Lab Exercises 10

Dr. Sarvar Abdullaev
`s.abdullaev@inha.uz`

December 26, 2020

You must attempt all exercises given in this lab. After completing it, you must push it to your corresponding GitHub Classroom repository. Note, you do not have to upload compiled Java bytecode or screenshot of your program's output.

1 Generic `isEqualTo` Method

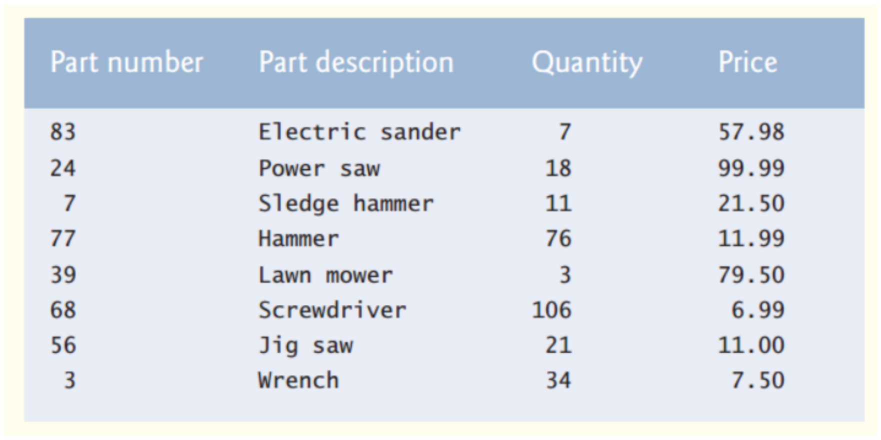
Write a simple generic version of method `isEqualTo` that compares its two arguments with the `equals` method and returns true if they're equal and false otherwise. Use this generic method in a program that calls `isEqualTo` with a variety of built-in types, such as `Object` or `Integer`. What result do you get when you attempt to run this program?

2 Generic Class `Pair`

Write a generic class `Pair` which has two type parameters—`F` and `S`—each representing the type of the first and second element of the pair, respectively. Add `get` and `set` methods for the first and second elements of the pair. [Hint: The class header should be public class `Pair<F, S>`.]

3 Manipulating a `Stream<Invoice>`

Create a class `Invoice` which includes four properties—a `PartNumber` (type `int`), a `PartDescription` (type `String`), a `Quantity` of the item being purchased (type `int`) and a `Price` (type `double`). Use the sample data shown in Fig. 1 to create an array of `Invoice` objects.



Part number	Part description	Quantity	Price
83	Electric sander	7	57.98
24	Power saw	18	99.99
7	Sledge hammer	11	21.50
77	Hammer	76	11.99
39	Lawn mower	3	79.50
68	Screwdriver	106	6.99
56	Jig saw	21	11.00
3	Wrench	34	7.50

Figure 1: Invoices

Perform the following queries on the array of `Invoice` objects and display the results:

1. Use lambdas and streams to sort the `Invoice` objects by `PartDescription`, then display the results.
2. Use lambdas and streams to sort the `Invoice` objects by `Price`, then display the results.
3. Use lambdas and streams to map each `Invoice` to its `PartDescription` and `Quantity`, sort the results by `Quantity`, then display the results.
4. Use lambdas and streams to map each `Invoice` to its `PartDescription` and the value of the `Invoice` (i.e., `Quantity * Price`). Order the results by `Invoice` value.
5. Modify previous task to select the `Invoice` values in the range `$200` to `$500`.

4 Duplicate Word Removal

Write a program that inputs a sentence from the user (assume no punctuation), then determines and displays the unique words in alphabetical order. Treat uppercase and lowercase letters the same.

5 Sorting Letters and Removing Duplicates

Write a program that inserts 30 random letters into a `List<Character>`. Perform the following operations and display your results:

1. Sort the `List` in ascending order.
2. Sort the `List` in descending order.
3. Display the `List` in ascending order with duplicates removed.