CS 149: Programming Fundamentals — **James Madison University**
Coding Exam #3 — **Fall 2015**

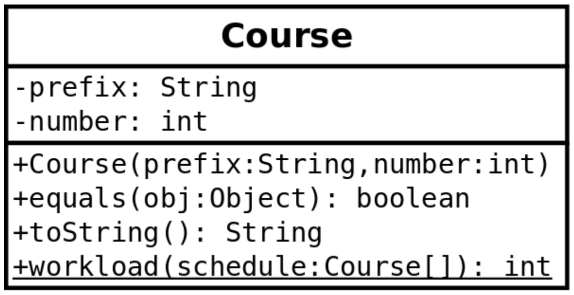
This work complies with the JMU Honor Code. I have neither given nor received unauthorized assistance, and I will not discuss the exam contents with anyone who has not taken it for credit.

Name:______________________________ Signature:______________________________

Schedule Planning

After completing a semester of college, you realize it would be helpful to plan ahead and estimate the workload of future years. However now that you've taken CS 149, doing simple calculations by hand (or even using Excel) is not fun anymore. You can't help but think about object-oriented programming solutions for everyday problems!

For this exam, you need to create two classes from scratch: **`Course`** and **`CourseTest`**. The first class must match the UML diagram below exactly. The second class must use JUnit to test your code. At the end of (and during) the exam, you will submit these two files via Web-CAT. You may submit as many times as you like; only the last submission will be graded.



```
Course
-prefix: String
-number: int
+Course(prefix:String,number:int)
+equals(obj:Object): boolean
+toString(): String
+workload(schedule:Course[]): int
```

- One example `Course` is CS 149 (with `prefix` "CS" and `number` 149).
- The **constructor** automatically converts the prefix to ALL CAPS. If the number is negative or greater than 999, it should be changed to zero.
- The **equals** method returns `true` if the courses have the same prefix and number, unless the number is zero, in which case they are not considered equal.
- The **toString** method returns the prefix and number separated by a space and formatted to three digits padded with zeros (e.g., "CS 005").
- The **workload** for a set of courses is measured in *points*. Lower division courses (with numbers below 300) get two points, upper division courses (in the 300-499 range) get three points, and graduate courses (500 and above) get four points. CS and MATH courses get an extra point. If the schedule has zero courses or is `null`, then workload should return -1.

You must write complete documentation comments for both classes (including your name and today's date) and all methods (including @param and @return tags). It's strongly recommended you complete the exam in this order:

1. Write documentation and stubs for all classes and methods. [10 min]
2. Implement your test cases; they should all fail at this point. [10 min]
3. Submit to Web-CAT to make sure that it compiles and runs. [10 min]
4. Write and test the first three methods; submit to Web-CAT. [15 min]
5. Write and test the `workload` method; submit to Web-CAT. [15 min]

You may use the back of this sheet as scratch paper. Good luck!