TP 3 : Concepts Avancés en Java

Partie 1 : Package

- ✓ Un package est un groupe de classes (Bytecode, c.a.d. fichiers .class) associées à une fonctionnalité et/ou qui coopèrent.
- ✓ Le regroupement des classes dans des packages permet d'organiser les librairies de classes Java et d'éviter d'éventuels conflits de noms.
- ✓ L'instruction package est optionnelle ; si elle est présente elle doit être la première instruction du fichier.
- ✓ Si l'instruction package est omise, le code définit dans le fichier est associé à un package par défaut. Ce package par défaut n'a pas de nom et correspond au répertoire courant.

Préparer l'environnement de travail

1) Créer un répertoire D:\NomEtudiant\tpPackage.
2) Puis créer les deux répertoires **p1** et **p2** sous tpPackage
3) Dans p1 mettre la classe X, dans p2 mettre la classe Y

```
package p1;
public class X {
public void toto() {
System.out.println("toto de X");
}
public static void main(String args[]) {
System.out.println("main de X");
p2.Y y = new p2.Y();
y.toto();
X x = new X();
x.toto();
}}
```

```
package p2;
public class Y {
public void toto() {
System.out.println("toto de Y");
}
public static void main(String args[]) {
System.out.println("main de Y");
Y y = new Y();
y.toto();
p1.X x = new p1.X();
x.toto();
}}
```

Compiler et exécuter

1) Compiler sur l'éditeur Eclipse ou en mode console :
   - ✓ ***javac p1\X.java*** puis ***javac p2\Y.java***
   - ✓ On peut faire aussi ***javac p1\X.java p2\Y.java***
2) ***2)*** Exécution de la classe X sur Eclipse ou en mode console ***java p1.X***
3) ***3)*** Exécution de Y sur Eclipse ou en mode console ***java p2.Y***

Pour que le compilateur javac puisse accéder simplement aux classes définies dans les répertoires p1 et p2, il faut déclarer le chemin qui va jusqu'a son point d'attache qui est ici le répertoire tppackage. Cette déclaration est faite à l'aide du paramètre d'environnement CLASSPATH. Notez bien que ce doit être la racine et non le répertoire que l'on doit ajouter.
Notez également l'emploi du modificateur d'accès public, dans ce cas la classe est accessible en dehors du package auquel elle appartient.

Accès aux éléments d'un package

1) Essayez d'enlever le mot public dans la classe Y et compiler de nouveau. Une erreur est apparue dénotant l'impossibilité d'accéder Y en dehors du package P2.
2) Essayez de:
- ✓ mettre la classe Y dans le même répertoire que X,
- ✓ enlever le modificateur public des deux classes Y et X
- ✓ mettre dans l'entête de Y package p1 au lieu de p2. Compiler à nouveau. Aucune erreur n'aura lieu.

Avec le modificateur d'accès public, dans ce cas la classe est accessible en dehors du package auquel elle appartient. Sans modificateur d'accès, dans ce cas l'accès est dit package et la classe n'est accessible qu'aux autres classes du même package.

```
package p1;
class X {
public void toto() {
System.out.println("toto de X");
}
public static void main(String args[]) {
System.out.println("main de X");
Y y = new Y();
y.toto();
X x = new X();
x.toto();
} }
```

```
package p1;
class Y {
public void toto() {
System.out.println("toto de Y");
}
public static void main(String args[]) {
System.out.println("main de Y");
Y y = new Y();
y.toto();
X x = new X();
x.toto();
}}
```

Pour utiliser une classe publique depuis un package autre que celui où elle a été définie, il existe trois possibilités :

a) en indiquant le nom complet de la classe.
b) en important la classe,
c) en important tout le package où est définie la classe

On vous demande de simuler les trois possibilités sur machine.

Remarque : Le nom complet d'une classe est **nomDuPackage.nomDeLaClasse** en conséquence deux classes peuvent avoir le même nom dans des packages de noms différents.

Partie 2 : Héritage – Interface et Exception

On Considère un programme de gestion des personnels employés par une entreprise. On peut distinguer dans cette entreprise, les différents types de salariés suivants :

- ✓ les employés qui sont décrits par une matricule, un nom, un nombre d'heures et le prix d'une heure.
- ✓ les vendeurs sont des employés qui touchent en plus une commission = Ventes * Pourcentage
- ✓ à part les employés et les vendeurs, l'entreprise emploie une 3ème catégorie de salariés qui sont les directeurs. Ceux-ci sont des salariés qui touchent un salaire fixe augmenté d'une prime et ils dirigent un certain nombre d'employés.

Définir une application java capable de :

- ✓ Saisir toutes les informations associées à un salarié,
- ✓ Retourner toutes les informations associées à un salarié,
- ✓ Calculer la paye d'un salarié.
- ✓ A vous de définir les méthodes nécessaire pour chaque classe tous en appliquant les principes de la POO (encapsulation, héritage et polymorphisme). Identifier les différentes classes et donner leur hiérarchie.

Créer la classe TableSalaries :

Vous devez maintenant utiliser un tableau pour stocker l'ensemble des salariés de l'entreprise. A partir de ce tableau, vous devez pouvoir :

- ✓ ajouter un salarié donné dans le tableau
- ✓ supprimer un salarié de rang donné du tableau
- ✓ afficher les informations concernant tous les salariés
- ✓ calculer le total des payes à verser.

Ecrire un programme principal permettant de créer un tableau de salariés et de le gérer à l'aide du menu suivant :

- ✓ Ajouter un salarié
- ✓ Supprimer un salarié
- ✓ Afficher les salariés
- ✓ Total des payes à verser
- ✓ Quitter

Interface : Modifier la classe salarié, sans faire appel à l'héritage mais en utilisant une interface.

Gestion d'exceptions.

Définir 3 classes exceptions suivantes :

- ✓ la classe ErreurAjout : représente l'erreur qui survient lorsqu'on tente d'ajouter un salarié alors que le tableau est saturé.
- ✓ La classe ErreurSuppression : représente l'erreur qui survient lorsqu'on tente de supprimer un salarié alors que le tableau est vide.
- ✓ La classe ErreurRang : représente l'erreur qui survient lorsqu'on tente de supprimer un salarié en donnant un rang erroné.
- ✓ et mettre en œuvre le mécanisme de gestion de ces erreurs dans le programme.

Partie 3 : Classe anonyme, interface fonctionnelle, référence au méthode et expression Lambda

Soient les codes java suivants :

```
import java.util.Scanner;
public interface Modifiable {
int incrémenter (int val ,int inc);
int décrémenter (int val,int inc);
int raz (int val);  }
public class ProjetInitial {
public static void main(String[] args) {
Modifiable ent = new Modifiable (){
@Override
 public int incrémenter(int val, int inc) {
   return val+inc;}
@Override
 public int décrémenter(int val, int inc) {
   return val-inc;}
 @Override
```

```
Scanner sc = new Scanner(System.in);
System.out.println("Saisir un entier:");
int valeur = sc.nextInt();
valeur = ent.incrémenter(valeur, 3);
System.out.println("valeur entier: "+valeur);
valeur = ent.décrémenter(valeur, 5);
System.out.println("valeur entier: "+valeur);
 valeur=ent.raz(valeur);
 System.out.println("valeur entier: "+valeur);
  } }
```

```
public int raz(int val) { val = 0;
  return val;}   };
```

```
import java.util.Scanner;
public interface Modifiable {
void incrémenter ();
void décrémenter ();
void raz (); }
public abstract class Nombre implements Modifiable {
public abstract void saisir ();
public abstract void afficher (); }

public class Projet2 {
  public static void main(String[] args) {
 Modifiable point = new Modifiable () {
private int x = 5;
private int y = 6;
@Override
public void incrémenter () {
x++; y++;
System.out.println("["+x + " "+y+"]");}
@Override
public void décrémenter () { x--; y--;
System.out.println("["+x + " "+y+"]");}
@Override
public void raz () { x = 0; y = 0;
System.out.println("["+x + " "+y+"]");}
};
point.décrémenter();
point.raz();
point.incrémenter()
```

```
Nombre entier = new Nombre () {
private int valeur = 0;
public void saisir () {
System.out.println("Donnez un entier: ");
Scanner sc = new Scanner(System.in);
valeur = sc.nextInt();
}
public void afficher () {
System.out.println("Entier = "+valeur);
}
public void incrémenter () { valeur++; }
public void décrémenter () { valeur--; }
public void raz () { valeur = 0; }
};
entier.saisir ();
entier.afficher();
entier.incrémenter();
entier.afficher();
entier.décrémenter();
entier.afficher();
entier.raz();
entier.afficher();
   } }
```

1) Expliquez les codes précédents en repérant les définitions des classes anonymes
2) Remplacez dans le premier code l'interface Modifiable par des interfaces fonctionnelles, puis donnez un code équivalent dans lequel vous utilisez des expressions Lambda. L'incrémentation est conditionnelle : si le nombre est pair on ajoute l'incrément, sinon on ajoute le double de l'incrément.
   Testez aussi l'expression Lambda de l'incrémentation est insérée via une méthode statique de la classe principale.
3) Donnez un code équivalent du deuxième code et dans lequel, vous remplacez les classes anonymes par des classes nommées explicitement. Testez l'incrémentation avec une référence à la méthode