Le langage Java

Apprentissage en lien avec le langage UML

Le langage Java

Apprentissage en lien avec le langage UML

Le contexte

- Dans notre contexte sécuriser un programme c'est :
  - Prévoir les erreurs
  - les détecter
  - et réagir
- L'un des objectifs du le langage Java est la sécurité
  - Il propose un mécanisme dédié : les exceptions

En pratique

- Pour les langages qui ne proposent pas de traitement spécifique des erreurs :
  - Les cas d'erreur doivent être prévus (analyse du code, de l'utilisation, ...)
  - Des procédures de test sont mises en place (manuelles et automatiques)
  - Des réactions aux erreurs sont prévues
    - **Traiter** : Avertissement, correction dynamique, arrêt contrôlé, ...
    - Mais aussi **remonter l'erreur** vers le code appelant.
- Le code de test est confondu avec celui de l'application
  - Difficile à lire
  - Difficile à maintenir
  - Qui traite l'erreur l'appelant ou l'appelé ?

Une classe, des tests et des réactions

```
package coursSSI3.exemples.exceptions;
import coursSSI3.exemples.animaux.Chien;

public class Traineau {
        public final int capacite;
        protected int occupation = 0;
        protected Chien[] contenu;
        public Traineau(int capacite) {
                this.capacite = capacite;
                contenu = new Chien[capacite];
        }
        public boolean estComplet() {
                return occupation == capacite;}
        public boolean estVide() {return occupation == 0;}

        public void ajouter(Chien c) {
                if (!estComplet()) contenu[occupation++] = c;
                else System.out.println("Traineau complet !");
        }
        public void liberer() {
                if (!estVide()) contenu[--occupation] = null;
                else System.out.println("Traineau vide !");
        }
}
```

Listing 1 – coursSSI3/exemples/exceptions/Traineau.java

Un programme principal incomplet

```
package coursSSI3.exemples.exceptions;
import coursSSI3.exemples.animaux.*;

public class Test {
        public static void main(String[] args) {
                Traineau t;
                int nbChiens;
                // Creation d'un traineau de taille donnee
                t = new Traineau(Integer.parseInt(args[0]));
                // Nb de chiens
                nbChiens = Integer.parseInt(args[1]);

                int i = nbChiens;
                while (i--> 0 && !t.estComplet())
                        t.ajouter(new Chien());
                i = nbChiens;

                System.out.println("occupation : "
                                +100*nbChiens/t.capacite);

                while (i--> 0 && !t.estVide())
                        t.liberer();
        }
}
```

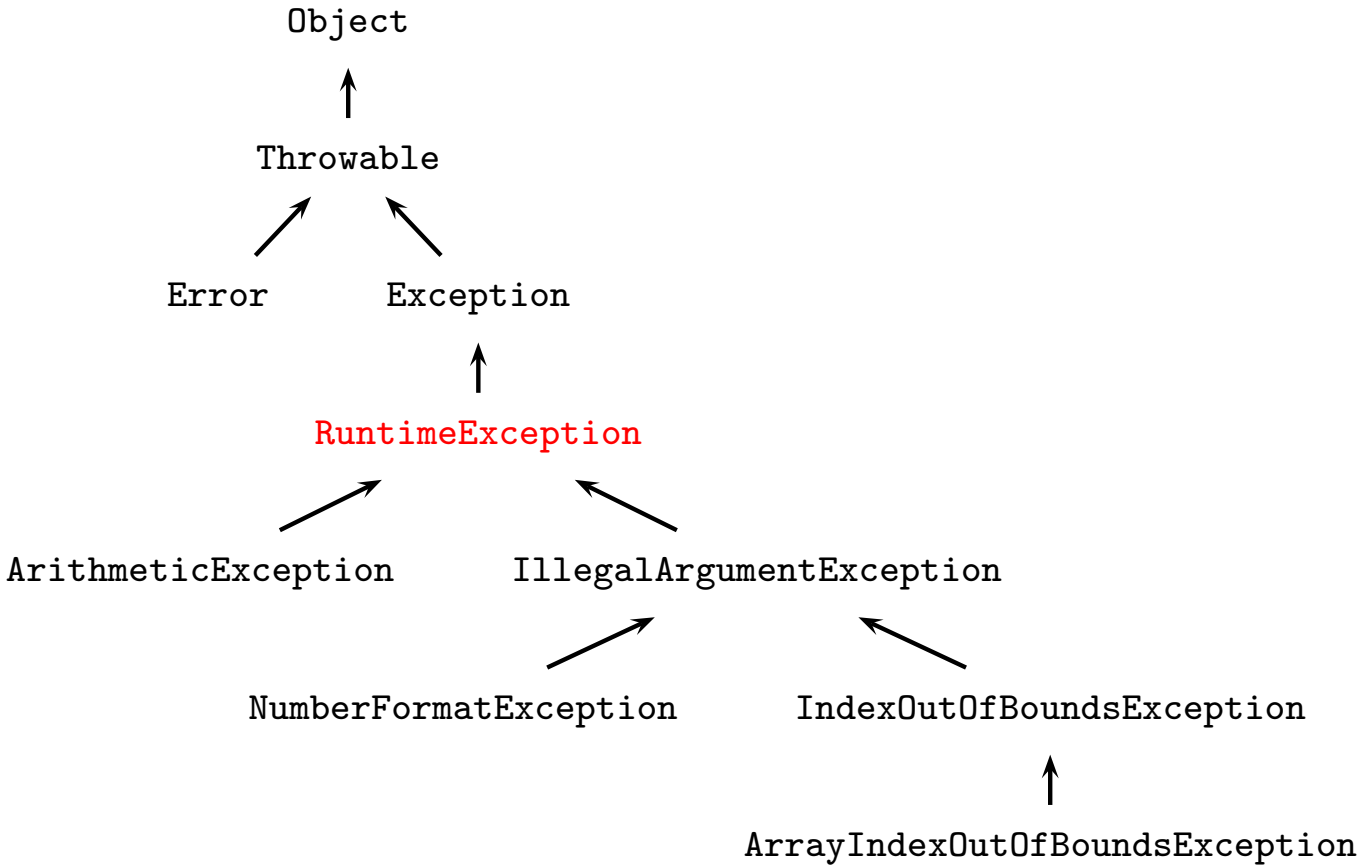
Listing 2 – coursSSI3/exemples/exceptions/Test.java

Avec Java

- Dans le programme précédent
  - Le programme teste et réagit à la même erreur plusieurs endroits
  - que se passe-t-il si :
    - Il n'y a pas assez de paramètres
    - Les paramètres (des chaînes) ne représentent pas nombres
    - Le premier paramètre vaut 0
- Pour traiter cela, java propose :
  - un mécanisme le bloc `try...catch...finally`
    - On exécute et on réagit éventuellement : mécanisme implicite
    - Séparation du code et du traitement des erreurs
    - Erreurs classiques : Division par zéro, dépassement de tableaux, ...
    - Extensibilité des erreurs
  - une hiérarchie de classe pour les erreurs

Avec Java

- Toutes les erreurs potentielles doivent être traitées par l'appelant
- Sauf celles de type RuntimeException :



Un programme principal amélioré

```
package coursSSI3.exemples.exceptions;

import coursSSI3.exemples.animaux.*;

public class TestErreur {
 public static void main(String[] args) {
  Traineau t = null;
  int nbChiens = 0;
  try {
    t = new Traineau(Integer.parseInt(args[0]));
    nbChiens = Integer.parseInt(args[1]);

    int i = nbChiens;
    while (i--> 0 && !t.estComplet()) t.ajouter(new Chien());
    i = nbChiens;
    while (i--> 0 && !t.estVide()) t.liberer();
  } catch (ArrayIndexOutOfBoundsException e) {
          System.out.println(
        "Test exception <TailleTraineau> <nbCheins>");
  } catch (NumberFormatException e) {
          System.out.println("Parametres non numeriques !"); }
 }
}
```

Listing 3 – coursSSI3/exemples/exceptions/TestErreur.java

La remontée d'erreur

- La première réaction possible a une erreur c'est de la transmettre au programme appelant
- En Java, on utilise l'instruction `throw` *`exception`*
  - dans le corps d'un programme (par exemple dans un `catch`)
- C'est le comportement par défaut pour les `RuntimeException`
- Toute erreur est interceptée au plus tard par le programme principal encadré par défaut par :

```
try
...
catch(Throwable t)
System.err.println(t.printStackTrace()) ;
```

Une classe qui remonte des exceptions

```
package coursSSI3.exemples.exceptions;

import coursSSI3.exemples.animaux.Chien;

public class TraineauThrows extends Traineau {

 public TraineauThrows(int capacite) {super(capacite);}

 public int getRatioOccupation() {
  try {
        return 100*occupation/capacite;
  } catch (ArithmeticException e) {
        return -1;}
 }

 public int getRatioOccupationSur() {
  try {
          return 100*occupation/capacite;
  } catch (ArithmeticException e) {
        System.out.println("Ratio non calculable");
        throw e; }
 }
}
```

Listing 4 – coursSSI3/exemples/exceptions/TraineauThrows.java

La création d'une nouvelle exception

- Pour traiter les cas d'erreur particulier à une application :
  - On étend la classe `Exception`
  - Ces erreurs devront être traitées par l'utilisateur
- Quand une méthode peut émettre une erreur, on l'indique dans la déclaration avec la directive `throws`
- Pour l'émettre, on crée une instance qui est émise pas `throw`

Des exceptions personnelles

```
package coursSSI3.exemples.exceptions;

public class TraineauVideException extends Exception {

}
```

Listing 5 – coursSSI3/exemples/exceptions/TraineauVideException.java

```
package coursSSI3.exemples.exceptions;

public class TraineauPleinException extends Exception {

}
```

Listing 6 – coursSSI3/exemples/exceptions/TraineauPleinException.java

Une classe qui émet ses exceptions

```
package coursSSI3.exemples.exceptions;

import coursSSI3.exemples.animaux.Chien;

public class TraineauErreur extends TraineauThrows {

        public TraineauErreur(int capacite) {
                super(capacite);
        }

        public void ajouterSur(Chien c)
          throws TraineauPleinException {
                if (!estComplet()) contenu[occupation++] = c;
                else throw new TraineauPleinException();
        }

        public void libererSur()
          throws TraineauVideException {
                if (!estVide()) contenu[--occupation] = null;
                else throw new TraineauVideException();
        }

}
```

Listing 7 – coursSSI3/exemples/exceptions/TraineauErreur.java

Le programme principal fini

```
package coursSSI3.exemples.exceptions;
import coursSSI3.exemples.animaux.*;
public class TestErreur2 {
  public static void main(String[] args) {
    TraineauErreur t = null;  int nbChiens = 0;
    try {t = new TraineauErreur(Integer.parseInt(args[0]));
      nbChiens = Integer.parseInt(args[1]); // Nb de chiens
      int i = nbChiens;try {while (i--- > 0)
          t.ajouterSur(new Chien());
      } catch (TraineauPleinException e) {
        System.out.println("Le traineau est trop petit !");}

      System.out.println("occup.: "+t.getRatioOccupationSur());

      i = nbChiens;try {while (i--- > 0 && !t.estVide())
          t.libererSur();
      } catch (TraineauVideException e) {
        System.out.println("Il n'y a plus de chiens !");}

    } catch (ArrayIndexOutOfBoundsException e) {
      System.out.println("TestExcep <TTraineau> <nbChiens>");
    } catch (NumberFormatException e) {
      System.out.println("Parametres non numeriques !");
    }
  }
}
```

Listing 8 – coursSSI3/exemples/exceptions/TestErreur2.java

La programmation assertionnelle

- Dans certains, cas le traitement de erreur peut être utilisé pour vérifier que le programme fonctionne dans un état prévu :
  - On vérifie que les variables restent dans des domaines précis
  - C'est lourd à mettre en place et à maintenir
- Java 5 propose les assertions
  - Cela permet de vérifier dynamiquement que des conditions sont toujours vraies
  - Quand ce n'est pas le cas, une exception est levée

L'utilisation des assertions

```
package coursSSI3.exemples.exceptions;

public class VehiculeAvecAssertion {
        int vitesse=0; /* la vitesse en km/s */

        public static void main(String args[]) {
                VehiculeAvecAssertion v =
                        new VehiculeAvecAssertion();

                v.vitesse = -3;
                v.vitesse = 400000;

                assert v.vitesse >= 0 && v.vitesse <= 300000;
        }

}
```

Listing 9 – coursSSI3/exemples/exceptions/VehiculeAvecAssertion.java