INF1256 Informatique pour les sciences de la gestion

– *Entrées-Sorties, Structures Sélectives et try* –

Johnny TSHEKE, Ing. Jr.

Université du Québec à Montréal
Département d'Informatique
tsheke_shele.johnny@uqam.ca

Séance 04

1 Introduction succincte sur les packages (Importation)

2 Saisie des données

3 Affichage des données

4 Structures sélectives : if et switch

5 Erreurs et exceptions

1 Introduction succincte sur les packages (Importation)

2 Saisie des données

3 Affichage des données

4 Structures sélectives : if et switch

5 Erreurs et exceptions

Classes et Packages (paquetages) en Java

En pratique :

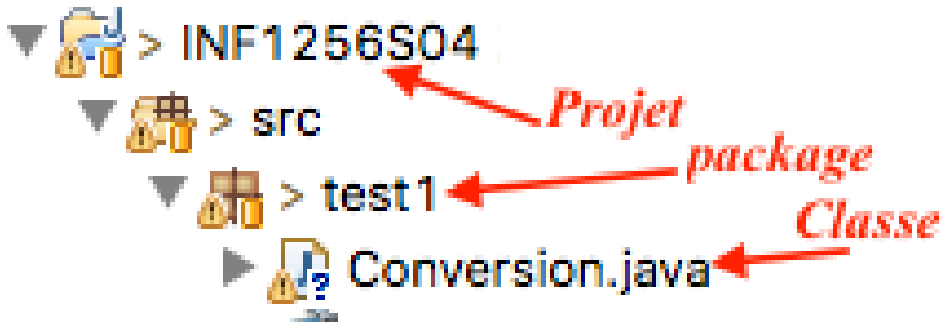
classe : Dans un fichier Java (ex : Conversion.java)

package : c'est un paquetage des classes. Concrètement, C'est un **dossier** qui contient :

- des classes et / ou
- des sous dossiers(sous `packages`)

Classes et Packages en Java : Illustration

Illustration des imbrications des éléments dans un projet Java



Importation des Classes

Permet de manipuler le type de données (méthodes, constantes, propriétés, etc) défini dans une classe dans une autre.

```
import nom_package.NomClasse; ou
import NomClasse;
```

Quelques cas particuliers

```
import NomClasse; // si dans le même package ou dossier
import nom_package.sous_package.NomClasse; // du sous package
import nom_package.*; // importer toutes les classes du  package
```

→ Pas d'extension (.java) dans `NomClasse` quand on importe
→ Importer classes / packages nécessaire pour son programme
→ On peut utiliser une classe importée comme un type de données :

```
NomClasse nc = new NomClasse(); // déclaration variable nc
```

⇒ L'importation se fait avant la déclaration de la classe qui importe

Quelques packages Java prédéfinis

- `java.io` pour les entrées / sorties (fichiers, ...)
- `java.lang` classes des types de données primitifs, `System`, ...
- `java.util` Classe `Scanner`, autres utilitaires
- `java.text` manipulation des textes, formatage, ...
- `java.math` précisions avec `BigInteger`, `BigDecimal`, ...
- `java.net` manipulation réseau, ...
- `java.sql` pour manipuler les bases de données relationnelles
- ...

→

`https://docs.oracle.com/javase/8/docs/api/overview-summary.html`

1 Introduction succincte sur les packages (Importation)

2 Saisie des données

3 Affichage des données

4 Structures sélectives : if et switch

5 Erreurs et exceptions

Entrée des données au clavier

⇒ Plus simple avec la classe Scanner

Importation → `import java.util.*;`

Déclaration variable → `Scanner clavier = new Scanner(System.in)`

Utilisation → `String nom = clavier.next();`

Fermeture → `clavier.close();// Quand on a fini d'utiliser`

⇒ La classe `Scanner` a plusieurs méthodes qui permettent de convertir les données directement au format approprié au moment de la lecture

Quelques méthodes de la classe Scanner

`next()` lire `String`

`nextBigDecimal` lire `BigDecimal`

`nextBigInteger()` lire `BigInteger`

`nextDouble()` lire `double`

`nextInt()` lire `Int`

`nextBoolean()` lire `Boolean`

Les méthodes `hasNextDouble()`, `hasNextInt()`, **hasNext()**, permettent de tester l'entrée

⇒ `https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html`

1 Introduction succincte sur les packages (Importation)

2 Saisie des données

3 Affichage des données

4 Structures sélectives : if et switch

5 Erreurs et exceptions

Affaichage simple

`System.out.print()` sans retour à la ligne

`System.out.println()` avec retour à la ligne

Exemple affichage sans formatage : AffichageSimple.java

```
package inf1256s04;

public class AffichageSimple {
    public static final double DIX = 10.0;
    public static final double TROIS = 3.0;
        public static void main(String[] args) {
                System.out.print("Afficher "+ DIX+"/"+TROIS+" = ");
                System.out.println(DIX/TROIS);
                System.out.println("Sans formatage ");

        }

}
```

L'exécution de cette classe donne le résultat suivant

```
Afficher 10.0/3.0 = 3.3333333333333335
Sans formatage
```

Affichage avec formatage

System.out.format("format",arguments) → dans la classe java.util.Formatter
format :

- `%[argument_index$][flags][width][.precision]conversion`
- début → % et fin → caractère de conversion. Les autres sont optionnels de droite vers la gauche
- `conversion` → `d` (entier), `f` (float), `s` (String), `n` (retour à la ligne), ...
- `precision` → nombre de chiffres après virgule
- `width` → nombre minimum de caractères à afficher
- `flag` → `-` (justifier à gauche), `+` (afficher signe), `0` (ajouter des 0), ...
- `argument_index` numéro argument sur la liste. se termine avec le signe de $

⇒`https://docs.oracle.com/javase/8/docs/api/java/util/Formatter.html#syntax`

Affichage avec formatage : Exemple

Exemple affichage avec formatage : AffichageFormatage.java

```
package inf1256s04;

public class AffichageFormatage {
    public static final double DIX = 10.0;
    public static final double TROIS = 3.0;
        public static void main(String[] args) {
                System.out.format("Afficher %2$.2f/%1$.2f = ",TROIS,DIX);
                System.out.format("%07.4f%n",DIX/TROIS);//avec retour à la ligne
                System.out.format("Avec formatage ");

        }

}
```

L'exécution de cette classe donne le résultat suivant

```
Afficher 10.00/3.00 = 03.3333
Avec formatage
```

1 Introduction succincte sur les packages (Importation)

2 Saisie des données

3 Affichage des données

4 Structures sélectives : if et switch

5 Erreurs et exceptions

Instruction if-else

Structure générale :

```
if(condition){ //block instructions
      }
else if (condition1){//block instructions
         }
else if (conditionN) {//block instructions
     }
else{ //block instructions
           }
```

On peut avoir :

- 0 ou plusieurs parties else if
- 0 ou une et une seule partie else

⇒ : L'évaluation des conditions (condition, condition1, conditionN) se fait selon l'ordre d'apparition.

⇒ : Éviter d'avoir deux conditions simultanément vraies.

Exemple if simple

Exemple if avec else

Exemple if ..else if ..else

Instruction switch

Comme l'instruction `if` mais plusieurs choix possibles

```
switch(expression){
case val1: //traitement cas 1
     break;
case val2: //traitement cas 2
     break;
case valN: //traitement cas N
     break;
default: //traitement par défaut
}
```

⇒ valeur de `expression` doit-être nombre, chaine, caractère ou énumération (`enum`)

⇒ On évalue l'`expression` et on exécute le case dont la valeur correspond. Sinon, on exécute default.

⇒ break ; à la fin de chaque case sinon, continue au case suivant.

⇒`http://www.javatpoint.com/java-switch`

Exemple switch

Exemple de sélection avec switch : CodeInternet.java

```
package inf1256s04;
import java.util.*;
public class CodeInternet {
	final static String CODE_FR="fr";
	final static String CODE_CA="ca";
	final static String CODE_BE="be";
	  public static void main(String[] args) {
		// TODO Auto-generated method stub
	 Scanner clavier = new Scanner(System.in);
	 System.out.println("Entrez un code Internet à 2 lettres");
	 String codeInt = clavier.next();
	 switch(codeInt){
	 case CODE_FR: System.out.println("France");
	     break;
	 case CODE_CA: System.out.println("Canada");
	     break;
	 case CODE_BE: System.out.println("Belgique");
	     break;
	 default: System.out.println("Je ne sais pas");
	 }
	  }

}
```

1 Introduction succincte sur les packages (Importation)

2 Saisie des données

3 Affichage des données

4 Structures sélectives : if et switch

5 Erreurs et exceptions

Contrôle d'erreurs de manière générale

- Exception → Erreur d'exécution et peut provoquer un arrêt ou un dysfonctionnement du programme
- On peut les contrôler avec l'instruction `try ...catch()`
- Au besoin, on peut aussi les créer volontairement avec l'instruction `throw new Exception("Message")`

Exemple vérification avec try de nombre réel strict en entrée (**clavier**) et création d'une exception

Exemple attraper et générer exception : verifierNombreReel4.java

```
package inf1256s04;
import java.util.*;
public class VerifierNombreReel4 {
        public static void main(String[] args) {
                double nombreReelStrict;
                Scanner clavier = new Scanner(System.in);
                System.out.println("Entrez un nombre réel strict -- pas entier svp");
                try{
                        try{
                        if(clavier.hasNextInt()){
                                //entier saisie, on génère une exception
                                throw new Exception("Nombre entier saisi");
                        }
                        nombreReelStrict = clavier.nextDouble();
                        }catch (Exception e) {
                                System.out.println("ERREUR: "+e.getMessage());
                                nombreReelStrict =clavier.nextDouble();
                        }
                        System.out.println("Le nombre réel obtenu est: "+↙
    ↳ nombreReelStrict);
                }catch (Exception e) {
                        System.out.println("Ce n'est pas un nombre réel");
                }
                clavier.close();
        }
}
```