Software Engineer Qualification Test

Introduction

Welcome to the Software Engineer Qualification Test. This is a programming skills test. The test will be evaluated by our specialists and based on the result you will or will not qualify for the enrollment in the interviewing process.

Please note: fulfilling this test successfully gives you the opportunity to apply for the position of Software Engineer at Commerzbank but under no circumstances guarantees a job offer.

Please follow these simple instructions:

1. Study the provided materials:
   - description of The Pyramid Problem
   - programming tasks
   - description of the provided source code
2. Understand the provided source code
3. Plan your approach and work on the solution
4. Send your solution and comments in electronic form to the Recruiting Consultant

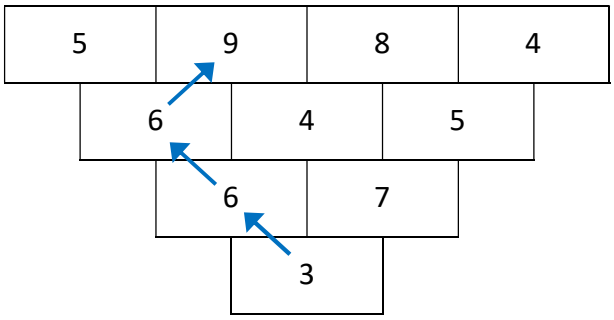
Solving all tasks typically takes ~90 minutes. Split your time among the tasks wisely: it is preferred that you produce at least a partial solution should you run out of time.

You are expected to use a computer with your preferred development environment to solve the tasks.

When responding through an email, ensure attached are exclusively java source files and a gradle build-file. Whatever else – especially compiled class files – may make your email blocked by security filters.
Alternatively, you can provide us with just a link to a public BitBucket/GitHub repository, Google Disk or any similar form of external storage with your solution.

The Pyramid Problem

Find the maximum sum path from the bottom to the top in an inverted pyramid as follows.
Example input:



Start at the bottom of the pyramid (value 3 in the example). In each step, move to one of the two *adjacent* fields above. Sum along all values thus visited until you get to the top row.

The maximum sum that you can achieve in this example is 3 + 6 + 6 + 9 = 24.

It is sufficient to determine the maximum sum; there is no requirement to list all the waypoints.

Programming Tasks

Task 1: Revive the project

The source code comes as a gradle-based project. Import the project into your development environment and fix possible problems to make the project buildable.
In case you decide not to fulfill this task, you can still go on by importing the sources into a new project in your IDE as you are used to. The source code is organized in a standard maven/gradle layout, i.e. `src/main/java` is a root of the "production" part and `src/test/java` holds tests. There are no resource files.

Task 2: NaivePyramidSolver

Find defects and apply fixes in *NaivePyramidSolver*. Comment your fixes in the code – describe what was wrong and what your change was about. (If you think that the algorithm is fundamentally wrong, do not rewrite it yet. You will get your chance in next task. Just describe what the problems are at this point.)
Prove the *NaivePyramidSolver* finds a maximum path in a pyramid by providing its unit tests. You can take an inspiration for the tests in *YourSolverTest* class that appears in the source code.

Task 3: YourSolver

Implement *YourSolver* class to solve The Pyramid Problem in a better way. Make code changes directly into the class and provide comments. Make sure that the updated *YourSolver* passes unit tests encoded in the *YourSolverTest*. Optionally consider increasing a size of "large" data in the code.

Provided Source Code

Together with this test there is also source code provided with the following contents:

- *Pyramid* - simple data structure representing the pyramid
- *PyramidGenerator* - API for generating the pyramid data structures
- *RandomPyramidGenerator* - Simple implementation of the generator API
- *PyramidSolver* - API of the problem solver
- *NaivePyramidSolver* – naïve Implementation of a solver. It contains some defects (see Task 2)
  For the use of *NaivePyramidSolver* see the *OurProgram.main()*
- *YourSolver* (see Task 3)
  For the usage of *YourSolver* (Task 3) see *YourProgram.main()*
- *YourSolverTest* (see Tasks 2 and 3)