Corrigé du TP_TD N° 6

Exercice1 :

Les classes enveloppes (Integer, Double, Boolean, Byte, …) convertissent les types de données primitifs en objets.

Les avantages :

- Les objets sont nécessaires si l'on souhaite modifier les arguments passés dans une méthode (car les types primitifs sont passés par valeur).
- Les classes du paquetage java.util ne traitent que les objets et, par conséquent, les classes enveloppes sont également utiles.
- Les structures de données du Framework Collection, telles que ArrayList et Vector, stockent uniquement des objets (types de référence) et non des types primitifs.
- Un objet est nécessaire pour prendre en charge la synchronisation en multithreading (accès simultanés).

Les inconvénients :

- L'objet enveloppant utilise plus d'espace mémoire que le type primitif. Par exemple, un int prend 4 octets en mémoire mais un Integer utilisera 32 octets sur une machine virtuelle en 64 bits (20 octets en 32 bits).
- L'objet enveloppant est immuable, c'est à dire qu'il ne peut pas être modifié, toute modification de sa valeur nécessite de créer un nouvel objet et de détruire l'ancien, ce qui augmente le temps de calcul.

Exercice2 :

```
package test;
import java.util.*;
public class Test {
   public static void main(String[] args) {
     List<String> list = new ArrayList<String>();
      list.add("ali");
      list.add("zineb");
      list.add("wafa");
      list.add("ali");
      list.add("imane");
      System.out.println("Avant le tri :");
      for (String s : list) {
         System.out.println("\t"+s);
      }
      Collections.sort(list);
      System.out.println("Après le tri :");
      for (String s : list) {
         System.out.println("\t"+s);
}
}}
```

Exercice3 :

```
package GestionBanque;
public class Compte{
	static int compteur=0;
	private int Numero;
	private String Nom;
	private float Solde;
	private float DecouvertAutorise;

	public Compte(String nom,float solde,float decouvertAutorise){
		compteur++;
		this.Numero = compteur;Nom=nom;Solde=solde;DecouvertAutorise=decouvertAutorise;
	}
	public void Crediter(float montant){
		this.Solde+=montant;
	}
	public boolean Debiter(float montant){
		float nouveauMontant = this.Solde-montant;
		boolean ok=true;
		if(nouveauMontant>=this.DecouvertAutorise)
			this.Solde=nouveauMontant;
		else
			ok=false;
		return ok;
	}
	public static int getCompteur() {
		return compteur;
	}
	public int getNumero() {
		return Numero;
	}
	public String getNom() {
		return Nom;
	}
	public float getSolde() {
		return Solde;
	}
	public float getDecouvertAutorise() {
		return DecouvertAutorise;
	}
	public boolean Transferer(float montant, Compte autreCompte){
		boolean ok =true;
		if(this.Debiter(montant))
			autreCompte.Crediter(montant);
		else
			ok=false;
			return ok; }}
```

```
package GestionBanque;
import java.util.ArrayList;
public class Banque {
	private ArrayList<Compte> mesComptes= new ArrayList<Compte>();
	public ArrayList<Compte> getMesComptes() {
		return mesComptes;
	}
	public void setMesComptes(ArrayList<Compte> mesComptes) {
		this.mesComptes = mesComptes;   }}
```

```
package GestionBanque;
import java.util.*;

public class Main {

	public static void main(String[] args) {

		Banque b = new Banque();

		b.getMesComptes().add(new Compte("c1",2500,-1000));
		b.getMesComptes().add(new Compte("c2",800,-200));
		b.getMesComptes().add(new Compte("c3",12000,-1000));
		b.getMesComptes().add(new Compte("c4",20000,-400));
		b.getMesComptes().add(new Compte("c5",6000,-600));
		for (Compte s : b.getMesComptes()) {
		System.out.println("Numéro:"+s.getNumero()+" Nom :"+s.getNom()+" Solde :"+s.getSolde()+"
DH"+" Decouvert :"+s.getDecouvertAutorise()+" DH");
		}
	b.getMesComptes().removeIf(p-> p.getNumero()==1); // supprimer le compte qui possède le numéro 1
	b.getMesComptes().removeIf(p-> p.getNumero()==2); // supprimer le compte qui possède le numéro 2
		for (Compte s : b.getMesComptes()) {
	System.out.println("Numéro:"+s.getNumero()+" Nom :"+s.getNom()+" Solde :"+s.getSolde()+" DH"+"
Decouvert :"+s.getDecouvertAutorise()+" DH");
		}
		b.getMesComptes().get(0).Crediter(100000);
		b.getMesComptes().get(1).Debiter(12);
		b.getMesComptes().get(0).Transferer(1000,  b.getMesComptes().get(1));
		for (Compte s : b.getMesComptes()) {
			System.out.println("Numéro:"+s.getNumero()+" Nom :"+s.getNom()+" Solde
:"+s.getSolde()+" DH"+" Decouvert :"+s.getDecouvertAutorise()+" DH");
}
}
}
```