Exercice 1. Les tableaux 1D

```
public static void main(String args[]) {
   int n=10 ;
   final int p=5 ;
   int t1[] = {1, 3, 5} ;
   int t2[] = {n-1, n, n+2} ;
   int t3[] = {p-2, p, p+3} ;
   int t4[] ;
   t4 = {1, 2, 3} ;
   float x1[] = {1, 2, p, p+7} ;
   float x2[] = {1.25, 2.5, 5} ;
}
```

- Quelles sont les erreurs glissées dans la fonction main ?
- Ecrire un programme qui crée un tableau comportant les valeurs des carrés des n premiers nombres pairs. Afficher le tableau via la syntaxe usuelle, puis via la syntaxe Java 5.
- Ecrire une classe utilitaire *UtilTab* pour manipuler les tableaux 1D de **double**. Cette classe devra contenir que des méthodes statiques (indépendantes de la création d'une instance) et devra fournir les services suivants : somme, moyenne, min, max, affichage, copie et `genereTabAlea(int nbCases)`. Ecrire une classe *TestUtilTab* pour tester toutes les méthodes de la classe *UtilTab*. *Math.random() génère des valeurs aléatoires dans [0 ;1[*.

Exercice 2. Les tableaux 2D

- Compléter la classe *UtilTab* précédente avec les services suivants pour les tableaux 2D de **float** : `somme`, `moyenne`, `sommeLigne(float tab[][], int indiceL)`, `sommeColonne(float tab[][], int indiceC)`, `affichage`, et `copie`.

Exercice 3. Les chaînes *String*

- Ecrire une méthode qui affiche un à un les caractères d'une chaîne de caractères passée en argument.
- Ecrire une méthode qui compte le nombre d'apparition d'une sous-chaîne dans une chaîne de caractères. Exemple : cette méthode avec la chaîne *abaabb* et la sous-chaîne *ab* doit retourner 2.
- Ecrire une classe Convert qui propose les services suivants :
  - conversion d'un type primitif passé en argument en String (valeur de retour) : `intToString`, `doubleToString` etc.
  - conversion d'un objet de type String en un type primitif : `stringToInt`, `stringToDouble` etc.
  - comparaison d'une valeur d'un type primitif et d'un objet de type String : `stringCompareToInt`, `stringCompareToDouble` etc. Elle doit retourner 0 si les 2 valeurs sont identiques, -1 si la chaîne contient une valeur plus petite et 1 sinon.

Exercice 4. Synthèse : gestion des étudiants inscrits à l'IUT

- Ecrire une classe *Etudiant* contenant le prénom, le nom et le numéro d'un étudiant (diagramme de classes proposé : http://yuml.me/edit/8a112c59).
    - Proposer un mécanisme pour associer automatiquement un numéro d'étudiant unique à toute nouvelle instance de la classe *Etudiant*.
- Ecrire une classe *RepertoireEtudiants* qui stocke un tableau d'étudiants : cette classe aura en plus du tableau un attribut *nbMax* désignant le nombre max d'étudiants que l'on peut stocker et un attribut *nbE* désignant le nombre d'étudiants actuellement stocké. Cette classe fournira les services suivants : `boolean ajouteEtudiant(Etudiant e)`, `int getNbEtudiant()`, `String getNom(int numeroEtu)`, `int getNumero(String nom)`, et `boolean supprimeEtudiant(Etudiant e)`. La méthode `getNumero` ne doit pas être sensible à la casse des caractères.
- Ecrire les commentaires *Javadoc* et générer la documentation *Javadoc* pour toutes les classes écrites pendant la réalisation de l'exercice 4.