Homework 4 (Due: *10 June 2021 at 6am*)

Submit in CSCMS (zipped folder containing exactly 1 ***.pdf*** file for Q-4a and 1 ***.java*** file for Q-4b.)

Q-4a (5 pts). Consider the code below, where you can find definitions of the related classes and variables in the accompanied ***.zip*** file.

```
List<String> tlst =
   nums.stream()
       .filter( num -> { System.out.println("1st char = t?");
                         return num.word.startsWith("t"); } )
       .map( num -> { System.out.println("Yes! Extract Num.word");
                      return num.word;} )
       .limit(2)
       .map( s -> {System.out.println("Finalist: " + s);
                   return s; } )
       .collect( toList() );

System.out.println( tlst );
```

The Experiment:

1. Examine the given code carefully, write down what you think will get printed. *Write it down as part of your answer.*
2. Replace the intermediate operation, **limit(2)**, with **skip(1)**. What do you think will be printed now? *Write it down as part of you answer*.
3. Write Java code to execute the code above. Execute it **twice**: one for the code with **limit(2)**, and the other with **skip(1)**. *Write down as your answer what actually gets printed*.
4. Compare the actual print-out against what you think would get printed. Are they same? Are they different? What do you think the key message is that this experiment tries to deliver? Are there any other observations? **Discuss** and *write down your discussion as part of the answer to be submitted*.

Q-4b (5 pts). Consider the code below, where you can find definitions of the related classes and variables in the accompanied ***.zip*** file. You task in this question is to refactor the given code using the Streams API and lambda expressions. Write a Java application to verify that your refactored code works correctly. Here is the additional requirements:

- You must write one (1) Java file only. This file should compile together with the provided Java code to produce the executable Java application;
- **Extra credits (2 pts)**: Prepare this Java file in such a way that both the original code (method) and the refactored code (method) co-exist in the same class, ***without*** having to change any of the method names.

```
// Here is the code to be refactored
public Set<String> findLongTracks(List<Album> albums) {
    Set<String> trackNames = new HashSet<>();
    for(Album album : albums) {
        for (Track track : album.getTrackList()) {
            if (track.getLength() > 60) {
                String name = track.getName();
                trackNames.add(name);
            }
        }
    }
    return trackNames;
}
```