

University of Pennsylvania
ESE 112: Introduction to Electrical & Systems Engineering

Lab 7: Intro to OOP with the Boe-Bot Platform

Objective

- To introduce Object-Oriented Programming (OOP)
- To write a class that is modeled using OOP paradigm

Introduction

An object is a way to bundle data and manipulate that data. This bundle is known as encapsulation. This is different from the procedural programming covered in the first part of the course, where data and code are separated. In the OOP model, a program is a collection of objects where these objects pass messages to each other. The goal of this lab is write classes using the OOP paradigm.

Background

Objected Oriented Programming (OOP):

Please refer to the OOP notes in the lecture for syntax.

Light Emitting Diode(LED):

We have used LEDs in the course before, but lets look at an LED from the OOP perspective. When connecting an LED to a microcontroller, the behaviors or actions we want the LED to have are:

1. Be able to connect to an I/O pin
2. Be able to turn on
3. Be able to turn off
4. Be able to inform its current state (i.e. on or off)
5. Be able to invert its state and report what it's doing at the moment

Given the nature of OOP, we want to write a class that will contain the common features for any LED. Given the above actions for an LED, the common features or data that any LED object should keep track are:

1. I/O pin number
2. Current state of LED (i.e. ON or OFF state)

Note that in order to carry out the action of turning on and off an LED, we will still need to use the methods from CPU class. However, when we actually want to work with LED, we create a LED object, which will connect to a particular I/O and be in a particular state. Over time we can change the state of the LED or the pin to which it is connected. Note that doing something to one LED object will not affect other LEDs.

Pulse Width Modulation (PWM):

Recall that to drive a Boe-Bot wheel, a servo must receive a pulse roughly every 20ms, where the length of that pulse (1ms, 1.5ms, 2ms or roughly 130, 173, 220 units), determines whether or not the servo will turn clockwise or counterclockwise. In this lab, instead of using the pulseOut method we have been using, we will use the PWM class. The PWM is a virtual peripheral, which means that it runs in the background for the entirety of the program or until it is stopped. The PWM class allows us to control the wheels in a manner similar to the pulseOut class, but using objects.

To use the PWM class, you will have to import the class just like you imported the CPU class. Since both classes (CPU and PWM) are located within the core folder, you will just need to put the following at the top of your program (where you usually import):

```
import stamp.core.*;
```

To create a new PWM object, use the following syntax:

```
PWM instance_name = new PWM(CPU.pinX);
```

Of course, replace “instance_name” and “pinX” with what you need. This will be very convenient for us — now we need only create a PWM object for the left and right servos and we can easily control our robots:

```
PWM left = new PWM(CPU.pin13);  
PWM right = new PWM(CPU.pin12);
```

Now how do we control each servo? Object classes generally have methods implemented to control each instance of that object. For our PWM objects, the methods we care about are:

```
start();  
stop();  
update(int highTime, int lowTime);
```

For more information on these methods, go to the Java Documentation for the PWM class (located on the ESE112 website along with the CPU class documentation).

Remember that our lowTime in general will be 2304 (in 8.68us units), or about 20ms, and our highTime will be what we have always used to make it go forward, backward, or stop (130, 173, or 220). In order to move one servo we need to do:

```
PWM right = new PWM(CPU.pin12);  
right.update(220, 2304);  
right.start(); //starts to the servo to propel wheel in fwd motion
```

Remember that your servos must still be calibrated to not move when sending a pulse width of 173 units (1.5ms).

Materials

- Boe-Bot unit with Javelin Stamp

Pre-Lab Questions

1. What is the use of a constructor in a class written using the OOP model?
2. What does the “new” operator do?
3. Explain the difference between global, local, and instance variables.
4. Consider the following code:

```
public class Toy{
    int toyNum;

    public Toy(int num){
        int toyNum = num;
    }

    public int getToyNum(){
        return toyNum;
    }
} //end of Toy class
```

```
public class TestToy{

    public static void main(){

        Toy t = new Toy(5);
        int n = t.getToyNum();
        System.out.println("Toy num = " + n);
    }
}
```

When the print is statement executed in the main method from the TestToy class, we observe that value of n is 0, and not 5. Why does this happen? How will you fix it?

Lab Instructions

Part I: Writing LED class

In the LED class, complete the code as stated in comments. Test your code by completing the main method in the FlashingLEDs class as stated in the comments.

Part II: Writing BoeBotControl class using PWM

Recall that a Java program is collection of Objects. We can create objects that use other objects to accomplish various tasks. You will write the BoeBotControl class consisting of PWM objects. These are basically to control the wheels of the BoeBot. In the future, you can think of adding more PWM objects (e.g. if we add another servo to control an arm or a sensor for any purpose). Complete the methods in the class. Test the BoeBotControl class by using an existing program (e.g. your navigation path or light sensitive navigation). Note that now you will have to create a BoeBotControl object using something like:

```
BoeBotControl b = new BoeBotControl()
```

To perform actions, use:

```
b.moveForward()
```

Note: You will still need to use the CPU class in your programs (e.g. to use CPU.pin12). Also, you should use the delay method to time your movements, instead of the counter you have used in past labs. Once you have updated the servos to move forward, insert a delay for however long you want the Boe-Bot to move.

Post-Lab Questions

The answers for the following questions should be part of your Lab 6 (Flashlight Lab) report. This combined report for Labs 6 and 7 is due in your lab during the week of November 2.

1. Which method of controlling the Boe-Bot's servos, PWM or CPU.pulseOut, do you prefer and why?
2. What are some areas where Object-Oriented Programming would be useful?

As always, submit all your Java programs (one per team) to Blackboard Digital Drop Box in one zipped folder using the format on the ESE112 website under the Course Information section.