Exercice1 :

package Exercice1;  
import java.util.Arrays;  
import java.util.List;  
import java.util.stream.Collectors;  
import java.util.stream.IntStream;  
  
public class App {  
 public static void main(String[] args) {  
 List<String> mots =Arrays.asList("chat", "chien", "lapin", "oiseau", "elephant", "tigre", "lion");  
 // Q1  
 System.*out*.println("\*\*\*\* Q1 \*\*\*\*\*");  
 mots.stream()  
 .filter(mot -> mot.contains("a"))  
 .collect(Collectors.toList())  
 .forEach(System.*out*::println);  
  
 //Q2  
 System.*out*.println("\*\*\*\* Q2 \*\*\*\*\*");  
 mots.stream().filter(mot->mot.length()>3)  
 .map(mot->new StringBuilder(mot).reverse())  
 .forEach(System.*out*::println);  
  
 //Q3  
 System.*out*.println("\*\*\*\* Q3 \*\*\*\*\*");  
 /\*mots.stream().filter(mot -> mot.contains("e"))  
 .flatMap(mot-> Stream.of(mot.toCharArray()))  
 .forEach(System.out::println);\*/  
  
 mots.stream()  
 .filter(mot -> mot.contains("e"))  
 .flatMap(mot -> mot.chars().mapToObj(c -> (char) c))  
 .collect(Collectors.toList())  
 .forEach(System.*out*::println);  
  
 //Q4  
 System.*out*.println("\*\*\*\* Q4 \*\*\*\*\*");  
 mots.stream()  
 //.map(String::toUpperCase)  
 .map(String::toUpperCase)  
 .forEach(System.*out*::println);  
  
 //Q5  
 System.*out*.println("\*\*\*\* Q5 \*\*\*\*\*");  
 mots.stream()  
 .mapToInt(String::length)  
 .forEach(System.*out*::println);  
  
 //Q6  
 System.*out*.println("\*\*\*\* Q6 \*\*\*\*\*");  
 mots.stream()  
 .flatMap(mot -> mot.chars().mapToObj(c -> (char) c))  
 .forEach(System.*out*::println);  
  
 //Q7  
 System.*out*.println("\*\*\*\* Q7 \*\*\*\*\*");  
 /\*mots.stream()  
 .map((mot, index) -> mots.get(index)+ " - " +index)  
 .collect(Collectors.toList());\*/  
  
 IntStream.range(0, mots.size())  
 .mapToObj(index -> mots.get(index) + " - " + index)  
 .collect(Collectors.toList())  
 .forEach(System.*out*::println);  
  
 }  
}

\*\*\*\* Q1 \*\*\*\*\*

chat

lapin

oiseau

elephant

\*\*\*\* Q2 \*\*\*\*\*

tahc

neihc

nipal

uaesio

tnahpele

ergit

noil

\*\*\*\* Q3 \*\*\*\*\*

c

h

i

e

n

o

i

s

e

a

u

e

l

e

p

h

a

n

t

t

i

g

r

e

\*\*\*\* Q4 \*\*\*\*\*

CHAT

CHIEN

LAPIN

OISEAU

ELEPHANT

TIGRE

LION

\*\*\*\* Q5 \*\*\*\*\*

4

5

5

6

8

5

4

\*\*\*\* Q6 \*\*\*\*\*

c

h

a

t

c

h

i

e

n

l

a

p

i

n

o

i

s

e

a

u

e

l

e

p

h

a

n

t

t

i

g

r

e

l

i

o

n

\*\*\*\* Q7 \*\*\*\*\*

chat - 0

chien - 1

lapin - 2

oiseau - 3

elephant - 4

tigre - 5

lion - 6

Process finished with exit code 0

Exercice 2

package Exercice2;  
  
public class Employe {  
 private String name;  
 private String department;  
 private double salary;  
  
 public java.lang.String getName() {  
 return name;  
 }  
  
 public void setName(java.lang.String name) {  
 this.name = name;  
 }  
  
 public java.lang.String getDepartment() {  
 return department;  
 }  
  
 public void setDepartment(java.lang.String department) {  
 this.department = department;  
 }  
  
 public double getSalary() {  
 return salary;  
 }  
  
 public void setSalary(double salary) {  
 this.salary = salary;  
 }  
  
 public Employe() {  
 }  
  
 public Employe(java.lang.String name, java.lang.String department, double salary) {  
 this.name = name;  
 this.department = department;  
 this.salary = salary;  
 }  
}

package Exercice2;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Optional;  
import java.util.stream.Collectors;  
  
public class App {  
 public static void main(String[] args) {  
  
 List<Employe> employees = new ArrayList<>();  
 employees.add(new Employe("Ali", "Finance", 5000));  
 employees.add(new Employe("Mohammed", "IT", 6000));  
 employees.add(new Employe("Yassin", "HR", 4500));  
 employees.add(new Employe("Reda", "Finance", 5500));  
  
 // Q1  
 double totalSalary = employees.stream()  
 .mapToDouble(Employe::getSalary)  
 .sum();  
 System.*out*.println("Somme totale des salaires : " + totalSalary);  
  
 // Q2  
 List<Employe> sortedEmployees = employees.stream()  
 .sorted((e1, e2) -> e1.getName().compareTo(e2.getName()))  
 .collect(Collectors.toList());  
 System.*out*.println("Liste des employés triés par nom : " + sortedEmployees);  
  
 // Q3  
 Optional<Employe> minSalaryEmployee = employees.stream()  
 .min((e1, e2) -> Double.*compare*(e1.getSalary(), e2.getSalary()));  
 minSalaryEmployee.ifPresent(employee -> System.*out*.println("Employé avec le salaire le plus bas : " + employee.getName() + " - " + employee.getSalary()));  
  
 // Q4  
 double thresholdSalary = 5000;  
 List<Employe> highSalaryEmployees = employees.stream()  
 .filter(employee -> employee.getSalary() > thresholdSalary)  
 .collect(Collectors.toList());  
 System.*out*.println("Liste des employés avec un salaire supérieur à " + thresholdSalary + " : " + highSalaryEmployees);  
  
 //Q5  
 Optional<Employe> maxSalaryEmployee = employees.stream()  
 .reduce((e1, e2) -> e1.getSalary() > e2.getSalary() ? e1 : e2);  
 maxSalaryEmployee.ifPresent(employee -> System.*out*.println("Employé avec le salaire le plus élevé : " + employee.getName() + " - " + employee.getSalary()));  
  
 // Q6  
 String concatenatedNames = employees.stream()  
 .map(Employe::getName)  
 .reduce((name1, name2) -> name1 + ", " + name2)  
 .orElse("");  
 System.*out*.println("Noms concaténés des employés : " + concatenatedNames);  
 }  
}

Somme totale des salaires : 21000.0

Liste des employés triés par nom : [Exercice2.Employe@6e8cf4c6, Exercice2.Employe@12edcd21, Exercice2.Employe@34c45dca, Exercice2.Employe@52cc8049]

Employé avec le salaire le plus bas : Yassin - 4500.0

Liste des employés avec un salaire supérieur à 5000.0 : [Exercice2.Employe@12edcd21, Exercice2.Employe@34c45dca]

Employé avec le salaire le plus élevé : Mohammed - 6000.0

Noms concaténés des employés : Ali, Mohammed, Yassin, Reda

Process finished with exit code 0