TP 3 : LES CONCEPTS DE BASE DE LA POO

Exercice 1 :

1- Un compte bancaire est caractérisé par un numéro et solde, les opération possibles sur un compte sont : déposer de l'argent, retirer de l'argent, vérifier que le solde est suffisant pour retirer un x montant et afficher les information compte. Déterminer la représentation UML de la classe Compte
2- Créer une classe Compte
3- Créer une classe TestCompte, dans laquelle instancier deux objets de la classe compte :
   - cpt1(num : 101, solde : 2000)
   - cpt2(num : 102, solde ; 1000)
4- Déposer 1000 DT dans co1 et retirer 500dt de cpt2, puis afficher cpt1 et cpt2

Exercice 2 :

1- Définir la classe **Date** qui permet de représenter le format de date suivant : 16/03/1998. Cette classe doit contenir les méthodes suivantes :
   - **nombreJours** : Donne le nombre de jours pour le mois d'une Date
   - **dateValide** : Permet de vérifier si une Date est valide.
   - **lendemain** : Donne la Date de demain
2- Écrire la méthode *main* dans une classe **TesterDate** qui :
   a) Crée une date à partir des valeurs jour, mois et année introduits à partir de la ligne de commande
   b) Affiche cette date sous le format précisé en haut
   c) Si la date est valide
      - affiche le nombre de jours du mois de cette date
      - affiche la date de demain

      sinon
      - affiche un message d'erreur

Remarques :

1- Respecter le principe de l'encapsulation en déclarant les attributs et les méthodes.
2- Compléter la classe Date par d'autres méthodes si c'est nécessaire

Exercice 3 :

1- Créer la classe Temps qui contient 3 attributs entiers privés : heure, minute, seconde
2- Définir les constructeurs suivants en utilisant chaque fois que c'est possible la référence this
   - Temps (int heure) : construit un objet Temps à partir d'une heure donnée
   - Temps (int heure, int minute) : construit un objet Temps à partir des deux valeurs heure et minute

- Temps (int heure, int minute, int seconde) : construit un objet Temps à partir des trois valeurs : heure, minute, seconde

3- Définir une méthode affiche ( ) qui affiche le temps sous la forme suivante :
Il est ... heures ... minutes ... secondes

4- Qu'obtient-on à l'exécution si on ajoute dans cette classe la méthode *main* suivante :

```
public static void main (String [ ] args)
   {
         Temps t = new Temps (10);
         t.affiche( );

         t = new Temps (10,12);
         t.affiche( );

         t = new Temps (10, 12, 45);
         t.affiche( );
   }
```

Exercice 4 :

Définir une classe **Pile** permettant de représenter une pile d'entiers stockée sous forme d'un tableau.
Les méthodes à prévoir sont :
Constructeur de la pile recevant en argument la taille de la pile

- **void empile (int e)** : met l'entier e à la fin de la pile
- **void depile ( )** : enlève le dernier élément de la pile
- **void affiche ( )** : affiche le contenu de la pile si la pile contient des éléments sinon affiche le message "La pile est vide"

Indication :
Se servir des deux méthodes privées pilePleine ( ) et pileVide ( ) indiquant respectivement si la pile est pleine ou non et si la pile est vide ou non

Écrire une classe **TestPile** permettant de tester la création d'une pile de 10 entiers, appeler les méthodes *empile*, *depile* et *affiche.*

Exercice 5 :

Écrire une classe nommée **UneChaine** contenant un attribut privé **str** du type String et un constructeur recevant en argument une chaîne de caractères permettant d'initialiser str.
Les méthodes suivantes sont à prévoir :
String inverse ( ) : permet de renvoyer l'inverse de l'attribut str
boolean estPalindrome ( ) : permet de dire si oui ou non str est un palindrome (se lit de la même manière de droite à gauche et de gauche à droite)
Écrire une classe **EssaiChaine** permettant de tester la classe **UneChaine**.

Indication :

```
String str, ch ;
Int i ;
str.length(); // Retoune la taille de la chaîne de caractères
str.charAt(i); // Retourne un caractère à la position i de la chaine de caractères  str
str.equals(ch); // Retourne vrai si les deux chaines sont egales,  faux sinon
```

Exercice 6 :

Ecrire une classe nommée **Javanais** qui contient un attribut privé ***phrase*** de type chaîne de caractères, un constructeur permettant d'initialiser ***phrase*** et les méthodes suivantes :

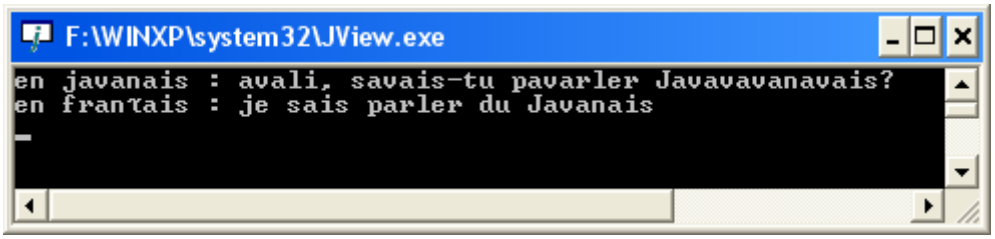
String versJavanais ( ) : retourne une chaîne de caractères résultat de la traduction de ***phrase*** du Français vers le Javanais
Traduire une phrase en Javanais revient à remplacer chaque lettre "a" de la phrase par "ava".
par exemple "ça va ?" est traduite par "çava vava ?"
String deJavanais ( ) : retourne une chaîne de caractères résultat de la traduction de ***phrase*** du Javanais vers le Français

Pour plus de précision on vous fournit la classe **TestJavanais** :

```
class TestJavanais {
      public static void main (String[] args) {
            Javanais j ;
            j = new Javanais ("ali, sais-tu parler Javanais? ");
            System.out.println ("en javanais : " + j.versJavanais( ) );

            j = new Javanais ("je savais pavarler du Javavavanavais ");
            System.out.println ("en français : " + j.deJavanais( ));
      }
 }
```

Sortie du programme :



Indication :

Vous pouvez faire appel à la méthode prédéfinie de la classe **String** :
boolean startsWith (String s, int i) : retourne true si la chaîne en question commence par la sous chaîne s à partir de la position i et false sinon
Exemple : `String ch = "bonjour " ;`
`ch.startstWith ("bon", 0) retourne true`
`ch.startsWith ("bon", 1) retourne false`