Rapport TAP TP5

Exercice 1

Dans ce TP nous avons créer des graphes en C# grâce à l'utilisation de Ienumerable. Nous avons ensuite affiché la liste des arêtes du graphe :

```
graph.ForEach(x =>

x.neighbors.ForEach(n=>Console.WriteLine("("+x.data+"-"+n.data+")"
)));
```

En utilisant des lambdas expressions, nous avons pu affiché l'ensemble des résultats.

Exercice 2

Dans l'exercice 2, nous avons compris que l'héritage se faisait entre les types de la collection pas entre les différentes collections (List<Orange> et List<Fruit> il n'y a pas d 'héritage, même si Orange hérite de Fruit).
Nous avons aussi utilisé le wildcard en java. Nous avons eu les résultats suivants qui ne marchent pas

```
 * Exercice 2
*question 2
 */
//First stattement
/*public static void f(List<? extends Number> list,Number m)
{
	list.add(m);
}*/

//Second statement
/*public static void g(List<? super Number> list,Number m)
{
	list.get(m);
}*/

//Third statement
/*public static void h(List<?> list,Number m)
{
	list.get(m);
	list.add(m);
}*/
```

Par la suite, nous avons implémenter une liste de listes en une unique liste :

```
public static <T> List<T> GFlatten(List<List<T>> toFlat)
{
```

```
        if(toFlat.isEmpty())
            return toFlat.get(0);

        List<T> list = toFlat.get(0);
        for(int i=1;i<toFlat.size();i++)
            list.addAll(toFlat.get(i));

        return list;
    }
```

Nous avons rendu notre fonction le plus générique possible.

Exercice 3

Dans l'exercice 3 vous avons implémenté le design pattern Visitor en utilisant le delegate de C#. Nous avons notre classe CompositeFigure dans laquelle nous avons implémenté la fonction accept de l interface Ifigure.

```
public class CompositeFigure : IFigure {

    public CompositeFigure(List<IFigure> list) {this.list = list;}

    private List<IFigure> list;

    public String GetName(){
        String name = "";
        foreach( IFigure sf in list ){
            name+= sf.GetName() + " ";
        }
        return name;
    }


    public T Accept<T>(VisitorFun<IFigure, T> v){

        T result = v(this);
        foreach( IFigure sf in list ){
                result = sf.Accept(v);
        }
        return result;
    }

}
```