Université Paris 1 Panthéon Sorbonne	Epreuve de contrôle continu POO-Java Khelifi Tesnim	UFR de mathématiques et Informatique- UFR 27 L2 MIASHS

Exercice 1 : (5 pts)

1. Définir une interface ***Produit*** qui a comme méthode void AfficherModeEmploi().
2. Définir deux classes ***Livre*** et ***Ordinateur*** qui implémentent l'interface Produit, chacune des deux classes possède un attribut code de type chaîne de caractère. On ne vous demande pas d'écrire le constructeur.
3. Définir une classe ***Fabrique*** disposant d'une méthode statique CréerProduit(). Celle-ci accepte une chaîne de caractère en entrée représentant le code d'un produit. Cette méthode crée et retourne un objet Livre si le code commence par un 9 et crée et retourne un objet ordinateur si le code commence par 4. La méthode doit retourner l'objet NULL si le code est erroné.
4. Définir une classe ***Application*** qui ne manipule pas les classes Ordinateur et Livre, mais plutôt leur interface Produit et utilise les services de la classe Fabrique.
   Dans la méthode main, à partir d'une chaîne de caractère introduite au clavier, afficher le mode d'emploi du produit correspondant.
5. Donnez les concepts POO utilisés dans cet exercice.

Exercice 2 : (5 pts)

1 / Soit le programme suivant, quelles sont les exceptions prédéfinies qu'on peut utiliser pour que les erreurs susceptibles de se produire soient gérées.

```
class Exceptions1{
      static int division(int[] tab, int indice, int diviseur){
          return tab[indice]/diviseur;
}
public static void main(String[] args){
          Scanner scanner = new Scanner(System.in);
          int[] tableau = {17, 12, 15, 38, 29, 157, 89,-22, 0, 5};
          int x, y;
          System.out.print("Entrez l'indice de l'entier à diviser : ");
          int x = scanner.nextInt();

          System.out.print("Entrez le diviseur : ");
          int y = scanner.nextInt();

          System.out.print("Le résultat de la division est : ");
          System.out.println(division(tableau, x, y));
}
}
```

2/ Rappelez les définitions des exceptions vérifiées (checked exceptions) et les exceptions non vérifiées (unchecked exceptions).

Exercice 3 : (10 pts)

On s'intéresse à une entreprise de transport de bus. Pour cela, on implémente trois classes principales : la classe **Ville** (concerne les villes desservies par cette entreprise), **Ligne** assurant les liaisons entre les villes telles que Paris-Lyon, Paris-Marseille, Paris-Rouen, La troisième classe est **Tronçon** représente le segment routier reliant deux villes.

1. Définir une classe Ville caractérisée par un nom, la liste des tronçons entrant à la ville et une liste de tronçons sortant de la ville. Rajouter le constructeur et redéfinir les méthodes to String() et equals(), on teste l'égalité de deux villes par rapport à leurs noms.

2. Définir une classe Tronçon caractérisée par une ligne, une ville de départ et une ville destination et une distance. Exemple, parmi, les tronçons de la ligne Paris-Lyon, nous avons les tronçons Paris → Auxerre et Auxerre→ Dijon et Dijon → Lyon.
   Ecrire la méthode int *tronçonsConsécutifs(tronçon t)* qui retourne *1* si le tronçon courant est successeur à t, -1 s'il précède t et retourne 0 sinon. Les deux tronçons ne sont pas forcément de la même ligne.

3. Définir une classe Ligne caractérisée par un code séquentiel automatique, un nom, la ville de départ et la ville finale (terminus) et une liste de ses tronçons. On suppose qu'à la création de ligne aucun tronçon n'est spécifié. La liste des tronçons respecte l'ordre dans lequel les villes sont desservies.

   Rajouter à la classe Ligne, les méthodes qui permettent de :

   - Retourner la distance entre les villes de départ et terminus.
   - Vérifier si une ville v est desservie par la ligne courante.
   - Rajouter une ville v2 après une ville v1 (celle-ci est différente de la ville finale) à la ligne courante en créant les tronçons adéquats. Exemple, on peut rajouter la ville Auxerre après la ville Paris pour la ligne Paris-Dijon.
   - Supprime une ville v de la ligne courante. Les villes départ et terminus de la ville ne peuvent pas être supprimées. Toute tentative de suppression des extrémités d'une ligne donne lieu à une exception personnalisée que vous devez créer.

Quelques méthodes :

ArrayList : list.add(e) qui ajoute un objet e à la fin de la liste ;

list.add (i, e) qui ajoute un objet e à la position i

list.remove(e) qui supprime e de la liste

Collections.sort(list) qui trie la liste par ordre croissant

List.contains(e) teste l'existence de e dans la liste