CIS351-Classes and Objects Lab

Objectives

Students will be able to:

- Write a class from which objects will be created.
- Use this class to make several objects.

Submission Instructions

1. Submit completed Car.java and CarTester.java in Blackboard.

Key Terms

- Class
- Object
- Constructor
- Accessor method
- Mutator method
- Fields (instance variables)

Download Materials

There is nothing to download today. You will write this program from scratch.

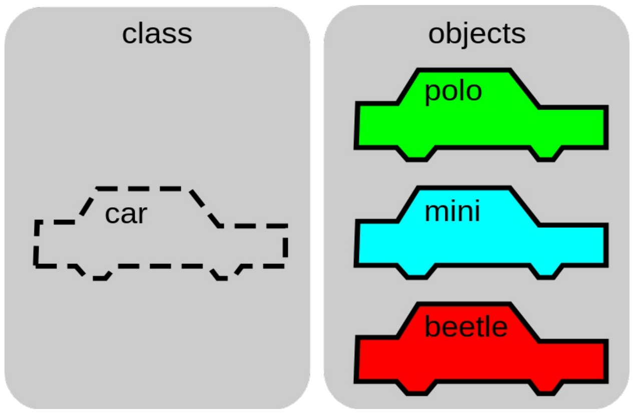
Background

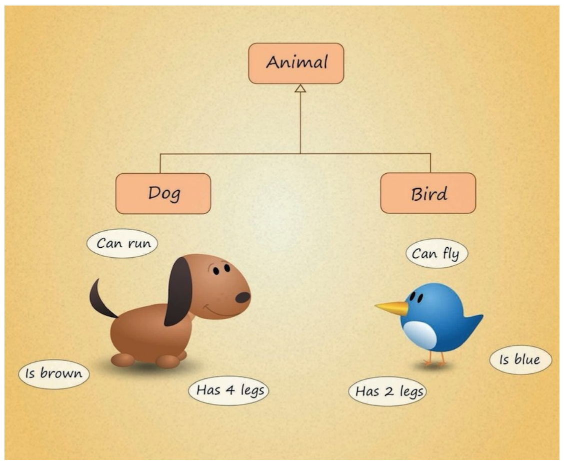
Object Oriented programming switches the view of a program as a series of steps to a view of a program as an interaction between objects. You design the objects with what they should look like (attributes or fields) and what behaviors they should have (methods), then your program will control how those objects interact in the world.

What is class and objects in Java: A **class** is a user defined blueprint or prototype from which *objects* are created. An **object** is a member or an "instance" of a **class**. An **object** has a state in which all of its properties have values that you either explicitly define or that are defined by default settings.

Look at the following two pictures carefully. Do you now see the subtle difference between class and objects?

- Car is a class, but polo, beetle and mini are objects created out of that class.
- Animal is a class, but Dog and Bird are objects created out of that class. What other animal you can create out of the Animal class? think !





What is UML: The Unified Modeling Language (UML) is a general-purpose, developmental, modeling language in the field of software engineering, that is intended to provide a standard way to visualize the design of a system/program/software.

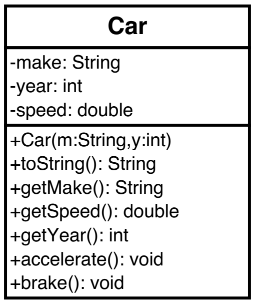
IMPORTANT: ***Before you start, you NEED TO KNOW, how to decode an UML Diagram. Please read the following tutorial before you move on with the rest of the lab: TUTORIAL TO READ***

Part 1: General Instructions:

1. Setup your program environment. Create a new folder for your files.
2. Create a new java file named Car.java.

Part 2: Create a basic Car class:

The goal for this lab is to implement a `Car` class so that it conforms to the UML below:



1. Create three instance variables within the class: an `int` that holds the car's model year, a `String` that holds the car's make, an `double` that holds the car's speed. These should be listed as `private`.

2. Create a constructor that accepts the model year and make and sets the speed to 0. Note that either you should make the constructor's parameters a different name than the instance variables or use the "`this`" qualifier when placing the parameter's value into the instance variable.

3. Code the `toString` method. This method should take no parameters and should return a string generated using the `String.format` method:

   ```
   String.format("A %d %s that is going %.1f mph", year, make, speed)
   ```

4. Compile your Car class and correct any errors.

Part 3: Test your Car.

1. Create a CarTester class (CarTester.java) which contains a main method.

2. Create two Car objects of your choice. (Hint: you need to call the constructor with the values you want for your "dream cars").

3. Display the Car objects so that you know that your constructor is working correctly by calling the toString method and printing that result. (Hint, you will need to call the method with the Car objects that you created)

4. Here is a sample of how to create/instantiate car objects and test them. The contents of `main` should look something like the following.

```
Car car1;
Car car2;

car1 = new Car("Ford", 1997);
car2 = new Car("Toyota", 2014);

System.out.println(car1.toString());
System.out.println(car2.toString());
```

Part 4: Refinement:

1. Edit Car.java.

2. Create **accessor** methods that let you access each individual element of the Car. They should be named with "get" followed by the name of the attribute. So one of your accessor methods would be `getSpeed`(). They should take no parameters and should return the value of that attribute.

3. Create two **mutator** methods, accelerate() and brake(). These methods are void methods and take no parameters. When called, the `accelerate`() method should increase the `speed` variable by 5 miles per hour. The `brake`() method should decrease the `speed` variable by 5 miles per hour.

Part 5: Testing the refinement:

1. Edit CarTester.java.

2. For the first car, call the accelerate() method 5 times in a row and call the getSpeed() method in a print statement (with an appropriate label) for each increase.

3. Do the same for the second car but accelerate only 3 times in a row also calling getSpeed() in a print statement for each acceleration.

4. For the first car, call brake 5 times in a row calling getSpeed() with each pass.

5. Print the result of a call to the toString() method for each of the cars. The first car should be at zero and the second at 15 mph.

Part 6: - (OPTIONAL) Just for fun:

1. It is possible to pass one object into the method for another object. For example, the String method equals is called by a String object and is passed another String object: `answer.equals("yes")` for example.

2. Add a mutator method to your Car class called `crash` that takes a Car object as a parameter and returns nothing. It should display the message "CRASH" and then set the speed of both the calling object and the called object to 0.

3. In your tester, call `accelerate` for the first car until it is the speed of the second. Call the first car's `crash`() method passing in the second car.

4. Display both of the cars after the `crash`() using the `toString`() method.

Grading Criteria

Total points: 10 points

Part 2 - 3pt
Part 3 - 2pt
Part 4 - 3pt
Part 5 - 2pt

Farzana Rahman / frahman@syr.edu