TD4

Exercice 1

```
Map<Double, String> employe =
l.stream().collect(Collectors.toMap(Employe::getSalaire, Employe::getNom));
System.out.println(employe);
```

Exercise 2

```
import java.util.ArrayList;
import java.util.List;
import java.util.function.ToLongFunction;

public class addition {

    public static void main(String[] args) {

        ToLongFunction<Integer> somme = add(3);
        System.out.println(somme.applyAsLong(5));

        List<Integer> list = new ArrayList<Integer>();
        list.add(1);
        list.add(4);
        list.add(7);
        list.add(5);
        list.forEach((num) ->
System.out.println(add(num).applyAsLong(78)));
    }

    public static ToLongFunction<Integer> add(Integer calcul){

        ToLongFunction<Integer> result = (value) -> value + calcul;

        return result;
    }
}
```

Exercise 3

```
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

public class Test {

    public static void main(String[] args) throws EmployeException {
        Entreprise e1 = new Entreprise("IBM");
        e1.ajouter(new Employe("Dupond", 15000));
        e1.ajouter(new Employe("Poiret", 16000));
        e1.ajouter(new Employe("Burot", 15700));
        e1.ajouter(new Employe("Pernaut", 14300));
```

```
        System.out.println(e1);

        // Remplit une liste avec les employés
        List<Employe> l = new ArrayList<>();
        for (Employe e : e1) {
            l.add(e);
        }

        Map<Double, String> employe = l.stream().filter((Employe) ->
Employe.getSalaire() >=
15000).collect(Collectors.toMap(Employe::getSalaire, Employe::getNom));
        System.out.println(employe);
    }
}
```