digitalent
FORMATION

Le langage Java

Exceptions

Sujets abordés

- Approche conventionnelle
- Gestionnaire d’exceptions
- Objet de type Exception
- Hiérarchie des exceptions
- Code protégé

Approche conventionnelle

- Utilisation de codes d'erreur
  - les fonctions retournent des codes
  - tester les codes à chaque appel de fonction
  - traiter les erreurs ou continuer l'exécution
- Maintenance difficile
- Pas de catégorie d'erreurs
- Risque de cas d'erreurs non traitées

Principaux rôles des exceptions

- séparer les traitements “exceptionnels” du code “normal”
  - éviter les structures conditionnelles difficiles à maintenir
- propager les erreurs dans la pile de messages
  - mécanisme très simple et intuitif pour le développeur
- classifier les erreurs
  - par la création de classes représentant les exceptions
  - classification “naturelle”
    - pas de codes arbitraires
  - utilisation de l’héritage pour regrouper les exceptions

Gestionnaire d'exceptions

- Pour traiter une anomalie
- «Envoi» d'un objet Exception
- Recherche du gestionnaire approprié
  - si trouvé (dans la pile d'appels)
    - traitement de l'exception
    - retour au programme
  - sinon
    - déclenchement d'une erreur et arrêt de l'exécution

Objet de type Exception

- Création d'un objet Exception
  - en général sous-classe de **Exception**
  - voir classe **java.lang.Exception**
  - `new ClasseException();`
- Envoi de l'exception
  - expression à insérer dans le code de la méthode
  - instruction **throw**
  - `throw new ClasseException();`

Déclaration des exceptions lancées

- La méthode “contient” une liste d’exceptions potentielles
- Il faut spécifier toutes les exceptions qui peuvent être envoyées
  - dans la déclaration de la méthode

```
typeRetour nomMethode(...) throws
    liste des types d'exceptions {
        ...
}
```

Code protégé

- Blocs try-catch
- Séparation exécution / traitement d'erreur

```
try {
//exécution sécurisée
} catch (TypeException e) {
//gestionnaire d'exception
}
```

Exemple

- Exemple d'exception

```
double diviser(int a, int b) throws DivException {
  if (b == 0) {
    throw new DivException();
  }
  return a / b;
}

double appelerDiviser(int x, int y) {
  try {
    return diviser(x, y);
  } catch (DivException e) {
  return 0;
}
```

Hiérarchie des exceptions

- Racine : classe Throwable
  - objet qui peut être «envoyé»
  - contient des méthodes générales
- Sous-classe de Throwable
  - **Error** (erreurs système et de compilation)
  - **Exception**
    - **RuntimeException** (erreurs à l'exécution)
    - ... (erreurs applicatives)

Opérations sur une exception

- Une exception peut recevoir des messages
- Exemples
  - e.getMessage()
  - e.printStackTrace()
  - e.getCause()
  - ...

```
catch (Exception e) {
    System.out.println(e.getMessage());
}
```

Compilation des programmes

- Exceptions «RuntimeException»
  - pas de problème à la compilation (pas besoin de try-catch), mais erreur déclenchée à l'exécution
  - deux causes :
    - mauvais développement ➔ à corriger
    - erreur déclenchée par le système (imprévisible) ➔ pas de correction, sinon code illisible
- Autres exceptions
  - doivent être traitées
    - **try-catch**
    - ou **throws *NomException***
      - la méthode ne traite pas l'exception mais la propage
  - sinon erreur de compilation

Créer une nouvelle exception

- Créer une nouvelle classe
  - sous-classe directe de **Exception**
  - ou utiliser les hiérarchies existantes

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

Bloc finally

- Assure l'exécution d'un bloc d'instructions
- Assurer un état cohérent dans le programme
  - fermeture de fichier par exemple

```
try {...}       // code protégé
catch (...) {...}   // plusieurs gestionnaires
catch (...) {...}   // d'exceptions
...
finally {...}  // bloc toujours exécuté
```

Exceptions - contraintes

- Une méthode redéfinie ne doit pas ajouter d'exception à la liste de ses exceptions potentielles
  - liste d'exceptions définie dans la super-classe
  - `...throws` ***`liste d'exceptions`***
  - sauf dans le cas d'un constructeur