Learning objective: manipulate Java Streams (sequential data-processing operations and basic collectors).

We assume that a class Dish is defined with the methods: getType, isVegetarian, getCalories and getName. All dishes are stored into menu, which is a list.

Exercise 1: external vs. internal iteration
Which stream operation would you use to refactor the following code?

```java
List<Dish> vegetarianDishes = new ArrayList<>();
for (Dish d: menu) {
    if (d.isVegetarian())
        vegetarianDishes.add(d);
}
```

What would be the resulting code?

Exercise 2: external vs. internal iteration
Which stream operation would you use to refactor the following code?

```java
List<String> highCaloricDishes = new ArrayList<>();
Iterator<String> iterator = menu.iterator();
while (iterator.hasNext()) {
    Dish dish = iterator.next();
    if (dish.getCalories() > 300)
        highCaloricDishes.add(d.getName());
}
```

What would be the resulting code?

Exercise 3: mapping
Given a list of numbers, how would you return a list of the square of each number? For example, given [1, 2, 3, 4, 5] you should return [1, 4, 9, 16, 25].

Exercise 4: filtering/slicing
Given a list of Dish with method getType, write the stream that puts in a list the first two meat dishes?

Exercise 5: nested streams
Write a stream that puts in a list only the integers from a first list that are contained in second one (like List::retainAll). For example, given the lists [1, 2, 3] and [1, 2, 4], the resulting list is [1, 2].

Exercise 6: flatMap
Given a list of lists of numbers, how would you return a single flattened list of all nested numbers? For example, given a list [[1, 2, 3], [2, 3, 4], [3, 4, 5]], you should return [1, 2, 3, 2, 3, 4, 3, 4, 5].

Exercise 7: flatMap
Given a list of strings, how would you return a list of all distinct sorted capitalized characters? For example, given two strings 'dcb' and 'abc', you should return ['A', 'B', 'C', 'D'].

Exercise 8: flatMap

Given two lists of numbers, how would you return all pairs of numbers? For example, given a list [1, 2, 3] and a list [3, 4] you should return [(1, 3), (1, 4), (2, 3), (2, 4), (3, 3), (3, 4)]. For simplicity, you can represent a pair as an array with two elements.

How would you extend the previous example to return only pairs whose sum is divisible by 3?

Exercise 9: reducing

How would you count the number of dishes in a stream using the map and reduce methods (i.e. without count)?

Exercise 10: intermediate vs. terminal operations

In the stream pipeline that follows, identify the intermediate and terminal operations?

```java
long count = menu.stream()
    .filter(dish -> dish.getCalories() > 300)
    .distinct()
    .limit(3)
    .count();
```

Exercise 11: intermediate vs. terminal operations

Which of the following operations is terminal? dropWhile - map - skip - collect

Exercise 12: short-circuiting

Which of the following operations allows short-circuiting? skip - findFirst - dropWhile - filter

Exercise 13: equivalence

Is it possible to rewrite any instruction that uses anyMatch with either noneMatch or allMatch?