BTS
BREVET DE TECHNICIEN SUPÉRIEUR
LYCÉE QUALIFIANT MOHAMED V
ESSAOUIRA

PROGRAMMATION ORIENTÉE OBJET (JAVA)

Les Exceptions

1. Définition

Un programme ne se déroule pas toujours comme prévu. Plusieurs cas d'erreurs peuvent venir perturber son bon déroulement: division par zéro, dépassement des bornes d'un tableau, fichier inexistant, …

Une exception est une interruption de l'exécution d'un programme suite à une erreur. Par exemple, une division par zéro provoque une exception de type ArithmeticException.

Les exceptions en java sont des instances de sous classes des classes:

- java.lang.Error (pour des erreurs graves, qui devront généralement conduire à l'arrêt du programme)

- java.lang.Exception (pour des événements inattendus, qui seront souvent traités de sorte qu'elle ne provoque pas l'arrêt du programme)

2. Attraper une exception

1. try et catch:

Le mot clé try permet de spécifier un bloc de code sur lequel on s'attend qu'une exception soit levée (possibilité d'avoir une erreur).

Le mot clé catch sert à spécifier le code à exécuter pour une exception donnée. Les blocs try et catch sont utilisés comme suit:

```
try
    {
     //zone contenant des instructions pouvant générer des
erreurs
    }
catch (NomException e)
    {
     //Traitement à faire dans le cas de l'exception e
    }
```

2. Attraper une exception

1. try et catch:

Exemple :

```
public class ExceptionExample {
    public static void main(String[] args) {
        try {
            int a = 10, b = 0;
            int result = a / b; // Provoque une ArithmeticException
            System.out.println("Résultat: " + result);
        } catch (ArithmeticException e) {
            System.out.println("Erreur : Division par zéro !");
        }
        System.out.println("Programme terminé.");
    }
}
```

2. Attraper une exception

2. throws:

Une méthode susceptible de lancer une exception sans l'attraper doit l'indiquer dans son en-tête par le mot clé throws.

Ce mot clé permet d'avertir le système qu'une certaine catégorie d'exception ne seront pas traitées.

Syntaxe:

En-tête méthode throws ClasseException

Exemple:

void fonction throws IOException

2. Attraper une exception

Exemple Checked (extends Exception):

```
class InvalidAgeException extends Exception {
    public InvalidAgeException(String message) {
        super(message);
    }
}
```

```
public class TestCustomException {
    // Méthode qui vérifie l'âge
    static void checkAge(int age) throws InvalidAgeException {
        if (age < 18) {
            throw new InvalidAgeException("Âge invalide : Vous devez avoir au moins 18 ans.");
        }
        System.out.println("Âge valide !");
    }

    public static void main(String[] args) {
        try {
            checkAge(16); // Provoque l'exception
        } catch (InvalidAgeException e) {
            System.out.println("Exception attrapée : " + e.getMessage());
        }
    }
}
```

2. Attraper une exception

Exemple UnChecked (extends RuntimeException):

```
class InsufficientBalanceException extends RuntimeException {
    public InsufficientBalanceException(String message) {
        super(message);
    }
}
```

```
class BankAccount {
    private double balance;

    public BankAccount(double balance) {
        this.balance = balance;
    }

    // Méthode pour retirer de l'argent
    public void withdraw(double amount) {
        if (amount > balance) {
            throw new InsufficientBalanceException("Fonds insuffisants. Solde actuel : " + balance);
        }
        balance -= amount;
        System.out.println("Retrait réussi ! Nouveau solde : " + balance);
    }
}
```

2. Attraper une exception

Exemple Unchecked (extends RuntimeException):

```
public class TestBankAccount {
    public static void main(String[] args) {
        BankAccount account = new BankAccount(500);

        try {
            account.withdraw(700); // Déclenche InsufficientBalanceException
        } catch (InsufficientBalanceException e) {
            System.out.println("Exception attrapée : " + e.getMessage());
        }
    }
}
```

2. Attraper une exception

Exemple 2:

Type d'Exception	Hérite de	Obligation de gestion	Exemple
Checked Exception	Exception	✅ Doit être capturée (try-catch) ou déclarée (throws)	IOException, SQLException
Unchecked Exception	RuntimeException	❌ Pas obligatoire de gérer	NullPointerException, IndexOutOfBoundsException

2. Attraper une exception

3. finally:

Un bloc finally est un bloc d'intsructions précédé du mot clé finally. Ce bloc sera toujours exécuté après le traitement d'exception. Il est en général utilisé pour fermer des fichiers, libérer des ressources, …

Un bloc finally suit un bloc try ou un bloc catch.

2. Attraper une exception

```java
import java.io.*;

public class FileExample {
    public static void main(String[] args) {
        FileReader file = null;
        try {
            file = new FileReader("test.txt");
            BufferedReader br = new BufferedReader(file);
            System.out.println(br.readLine());
        } catch (IOException e) {
            System.out.println("Erreur : Fichier introuvable !");
        } finally {
            try {
                if (file != null) {
                    file.close();
                    System.out.println("Fichier fermé.");
                }
            } catch (IOException e) {
                System.out.println("Erreur lors de la fermeture du fichier.");
            }
        }
    }
}
```

3. Quelques classes d'exception

Exception	Type d'exception
IOException	Entrée Sortie (Exemple: lecture d'un fichier inexistant)
ArithmeticException	Arithmétique (Exemple: division par zéro)
ArrayIndexOutOfBoundsException	Dépassement des bornes d'un tableau
NumberFormatException	Format du nombre n'est pas respecté (Exemple: un int représenté en float)
OutOfMemoryException	Dépassement de mémoire

4. Importation

En Java, il existe **deux façons d'importer** des classes d'un package :

1.Importation explicite → On importe **uniquement** les classes nécessaires.

2.Importation implicite → Certaines classes sont accessibles **sans import**, ou via *.

4. Importation

Exemples:

1- Importations pour la gestion des exceptions

```
import java.lang.Exception;          // Classe de base des exceptions vérifiées
import java.lang.RuntimeException;   // Classe de base des exceptions non vérifiées
import java.io.IOException;          // Erreur d'entrée/sortie
import java.util.InputMismatchException; // Erreur de type lors de la saisie avec Scanner
import java.lang.NullPointerException;  // Accès à un objet null
import java.lang.IllegalArgumentException; // Argument invalide passé à une méthode
import java.lang.ArithmeticException;  // Erreur de calcul (ex: division par zéro)
```

4. Importation

Exemples:

2- Importations pour la saisie utilisateur

```
import java.util.Scanner; // Lecture des entrées utilisateur
```

4. Importation

Exemples:

3- Importations pour les collections

```
import java.util.List;          // Interface pour une liste (ArrayList, LinkedList)
import java.util.ArrayList;     // Liste dynamique
```