Introduction to Programming
Lab Exercise for Week 19 (Trimester 2, week 6)

Updating the Student and Application3 classes

A slightly revised version of the Student class given in last week's lecture can be downloaded from the T2-Week06 subpage on Moodle – the code is also given here:

```
public class Student {
	private final String name; // Student's name, a constant.

	private double test1, test2, test3; // Grades on three
	// tests.

	public Student(String n) {  // Constructor
		// Check that n is not an empty string
		if (n == null || n.equals("")) // n is empty String
			throw new IllegalArgumentException();
		name = n;
	}

	// Getters (accessors – do not change object state)

	public String getName()   {
		return name;
	}
	public double getTest1() {
		return test1;
	}
	public double getTest2() {
		return test2;
	}
	public double getTest3() {
		return test3;
	}
	public double getAverage() { // compute average
		// test grade
		return (test1 + test2 + test3) / 3;
	}
	public String toString() {
		return "Name: " + name + "; Test scores: "
				+ test1 + ", " + test2 + ", " + test3
				+ "; " + "Average grade = " + getAverage();
	}

	// Setters (mutators – these do change object state)

	// private helper method for setters
	private void checkRange(double grade) {
		if (grade < 0.0 || grade > 100.0)
			throw new IllegalArgumentException();
	}

	public void setTest1(double grade) {
		checkRange(grade); test1=grade;
	}
	public void setTest2(double grade) {
		checkRange(grade); test2=grade;
	}
```

```
	public void setTest3(double grade) {
		checkRange(grade); test3=grade;
	}

}
```

One potential issue with this design is that it fixes the number of tests at 3, as there is a separate named getter and setter for each test score and if, for example, test 3 were removed any code that called setTest3() would no longer compile. One way to avoid this is to have the class be modified so that it declares a static method, numberOfTests(), to indicate the number of tests, and that only a single getter and a single setter be provided, which take the test number as a parameter:

```
public class Student {

	private static final int NUMBER_OF_TESTS = 3;

	public static int numberOfTests() {
		return NUMBER_OF_TESTS;
	}


	// Other declarations...

	public double getTest(int testNum) {...}
	public void setTest(double grade, int testNum) {...}

}
```

Why are the private constant and the method that returns it declared as static?

With this arrangement it is possible to write code that uses the Student class in such a way that changing the number of tests would not require client code to be amended and recompiled – particularly if anyone writing such code was aware that the number of tests might be varied in future.

1. Modify the Student class to adopt this approach. What changes to the Student class fields (that is, its instance variables) are suggested as a result of these changes to the public interface of the class?

2. Download the Application3.java file from the T2-Week06 subpage and modify it to use the amended Student class that you have produced, rather than the StudentRecord class discussed in the previous lecture, and run the program, entering data for some students at the keyboard.