**Tp « Stream »**

**Exercice 1 :**

Coder la traiteList(List<Point> liste, Predicate<Point> predicate, Comparator<Point> comparator, Consumer<Point> consumer) à l’aide des Streams ;

**Exercice 2 :**

Soit la classe Dish :

**public** **class** Dish {

**private** **final** String name;

**private** **final** **boolean** vegetarian;

**private** **final** **int** calories;

**private** **final** Type type;

**public** Dish(String name, **boolean** vegetarian, **int** calories, Type type) {

**this**.name = name;

**this**.vegetarian = vegetarian;

**this**.calories = calories;

**this**.type = type;

}

**public** String getName() {

**return** name;

}

**public** **boolean** isVegetarian() {

**return** vegetarian;

}

**public** **int** getCalories() {

**return** calories;

}

**public** Type getType() {

**return** type;

}

**public** **enum** Type {

***MEAT***,

***FISH***,

***OTHER***

}

@Override

**public** String toString() {

**return** name;

}

**public** **static** **final** List<Dish> ***menu*** = Arrays.*asList*(

**new** Dish("pork", **false**, 800, Dish.Type.***MEAT***),

**new** Dish("beef", **false**, 700, Dish.Type.***MEAT***),

**new** Dish("chicken", **false**, 400, Dish.Type.***MEAT***),

**new** Dish("french fries", **true**, 530, Dish.Type.***OTHER***),

**new** Dish("rice", **true**, 350, Dish.Type.***OTHER***),

**new** Dish("season fruit", **true**, 120, Dish.Type.***OTHER***),

**new** Dish("pizza", **true**, 550, Dish.Type.***OTHER***),

**new** Dish("prawns", **false**, 400, Dish.Type.***FISH***),

**new** Dish("salmon", **false**, 450, Dish.Type.***FISH***)

);

}

Soit la méthode

**public** **static** List<String> getLowCaloricDishesNames (List<Dish> dishes) {

List<Dish> lowCaloricDishes = **new** ArrayList<>();

**for** (Dish d : dishes) {

**if** (d.getCalories() < 400) {

lowCaloricDishes.add(d);

}

}

List<String> lowCaloricDishesName = **new** ArrayList<>();

Collections.*sort*(lowCaloricDishes, **new** Comparator<Dish>() {

@Override

**public** **int** compare(Dish d1, Dish d2) {

**return** Integer.*compare*(d1.getCalories(), d2.getCalories());

}

});

**for** (Dish d : lowCaloricDishes) {

lowCaloricDishesName.add(d.getName());

}

**return** lowCaloricDishesName;

}

Déclarer la méthode getLowCaloricDishesNames en utilisant les streams ;

**Exercice 3**

Interpréter le code suivant :

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

***menu***.forEach(System.***out***::println);

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

List<Dish> vegetarianMenu = ***menu***.stream()

.filter(Dish::isVegetarian)

.collect(*toList*());

vegetarianMenu.forEach(System.***out***::println);

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

List<Integer> numbers = Arrays.*asList*(1, 2, 1, 3, 3, 2, 4);

numbers.stream()

.filter(i -> i % 2 == 0)

.distinct()

.forEach(System.***out***::println);

List<Dish> specialMenu = Arrays.*asList*(

**new** Dish("season fruit", **true**, 120, Dish.Type.***OTHER***),

**new** Dish("prawns", **false**, 300, Dish.Type.***FISH***),

**new** Dish("rice", **true**, 350, Dish.Type.***OTHER***),

**new** Dish("chicken", **false**, 400, Dish.Type.***MEAT***),

**new** Dish("french fries", **true**, 530, Dish.Type.***OTHER***));

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

List<Dish> filteredMenu = specialMenu.stream()

.filter(dish -> dish.getCalories() < 320)

.collect(*toList*());

filteredMenu.forEach(System.***out***::println);

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

List<Dish> slicedMenu1 = specialMenu.stream()

.takeWhile(dish -> dish.getCalories() < 320)

.collect(*toList*());

slicedMenu1.forEach(System.***out***::println);

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

List<Dish> slicedMenu2 = specialMenu.stream()

.dropWhile(dish -> dish.getCalories() < 320)

.collect(*toList*());

slicedMenu2.forEach(System.***out***::println);

List<Dish> dishesLimit3 = ***menu***.stream()

.filter(d -> d.getCalories() > 300)

.limit(3)

.collect(*toList*());

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

dishesLimit3.forEach(System.***out***::println);

List<Dish> dishesSkip2 = ***menu***.stream()

.filter(d -> d.getCalories() > 300)

.skip(2)

.collect(*toList*());

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

dishesSkip2.forEach(System.***out***::println);

**Exercice 4 :**

Interpréter les fonctions suivantes :

// Stream.of

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Stream<String> stream = Stream.*of*("IIR", "Casablaca", "Maroc", "");

stream.map(String::toUpperCase).forEach(System.***out***::println);

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Stream<String> emptyStream = Stream.*empty*();

emptyStream.forEach(System.***out***::println);

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

**int**[] numbers = { 2, 3, 5, 7, 11, 13 };

System.***out***.println(Arrays.*stream*(numbers).sum());

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Stream.*iterate*(0, n -> n + 2)

.limit(10)

.forEach(System.***out***::println);

Stream.*generate*(Math::*random*)

.limit(10)

.forEach(System.***out***::println);

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

IntStream.*generate*(() -> 1)

.limit(5)

.forEach(System.***out***::println);