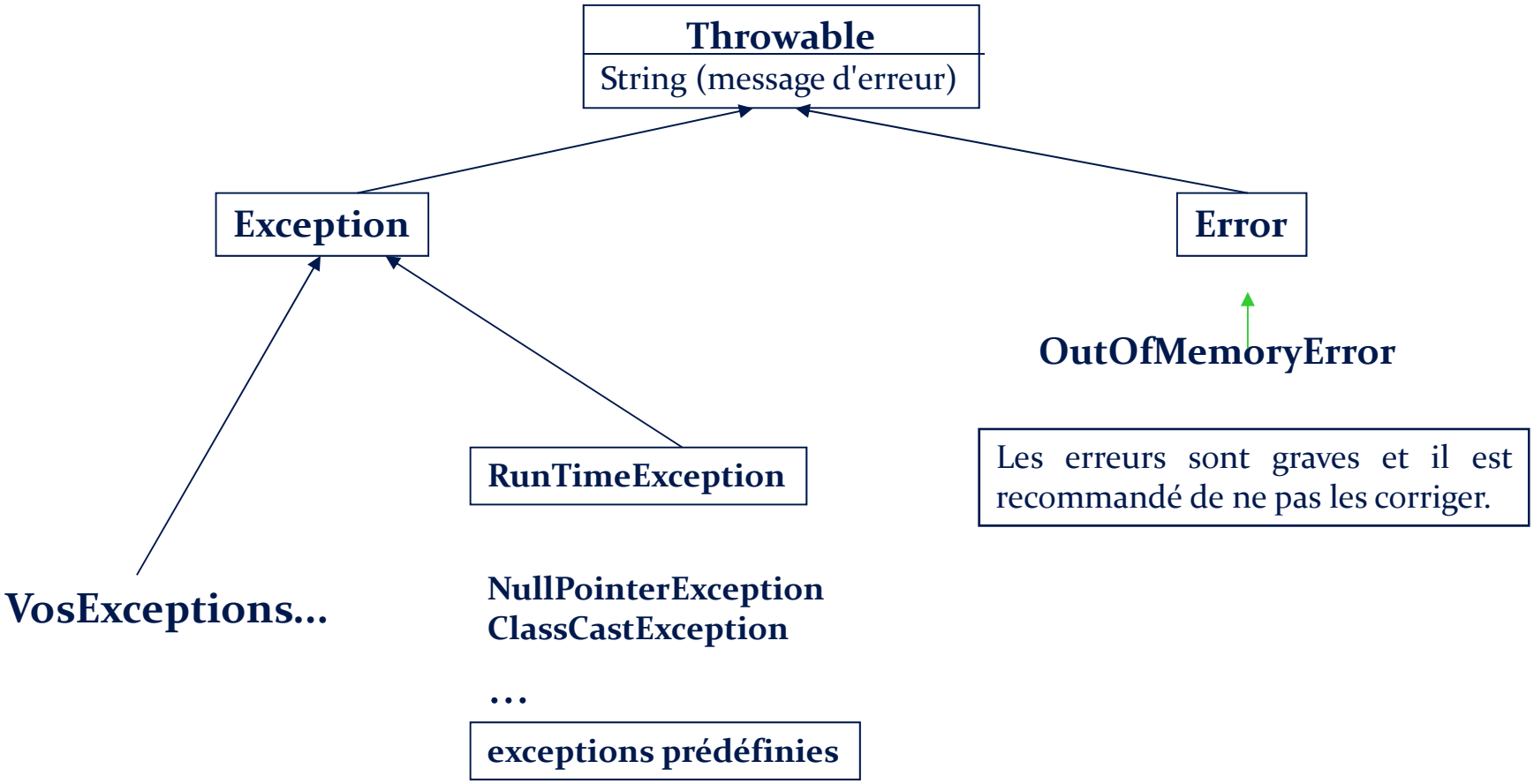
Les exceptions

PRÉVOIR LES ERREURS D'EXÉCUTION

- Certains cas d'erreurs peuvent être prévus à l'avance par le programmeur.
  Exemples:
  - ✓ erreurs d'entrée-sortie (I/O fichiers)
  - ✓ erreurs de saisie de données par l'utilisateur

- Le programmeur peut :
  - ✓ «Laisser planter» le programme à l'endroit où l'erreur est détectée
  - ✓ Manifester explicitement le problème à la couche supérieure
  - ✓ Tenter une correction

ARBRES DES EXCEPTIONS



NATURE DES EXCEPTIONS

- En Java, les exceptions sont des objets ayant 3 caractéristiques:
  - ✓ Un type d'exception (défini par la classe de l'objet exception)
  - ✓ Une chaîne de caractères (option), (hérité de la classe Throwable).
  - ✓ Un « instantané » de la pile d'exécution au moment de la création.
- Les exceptions construites par l'utilisateur étendent la classe Exception
- **RunTimeException**, **Error** sont des exceptions et des erreurs prédéfinies et/ou gérées par Java.
- Quelques exceptions prédéfinies en Java :
  - ✓ Division par zéro pour les entiers : **ArithmeticException**
  - ✓ Référence nulle : **NullPointerException**
  - ✓ Tentative de forçage de type illégale : **ClassCastException**
  - ✓ Tentative de création d'un tableau de taille négative : **NegativeArraySizeException**
  - ✓ Dépassement de limite d'un tableau : **ArrayIndexOutOfBoundsException**

CAPTURE D'UNE EXCEPTION

- Les sections **try** et **catch** servent à capturer une exception dans une méthode (attraper la bulle…)

```
try
   {
      // code à tester
   }
catch (<une-exception>)
   {
      // code si exception
   }
catch (<une_autre_exception>)
   {
      // code si autre exception
   }

...

finally
   {
      // code facultatif exécuté dans tous les cas
   }
```

TRAITEMENT DES EXCEPTIONS

- Le bloc **try** est exécuté jusqu'à ce qu'il se termine avec succès ou bien qu'une exception soit levée.
- Dans ce dernier cas, les clauses **catch** sont examinées l'une après l'autre dans le but d'en trouver une qui traite cette classe d'exceptions (ou une superclasse).
- Les clauses **catch** doivent donc traiter les exceptions de la plus spécifique à la plus générale.
- Si une clause **catch** convenant à cette exception a été trouvée et le bloc exécuté, l'exécution du programme reprend son cours.
- Si elles ne sont pas immédiatement capturées par un bloc **catch**, les exceptions se propagent en remontant la pile d'appels des méthodes, jusqu'à être traitées.
- Si une exception n'est jamais capturée, elle se propage jusqu'à la méthode **main()**, ce qui pousse l'interpréteur Java à afficher un message d'erreur et à s'arrêter.
- L'interpréteur Java affiche un message identifiant :
  - ✓ l'exception,
  - ✓ la méthode qui l'a causée,
  - ✓ la ligne correspondante dans le fichier.
- Un bloc **finally** permet au programmeur de définir un ensemble d'instructions qui est toujours exécuté, que l'exception soit levée ou non, capturée ou non, comme libérer les ressources.
- La seule instruction qui peut faire qu'un bloc **finally** ne soit pas exécuté est **System.exit()**.

INTERCEPTION VS PROPAGATION

- Si une méthode peut émettre une exception (ou appelle une autre méthode qui peut en émettre une) il faut :
  - ✓ soit propager l'exception (la méthode doit l'avoir déclarée);
  - ✓ soit intercepter et traiter l'exception
- Exemple de propagation :

```
public int ajouter(int a, String str) throws NumberFormatException {
  int b = Integer.parseInt(str);
  a = a + b;
  return a;
}
```

- Exemple d'interception :

```
public int ajouter(int a, String str) {
  try {
    int b = Integer.parseInt(str);
    a = a + b;
  }
  catch (NumberFormatException e) {
    System.out.println(e.getMessage());
  }
  return a;
}
```

LES OBJETS EXCEPTION

- La classe **Exception** hérite de La classe **Throwable**.
- La classe **Throwable** définit un message de type String qui est hérité par toutes les classes d'exception.
- Ce champ est utilisé pour stocker le message décrivant l'exception.
- Il est positionné en passant un argument au constructeur.
- Ce message peut être récupéré par la méthode **getMessage()**.
- Exemple :

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

LEVÉE D'EXCEPTIONS

- Le programmeur peut lever ses propres exceptions à l'aide du mot réservé **throw**, qui prend en paramètre un objet instance de **Throwable** ou d'une de ses sous-classes.
- Les objets exception sont souvent instanciés dans l'instruction même qui assure leur lancement.
- Exemple : `throw new MonException("Mon exception s'est produite !!!");`
- Et l'émission :

```
public void ouvrirFichier(String name) throws MonException
{
  if (null == name) throw new MonException();
  else
    {...}
}
```

- Pour "laisser remonter" à la méthode appelante une exception qu'il ne veut pas traiter, le programmeur rajoute le mot réservé **throws** à la déclaration de la méthode dans laquelle l'exception est susceptible de se manifester.
- La classe de l'exception indiquée peut tout à fait être une super-classe de l'exception générée.
- Une même méthode peut tout à fait "laisser remonter" plusieurs types d'exceptions (séparés par des ,).
- Une méthode doit traiter ou "laisser remonter" toutes les exceptions qui peuvent être générées dans les méthodes qu'elle appelle.