# Streams & Lambda Expressions

**SELF-STUDY TOPICS:** streams, lambda expressions, method reference, functional interfaces, Optional.

1. Create a string that consists of the first letter of each word in a list of Strings provided. (HINT: Use a StringBuilder to construct the result.)

2. Remove the words that have odd lengths from the a list of Strings. (HINT: Use one of the new methods from JDK 8.)

3. Replace every word in a list of Strings with its upper case equivalent. (HINT: Again, use one of the new methods from JDK 8.)

4. A map with pairs of type (String,String) is given. Convert every key-value pair of the map into a string and append them all into a single string, in iteration order. (HINT: Again, use a StringBuilder to construct the result String. Use one of the new JDK 8 methods for Map.)

5. A list of strings is given. Create a new list with all the strings from original list converted to lower case and print them out.

6. Modify exercise 5 so that the new list only contains strings that have an odd length.

7. Join the second, third and forth strings of the list into a single string, where each word is separated by a hyphen (-). Print the resulting string.

8. Produce an unbounded list of the powers of two. (An infinite stream.)

9. A list of integers is provided. Create a method which receives as parameter a number (integer), and if the number is palindrome it returns the number, otherwise it returns null. The return type of the method must be Optional<Integer>. Use the method to determine how many palindromes are in the list.

10. A list of employees is provided. (An employee has: name (String), age (int) and salary (double)). Compute the aveage salary of all employees.

11. A list of students is provided. ( A student has: name (String), avergae grade (double), list of courses taken (List<String>)). Create a map with pairs of type (String, Integer), where the key is the name of a student, and the value is the number of courses taken.

12. A list of cars is provided. ( A car has: brand (String), year of fabrication(int), colour(String), and price(double).) Sort the cars in descending order based on the price.

13. Determine the sum of all fibonacci numbers up to a given limit. (HINT: use Stream.iterate

**NOTE:** The key part of this homework is to use the new features of JDK 8 in the form of Lambda expressions and changes to existing APIs. Although parts of this you could easily do with loops that is not the goal. You should be able to complete all the exercises without resorting to a for or while loop. When you develop your Lambda expressions have a look at them and see if they can be replaced with an equivalent method reference.