CS 106A Section 6 Handout (Week 7)

Classes and Objects

1. **Student.** *(Thanks to Mehran Sahami for this problem.)* Define a class named `Student`. A `Student` object represents a Stanford student that, for simplicity, just has a name, ID number, and number of units earned towards graduation. Each `Student` object should have the following public behavior:

`new Student(`***`name, id`***`)`	Constructor that initializes a new `Student` object storing the given name and ID number. The unit count should initially be 0.
`s``.getName();` ***`s`***`.getID();` ***`s`***`.getUnits();`	Returns the name, ID, or unit count of the student, respectively.
`s``.incrementUnits(`***`units`***`);`	Adds the given number of units to this student's unit count.
`s``.hasEnoughUnits();`	Returns whether the student has enough units (180) to graduate.
`s``.toString();`	Returns the student's string representation, e.g. `"Nick (#42342)"`.

2. **TimeSpan.** Define a class named `TimeSpan`. A `TimeSpan` object stores a span of time in hours and minutes (for example, the time span between 8:00am and 10:30am is 2 hours, 30 minutes). Each `TimeSpan` object should have the following public behavior:

`new TimeSpan(`***`hours, minutes`***`)`	Constructor that initializes a new `TimeSpan` object storing the given time span of minutes and seconds. Assume that the parameters are valid: the hours are non-negative, and the minutes are 0 - 59.
`ts``.add(`***`hours, minutes`***`);`	Adds the given amount of time to the span. For example, (2 hours, 15 min) + (1 hour, 45 min) = (4 hours). Assume that the parameters are valid: the hours are non-negative, and the minutes are between 0 and 59.
`ts``.add(`***`timespan`***`);`	Adds the given amount of time (stored as a time span) to the current time span.
`ts``.getHours();` ***`ts`***`.getMinutes();`	Returns the hours or minutes of the time span, respectively.
`ts``.getTotalHours();`	Returns the total time in the time span as the real number of hours, such as 9.75 for (9 hours, 45 min).
`ts``.toString();`	Returns a string representation of the time span of hours and minutes, such as `"28h46m"`.

The minutes should always be reported as being in the range of 0 to 59. That means that you may have to "carry" 60 minutes into a full hour. The following code creates a `TimeSpan` object and adds to it three times:

```
TimeSpan ts = new TimeSpan(1, 50);
ts.add(1, 55);
println(ts + " is " + ts.getTotalHours() + " hours");   // 3h45m is 3.75 hours

TimeSpan ts2 = new TimeSpan(2, 30);
ts.add(ts2);
println(ts + " is " + ts.getTotalHours() + " hours");   // 6h15m is 6.25 hours

ts.add(0, 51);
println(ts + " is " + ts.getTotalHours() + " hours");   // 7h6m is 7.1 hours
```

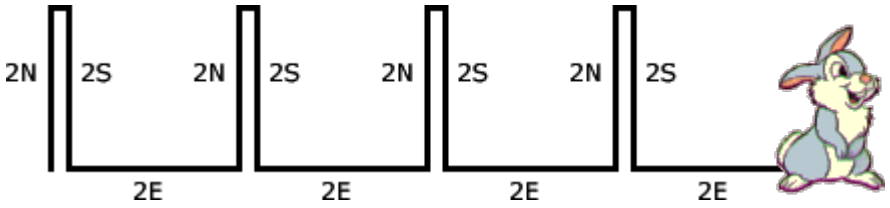
Notice that the second time is not 5 hours, 75 min, although that's what you'd get by just adding field values.

CS 106A Section 6 Handout (Week 7)

Critter Classes

3. **Rabbit.** Define a Critter class named `Rabbit` with the following behavior:

constructor	`public Rabbit()`
color	dark gray (`Color.DARK_GRAY`)
eating behavior	alternates between `true` and `false` (`true`, `false`, `true`...)
fighting behavior	if opponent is a `Bird` or `Vulture`, then scratch; otherwise, roar
movement behavior	hops in an L-shaped pattern: first north twice, then south twice, then east twice, and repeat
toString	`"R"`



4. **WhiteRabbit.** Define a Critter class named `WhiteRabbit` with the following behavior:

constructor	`public WhiteRabbit()`
color	white
eating behavior	alternates between `true` and `false` (`true`, `false`, `true`, ...)
fighting behavior	if opponent is a `Bird` or `Vulture`, then scratch; otherwise, roar
movement behavior	hops in an L-shaped pattern, in cycles of 5: first north 5 times, then south 5 times, then east 5 times, and repeat
toString	`"R"`

Hint: This Critter is similar, but not identical, to the `Rabbit` we defined above. How can we use the class we've already defined to avoid re-writing the common behavior?

5. **Frog.** Define a Critter class named `Frog` with the following behavior:

constructor	`public Frog(int age)` The age passed will be between 1 and 9 inclusive.
color	green
eating behavior	never eats (this is the default eating behavior)
fighting behavior	always forfeits in a fight (this is the default fighting behavior)
movement behavior	moves east sometimes, otherwise stays put (center), based on the frog's age: If the frog is 1 year old, moves to the east every move. E, E, E, E, E, E, E, E, E, E... If the frog is 2 years old, moves to the east once every 2 moves. C, E, C, E, C, E, C, E, C, E... And so on...
toString	`"F"`

This document and its contents are copyright © Stanford University. All rights reserved.