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

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 a class from scratch: **Course.** This class must match the UML diagram below exactly. At the end of (and during) the exam, you will file via Autolab. You may submit as many times as you like; only the last submission will be graded.

- 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 your class (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. Construct some test cases(not code but logical ones). [10 min]
    3. Submit to https://autolab.cs.jmu.edu to make sure that it compiles and runs. [10 min]
    4. Write and test the first three methods(constructor, equals, and toString); submit to Autolab. [15 min]
    5. Write and test the workload method; submit to Autolab. [15 min]

```
┌─────────────────────────────────────────────┐
│                   Course                     │
├─────────────────────────────────────────────┤
│ -prefix: String                              │
│ -number: int                                 │
├─────────────────────────────────────────────┤
│ +Course(prefix:String,number:int)           │
│ +equals(obj:Object): boolean                 │
│ +toString(): String                          │
│ +workload(schedule:Course[]): int            │
└─────────────────────────────────────────────┘
```

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