Practice 2

Deadline: 2 weeks from now. Should be checked onsite (during labs).

Task 1

Given 1000 students, whose student IDs are **consecutive** and start with 12010001, choose an implementation to store their ages so that we can access their ages quickly using their student IDs.

Implementation 1:

Probably the most intuitive implementation is to use `HashMap<Integer, Integer>`, where key is student ID and value is age.

Implementation 2:

Another choice is to use `ArrayList` to store ages; when accessing the age, use `index = student ID - 12010001`.

Implementation 3:

The third choice is to use `LinkedList` to store ages; when accessing the age, use `index = student ID - 12010001`.

Implementation 4:

We can also use `int[]` to store ages; when accessing the age, use `index = student ID - 12010001`.

Installing JMH

We'll use the `JMH` (Java Microbenchmark Harness) library to profile different implementations and see which implementation is faster.

Approach 1

Download the jar files for JMH Core and JMH Generators: Annotation Processors. Click `File->Project Structure->Modules->Dependencies`, then click `+` to add these two .jars to your project.

Approach 2

Create a new Maven project in IntelliJ (File->New->Project->Maven). If you already have an existing project that's not Maven project, right-click the project and "Add Framework Support", then click Maven. Open its `pom.xml` file and add the following dependencies.

```
    <dependencies>
        <dependency>
            <groupId>org.openjdk.jmh</groupId>
            <artifactId>jmh-generator-annprocess</artifactId>
            <version>1.35</version>
        </dependency>
        <dependency>
            <groupId>org.openjdk.jmh</groupId>
            <artifactId>jmh-core</artifactId>
            <version>1.35</version>
        </dependency>
    </dependencies>
```

Right-click `pom.xml`, click "Maven->Reload project", and maven will automatically download these jars for you if you haven't done so.

Using either approach, we can use JMH in our project. In `src/main/java`, create a package `practice.lab2` and put `Practice2.java` in it. We've already implemented `HashMap<Integer, Integer>` (Implementation 1). Specifically, initializing the HashMap is done in `MyState.setUp` and getting the age by student ID is done in `testintmap()`.

In this practice, please implement the problem also using `ArrayList` (Implementation 2), `LinkedList` (Implementation 3) and `int[]` (Implementation 4) by filling in the `TODO` in `Practice2.java`. Specifically, you should also perform initialization in `MyState.setUp` and element accessing in `testarraylist`, `testlinkedlist` and `testarray`.

Finally, executing the `main` method.

Sample output

If you have done everything correctly, you should see that JMH is calculating the running time of `test*` methods (which may take a while) and finally output the following benchmark results (results may vary on different machines). You should submit this output to us.

```
Benchmark                          Mode  Cnt    Score      Error  Units
Practice2Answer.testarray          avgt    3    1.050 ±    0.069  ns/op
Practice2Answer.testarraylist      avgt    3    2.024 ±    2.246  ns/op
Practice2Answer.testintmap         avgt    3    4.275 ±   10.259  ns/op
Practice2Answer.testlinkedlist     avgt    3  157.808 ± 2118.662  ns/op
```

Task 2

Download `alice.txt` from Blackboard. Write a program to read `alice.txt`; compute and print the top 5 words that have the highest frequency. Use proper data structures and operations from the *Java Collections Framework*.

Sample output:

```
Word : Count
the  : 1000
and  : 900
to   : 800
a    : 700
you  : 600
......
```

Evaluation

The practice will be checked by teachers or SAs. What will be tested:

1. That you understand every line of your own code, not just copy from somewhere
2. That your program compiles correctly (javac)
3. Correctness of the program logic
4. That the result is obtained in a reasonable time

Late submissions after the deadline will incur a 20% penalty, meaning that you can only get 80% of this practice's score.