Department of Informatics

Professor: R. Ingold
Teaching Assistants: M. Alberti, L. Vögtlin
Assistant: R. Geissbühler

Assignment 11

1 Theory

In this exercise you have to answer the following theoretical questions. Keep your answers short and precise.

(a) What is a collection from a general point of view and from a Java point of view?

(b) What are the basic operations offered by a `Collection`?

(c) What is an iterator? Why is it useful?

(d) What is the difference of iterating a `Collection` with a `for each` or with the iterator explicitly ?

(e) What are the differences between a `Set` and a `List` ?

(f) What methods should be correctly implemented on elements inserted into an `Hashset` or `Hashmap` ?

(g) What are the differences between a queue and a deque?

(h) What is the minimal requirement in Java for an object to become ordered?

(i) Provide the exact link to the Javadoc for the class `Collections`. *Hint: the Javadoc is located at* `docs.oracle.com/` *but is easier to access it trough Google directly!*

(j) Mention 5 different methods of the class `Collections` which address very common situation and hence are very useful.

2 Implementation

For this exercise you are required to submit a file: Main.java, implemented as described below.

(a) Implement a `Main` class with a `main()` method which creates an `ArrayList` and a `LinkedList` of `Integer`. Populate both lists with 10′000 random numbers.

(b) In a `for loop` select 1′000′000 (in the code you can write `1e6`) random elements of the `ArrayList` and integrate their value (sum all of them). Measure the time necessary to do this operation with `System.nanoTime()`.

(c) Do the very same thing for the `LinkedList`.

(d) Compare the time required for doing the same operation of the two types of lists. If there is a difference explain why this happens. If there is no difference increase the number of elements in the list to 1′000′000 and try again.

3 Debugging

In this exercise we want to practice the use of the debugger and familiarize with the error messages of the compiler. Refer to the files `Main.java` attached to this series for this exercise.

(a) Run the `Main` file. Notice that error are risen in `removeEvilEvenNumbers` 2 and 3.

(b) Report the *exact* error message and explain why is this happening for both cases.

(c) Fix `removeEvilEvenNumbers2` by modifying the code such that it makes explicit use of iterators.

(d) Make use of the debugger to find out why `removeEvilEvenNumbers1` works, even though is very similar to `removeEvilEvenNumbers3` counterpart. *Hint: inspect the boundaries of the loop at different iterations.*

(e) This point is not mandatory, but if you feel like trying something out go and replace the slow iteration in `removeEvilEvenNumbers3` by making use of the `Collection` method `.removeif()`. Here I Googled it for you: https://bit.ly/2uXILrM and https://bit.ly/2qilThf.

For reference (you don't have to rewrite them yourself, we provide the java files attached to the series):

```java
package core;

import java.util.ArrayList;
import java.util.Random;

public class Main {
    public static void main(String[] args) {
        Random random = new Random();
        ArrayList<Integer> arrayList = new ArrayList<>();
        for (int i = 0; i < 10000; i++) {
            arrayList.add(random.nextInt(10));
        }

        removeEvilEvenNumbers1(arrayList);
        removeEvilEvenNumbers2(arrayList);
        removeEvilEvenNumbers3(arrayList);

    }

    private static void removeEvilEvenNumbers1(ArrayList<Integer> arrayList) {
        for (int i = 0; i < arrayList.size(); i++) {
            if(arrayList.get(i) % 2 == 0){
                arrayList.remove(i);
            }
        }
    }

    private static void removeEvilEvenNumbers2(ArrayList<Integer> arrayList) {
        for(int i : arrayList){
            if(arrayList.get(i) % 2 == 0){
                arrayList.remove(i);
            }
        }
    }

    private static void removeEvilEvenNumbers3(ArrayList<Integer> arrayList) {
        int n = arrayList.size();
        for (int i = 0; i < n; i++) {
            if(arrayList.get(i) % 2 == 0){
                arrayList.remove(i);
            }
        }
    }
}
```

4 Bonus exercise

Submit a text where you motivate the design choices you made in your project. This can be done with the help of a diagram/images or just plain text but you need at least a diagram of your Game Object hierarchy. In particular focus on those choices that you made yourself and try to elaborate on whether in retrospective (after fully developing the project) you are happy with them or not (e.g. would you make them again or change something?). This is a introspective work and should be longer than 1000 words of plain text.

5 Project

Your Space Invaders game should now be fully functional. You have no time until the

19.05.2019 23:59 CET

to polish your project (add some exceptions, polish the GUI, create better looking sprites, etc.) and hand it in on moodle. Upload just the final version of your project including all the files.
For those who want to have a runnable JAR (a single file which you can share with others) check out this link.

5.1 listIterator() of Series 10

If some of you did not managed to implemented the `listIterator()` of series 10 here the pseudocode:

```
Node<T> n = head
while index-- > 0:
    n = n.next

return new Iterator<>(n);
```