*UNBC* **Laboratory Assignment**
Computer Science 100—Fall 2012

# Classes and Objects

## Purpose:

To be able to create a basic class and manipulate objects of that class.

## Due Date:

The completed lab assignment is due Friday, November 23, 2012 *at the beginning of lecture*.

## A Time Class

Write a simple `Time` class that has the following methods:

- (to set components of the `Time`)

  ```
  public void setHour   (int hour)   {/* ... */}
  public void setMinute (int minute) {/* ... */}
  public void setSecond (int second) {/* ... */}
  ```

  These methods should ensure that the resulting `Time` object is legitimate.

- (to get attributes)

  ```
  public int getHour   () {/* ... */}
  public int getMinute () {/* ... */}
  public int getSecond () {/* ... */}
  ```

  methods that return the corresponding value from the object.

- a method

  ```
    public String toString() {/* ... */}
  ```

  that produces a string like "`22:03:12`" from a time.

- a method

  ```
    public void advanceBy(int seconds) {/* ... */}
  ```

  that changes the time by a given number of seconds.

⇒ Write a `TestTime` class that uses the various methods of the `Time` class to show that they work.

⇒ Write a `TimeQuestion` class that implements a question-asking program that asks the user for a time and for a number of seconds to add, then writes a message like

```
  Advancing   134 seconds from 12:03:15 gives 12:05:29.
```

### Hand in for the `Time` program

A clean scriptfile containing

- listings for the `Time` class, the `TestTime` class, and the `TimeQuestion` class; and
- test runs `TestTime` and `TimeQuestion`.

## A Conversion Problem

Frequently, small programs to convert data from one format to another are incredibly useful. Although the details are not accurate, the following problem is similar to one encountered by BC Forest fire-fighters who collect GPS data in one format, and need to convert it to a different format for use with mapping software.

### The input data format

Input data are longitudes and latitudes (similar to those Lab 3) stored in a text file. Each line of the file consists of one longitude, followed by spaces, followed by one latitude. Both the longitudes and the latitudes are in decimal degrees, with positve numbers denoting East longitudes and North latitudes, and negative numbers denoting West longitudes or South latitudes. An example file is shown in Figure 2.

sample-in.txt

```
 135.0990  65.0915
  24.6912 -39.6077
 -44.7398  68.4376
  57.0247  -7.5715
  72.2593  74.3285
 -66.7411  68.2415
```

Figure 1: Sample input data

### The output data format

The output data are also longitudes and latitudes, but in a slightly odder format.

Longitudes are written in a format like `135E5.9'` or `66W44.5'`, that is, as an integral number of degrees, followed by a direction letter (`E` or `W`), followed by a number of minutes (there are sixty minutes to a degree) that is accurate to one decimal place.

Latitudes are written in a similar format, but with `N` or `S` for the direction letter. A longitude-latitude pair is written in parentheses.

The data in Figure **??** correspond to those in Figure 2.

sample-out.txt

```
(135E 5.9',65N 5.5')
( 24E41.5',39S36.5')
( 44W44.4',68N26.3')
( 57E 1.5', 7S34.3')
( 72E15.6',74N19.7')
( 66W44.5',68N14.5')
```

Figure 2: Sample output data

### Programming tasks

⇒ Write a [`static`] method (possibly with several sub-methods) that takes two arguments, an input file name, and an output file name, and that then reads longitude-latitude pairs from the input file and writes the re-formatted pairs to the output file.

⇒ Write a `main`-method that tests your program on the files `"sample1-in.txt` and `"sample2-in.txt` found on the blackboard site.

---

## Take two

⇒ Write a `LonLatPair` *class* that has the following methods:

- a `LonLatPair` constructor (with two double arguments),
- non-static `setLatitude` and `setLongitude` methods that take a `double` argument in degrees, with the convention that negative arguments represent South or West.
- non-static `getLatitude` and `getLongitude` methods that return a `double` number of degrees, with the convention that negative values represent South or West.
- a `boolean isNorth` method that returns `true` iff the latitude is $> 0$,
- a `boolean isEast` method that returns `true` iff the longitude is $> 0$,
- a method

```
    public static double minutesFromDegrees(double) { ... }
```

that computes the minutes part of the (absolute value of) a number of decimal degrees. For instance, `minutesFromDegrees(-11.51)` should give `30.6`.

- a method

```
    public static String format2(LonLatPair ll) { ... }
```

that produces a `String` like one line of the output file above.

⇒ Rewrite your program above to take advantage of the `LonLatPair` class.

## Hand in for the conversion problem

A clean scriptfile containing

- listings for all of your programs; and
- test runs of both programs for both test data files.