CS 141: Introduction to Computer Science
Winter Term, 2013

# Lab 1

## Cars

In this lab, we will use BlueJ to practice writing classes.

Begin by starting BlueJ. Select `New Project` under the Project menu and create a project named "lab1". This will bring up a new project window; close the old one.

Create a class called `Car`. This will give you the shell of a class definition.

Within the braces, add declarations for three integer attributes. Call them `originalValue`, `speed` and `damage`. These will store the value of the car if it is undamaged, its current speed, and the number of dollars worth of damage it has sustained. Recall that the form for an attribute declaration is

```
attributeType attributeName;
```

Next we need a constructor. Recall that these have the form

```
className(argumentList) {
  list of statements;
}
```

Write a constructor that takes one integer argument called `value`. Your constructor should set the `originalValue` attribute to the constructor's argument and set the other two attributes to 0.

At this point, compile your program and see that the constructor works by creating a `Car` object and double-clicking on it to view the values of its attributes.

Next, write the signature of a method `getSpeed` which takes no arguments and returns the current speed of the car.

Then, use this signature to write the method. Recall that methods have the form

```
returnType methodName(argumentList) {
  list of statements;
}
```

and that you cause a method to return a value with a statement of the form `return value;`

Again, once you've written the method, try it out. (In this case, create a `Car` object and call its `getSpeed` method.)

The `Car` class is pretty boring since its attributes never change. To fix this, write a method `accelerate`, which takes an integer called `value` and doesn't return anything. The body of this method should increases `speed` by `value`. (You do this by setting `speed = speed + value;`)

Once you have written this method, compile your program again. This time create a car and ask it to accelerate a couple of times. Check that its state changes appropriately.

To complete the `Car` class, write the following other methods:

- `crash`, which increases damage by 20 times the speed (the symbol for multiplication is an asterisk *). The method should also change the speed to zero. It doesn't return anything.

- `currentValue`, which takes no arguments and returns the current value of the car, i.e. its original value minus the amount of damage it currently has.

Use the Object Inspector to make sure your methods work as advertised. Test with the following actions:

1. Create a `Car` with original value 10000.
2. Accelerate by 20 mph.
3. Crash.
4. Check the car's current value. (It should be 9600.)
5. Accelerate by 30 mph.
6. Crash again.
7. Check the car's current value. (It should be 9000.)

## Planes

If you have more time, see if you can write a `Plane` class. Again, start by creating a new class. You want your class to support the following operations:

- A constructor that takes no arguments and creates a plane that is empty. The gas tank (with capacity 1000 gallons) is full, however.
- `changePassengers`, called when the plane lands and takes on new passengers. All previous passengers get off and the method takes an argument giving the number of passengers who replace them.
- `travel` makes the `Plane` fly to a new destination. It takes an `int` giving the distance traveled in miles. The plane uses a gallon of fuel for each mile travled.
- `costToFuel` takes the current price per gallon of fuel and returns the amount it would cost to fill the plane's fuel tank. (Both amounts are in cents— the computer doesn't know this (or care), but you need to so you can interpret the program's output.)
- `buyFuel`, which is called when the owners decide to fill the plane's fuel tank. It takes an integer argument giving the current price of a gallon of fuel in cents.
- `profitSoFar` takes no arguments and returns the amount of profit the `Plane` has earned since its creation. This profit is the amount passengers pay for flying minus the cost of purchased fuel. The plane earns 10 cents per mile traveled for each passenger carried.

Think about what attributes a `Plane` object must have to support all these operations. (Feel free to ask once you've thought about it.) Then write the class and test it with the following actions:

1. Create a plane.
2. Load 30 passengers.
3. Travel 600 miles.
4. Check the total profit, which should be $1800.00.
5. Let the passengers off and load 23 more.
6. Travel 250 miles.
7. Refuel at $1.37 per gallon.
8. Check the total profit, which should be $1210.50.