# COSC-211: DATA STRUCTURES
# HW1: PRACTICE WITH CLASSES

Due Tuesday, February 5, 11:59pm

**Reminder regarding intellectual responsibility:** This is an individual assignment, and the work you submit should be your own. Do not look at anyone else's code, and do not show anyone your code (except for me and the course TAs). If you talked with any of your classmates about this assignment, please list their names in a comment at the top of your submission.

## 1 Introduction

The department has two servers named `remus` and `romulus` that are available for student use. They are identical; it doesn't matter which you use. If you've taken COSC-111 or COSC-112 at Amherst, you're probably already set up to use them. If you're a five college student, you will need to visit IT to get an account set up. You are welcome to write code on your own computer, but I will test your code on `remus/romulus` and you are responsible for making sure your code works properly on those servers.

## 2 Setup

The purpose of this assignment is to give you some practice with creating classes and writing methods that use classes. If you are unsure of how this works, now is the time to ask questions!

On `remus/romulus`, begin by creating a directory for this course, `cd`-ing into that directory, creating a directory for this assignment, and `cd`-ing into that directory. Then copy two Java files:

```
$ wget -nv -i https://tinyurl.com/cosc211hw1
```

You should now see in your directory the two files `Day.java` and `UseDay.java`. Compile both files, the run the program using the command:

```
$ java UseDay February 1 2019
```

You may enter any day you like; the program requires input in the form `<month> <day> <year>`. When you run the program it should print out whether the year you entered is a leap year.

Open up `UseDay.java`. You'll see that right now the program doesn't do very much. It creates a new `Day` object (an instance of the `Day` class), calls a method called `setYear()`, and prints out whether the year is a leap year.

Now open up `Day.java`. The `Day` class currently contains one field, `int year`, and a few methods. Note that the `year` field is declared as `private`. Make sure you understand what all of the methods do and how to call them from `UseDay`.

## 3 Your Tasks

Your job in this assignment is to flesh out the `Day` class so that it does something more interesting and meaningful than just store a year.

First, make some modifications to `Day.java` to give the class some basic functionality.

1. Add fields to store a month and a day (the day of the month—i.e., 26—not the day of the week). You may choose whether you want to represent each piece of information as an `int`, as a `String`, or in some other way.

2. Add `get` and `set` methods for your month and day fields. These should follow the format of the `getYear()` and `setYear()` methods that I have provided. Your `getMonth()` method should return a `String` containing the name of the month, so if you have chosen in part (1) to represent months using some other type, you will need to write some code to translate your alternative representation into a `String`.

3. Add a constructor that takes as input a `String month`, `int day`, and `int year` **in that order and with those types** and sets the fields accordingly.

Next, add some methods to the `Day` class. After writing each method, add code to `UseDay.java` to *test* your method. Make sure you test multiple cases, especially if there are corner cases that might behave unusually.

5. Add a method called `nextMonth()` that returns, as a `String`, the month that follows this `Day`'s month.

6. Add a method called `daysInMonth()` that returns the number of days in this `Day`'s month. The following poem might help if you're prone to forgetting how many days there are in each month:

   *Thirty days has September*
   *April, June and November.*
   *All the rest have 31*
   *Except February, it's a different one*
   *It has 28 days clear, and 29 each leap year*
   –Richard Grafton, *Abridgment of the Chronicles of England*, 1562

   You may find it helpful to call the `isLeapYear()` method that I have provided in the `Day` class.

7. Add a method called `nextDay()` that creates and returns a new `Day` object storing the month, day, and year of the day immediately following this one.

Finally, do something interesting.

8. Add some more methods (at least 2) to `Day.java` that manipulate dates in various ways. These methods can do whatever you want. Try to write methods that use different numbers and types of arguments, different return types, etc. Include **comments** in your `Day.java` code that explain what your new methods do, and include code in `UseDay.java` to **test** your new methods.

## 4 Submit your work

If you talked with any of your classmates about this assignment, please list their names in a comment at the top of your submission.

Submit your modified `Day.java` and `UseDay.java` using either the submission web site: www.cs.amherst.edu/submit or the `cssubmit` command:

```
cssubmit *.java
```

and use the numeric menu to select the correct course and assignment.

**This assignment is due on Tuesday, February 5, 11:59pm.**