Lab #1: Student GradeBook

Introduction to Object-Oriented Programming in Java

Contents

1 Narrative Overview

In this lab, you will build a **Student GradeBook** system using Java and object-oriented programming principles. Each student will be represented as an *object* that stores identifying information (name) and academic data (grades).

Through this project, you will practice:

- Designing a class and instantiating objects
- Encapsulation using `private` attributes
- Writing constructors and overloaded constructors
- Using arrays to store and process data
- Performing basic statistical computations
- Testing object behavior using a driver program

This lab is intentionally incremental. You are encouraged to compile and test your code frequently as you progress.

—

2 Instructions

1. Create a dedicated workspace where you will store all assignments for this course.

2. Inside the `src` directory, create a new package named:

`Lab1`

3. Download the files `Main.java` and `StudentGradeBook.java` from Brightspace and place them into the `Lab1` package.

4. **Smoke Test:** Temporarily add a line in `main` that prints:

`"Hello world"`

Run the program to ensure there are no compilation or runtime issues. Once confirmed, remove or comment out the line.

5. Declare the following **private** attributes in `StudentGradeBook`:

- `firstName` (`String`)
- `lastName` (`String`)
- `grades` (`int[]`)

This demonstrates **encapsulation**, one of the four core principles of OOP.

6. Update the constructor so it accepts `firstName` and `lastName`. Inside the constructor:

- Assign the name attributes using the `this` keyword
- Allocate an integer array of size 5 for grades

By default, an integer array of size 5 is initialized as:

$$\{0, 0, 0, 0, 0\}$$

7. Implement the getter and setter for the student's full name.

Getter example:

```java
String name = student.getStudentFullName();
// returns "Marty McFly"
```

Setter example:

```java
student.setStudentFullName("Doc", "Brown");
```

8. Implement `reformatName(String name)`. This helper method:

- Takes a `String`
- Returns the same string with the first letter capitalized

Useful String Methods You may find the following `String` methods helpful when implementing `reformatName(String name)`.

- `substring(int start, int end)`
- `toUpperCase()`

substring(start, end) The `substring` method returns a *new String* consisting of the characters from index `start` (inclusive) to index `end` (exclusive).

Important: Java strings are *zero-indexed*.

```java
// Example string
String s = "marty";

// Indices:   m   a   r   t   y
//            0   1   2   3   4

s.substring(0, 1);    // "m"
s.substring(1, 3);    // "ar"
s.substring(2, 5);    // "rty"
s.substring(0, s.length()); // "marty"
```

toUpperCase() The `toUpperCase()` method is defined for the `String` class and returns a new String where all alphabetic characters are converted to uppercase.

```java
String s = "m";
String capital = s.toUpperCase();  // "M"
```

These methods can be chained together since both return `String` objects.

9. Update the setter to assign reformatted names, then update the constructor to call the setter. This demonstrates **code reuse** and the **single-responsibility principle**.

10. Implement `addGrade(int grade, int index)`.

- Valid grades lie in $[0, 100]$
- Valid indices lie in $[0, 4]$

Java will automatically throw an `ArrayIndexOutOfBoundsException` if an invalid index is used.

11. Implement `generateGradeList()`.

The purpose of `generateGradeList()` is to produce a **single String** representation of all grades stored in the array.

The returned String should consist of the integer grades *delimited by commas*, with no trailing comma.

Example:

If the internal grades array is:

$$\{95, 100, 80, 90, 85\}$$

then `generateGradeList()` should return:

```
"95,100,80,90,85"
```

This method should *return* the String, not print it directly.

12. Implement the statistical methods.

Note: If you are unfamiliar with floating-point numbers in Java, you may consult Appendix A for a detailed explanation of `float` vs `double` before continuing.

1. `getMean()` returns a `float`.
   Example usage:
   ```java
   float avg = student.getMean();
   ```
   Minimal float example:
   ```java
   float a = 1f;
   float b = 2f;
   float c = a / b;  // 0.5
   ```
2. `getMedian()` To compute the median:
   - Copy the grades array using `Arrays.copyOf`
   - Sort the copy using `Arrays.sort`
   - Return the middle element

   Why a copy is required: Arrays in Java are passed by **reference**. Sorting an alias of the original array permanently changes it. We want to preserve the original grade order.
3. `getMaxGrade()` and `getMinGrade()` You must implement a method to return the max (or min) grade from the grades array. This requires you to iteratively compute the max (or min) integer of the array. Below are methods under the Math class that you may use in your implementation:

```java
Math.max(a, b)
Math.min(a, b)
```

13. Implement `getLetterGrade(float grade)`:

Grading Scheme Use the following grading scale when implementing `getLetterGrade(float grade)`:

Numerical Range	Letter Grade
93 – 100	A
90 – <93	A-
87 – <90	B+
83 – <87	B
80 – <83	B-
77 – <80	C+
73 – <77	C
70 – <73	C-
60 – <70	D
<60	F

Define the contents of this method using conditional statements (`if`, `else if`, `else`).

Important: Any numerical grade outside the interval $[0, 100]$ should return the String:

```java
"Invalid_grade"
```

14. Write a second constructor that accepts `firstName`, `lastName`, and an integer array of grades. This is an example of **constructor overloading**.

15. Implement `generateSummary()` :

The `generateSummary()` method is responsible for printing a formatted summary of a student's academic performance based on the data stored in the object.

This method does *not* return a value. Instead, it prints output directly to the console.

Example Usage Consider the following object instantiation inside `main`:

```java
StudentGradeBook student4 = new StudentGradeBook(
    "michael", "keaton",
    new int[] {95, 100, 80, 90, 85}
);

student4.generateSummary();
```

Expected Output Your implementation of `generateSummary()` should produce output in the following format:

```
=================
First name: Michael
Last name: Keaton
Grades: 95, 100, 80, 90, 85
Mean: 90.0
Median: 90.0
Max grade: 100
Min grade: 80
Letter grade: A-
=================
```

Important:

- The output format (labels, spacing, and order) must match exactly.
- Capitalization of names should reflect the behavior of `reformatName`.
- Numerical values should be computed using the appropriate helper methods.

3 Submission Instructions

Submit a **ZIP file** to Brightspace containing:

```
Lab1/
  Main.java
  StudentGradeBook.java
```

Do **not** submit:

- `.class` files
- `.jar` files
- IDE metadata

A Appendix: Floating-Point Numbers in Java

Floats are 32-bit floating-point numbers (as opposed to `double`, which are 64-bit). Floats can reliably store approximately 7 digits of precision (about 16 for doubles).

```java
double pi = 3.141592653589792;
float pi = 3.141592f;
```

Important: A float literal must end with `f`.

Scientific Notation

```java
double x_standard = 2000;
double x_scientific = 2E+3;

float y_standard = 2000f;
float y_scientific = 2E+3f;
```

Negative exponents:

```java
double x_standard = 0.002;
double x_scientific = 2E-3;

float y_standard = 0.002f;
float y_scientific = 2E-3f;
```

You will learn more about floating-point representation under the IEEE standard in:

- CSCI 240: Computer Organization & Assembly language
- CSCI 361: Numerical Methods

—

—

B Appendix: Benchmarks (Self-Assessment)

If your implementation is correct, the following instantiations should produce the corresponding output.

Test Code

```java
StudentGradeBook student1 = new StudentGradeBook("marty", "mcFly");
student1.addGrade(85, 0);
student1.addGrade(90, 1);
student1.addGrade(75, 2);
student1.addGrade(95, 3);
student1.addGrade(100, 4);
student1.generateSummary();
```

Expected Output

```
=================
First name: Marty
Last name: McFly
Grades: 85, 90, 75, 95, 100
Mean: 89.0
Median: 90.0
Max grade: 100
Min grade: 75
Letter grade: B+
=================
```

—