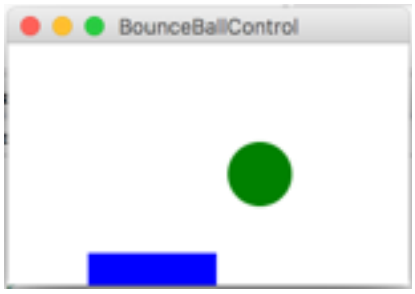
Submission deadline: April 7, 2017 23:59:59.
Question#1

Modify the bouncing ball example demonstrated in class to include a rectangle as shown in the figure here:



The width of the rectangle should be equal to radius*4, while the height should be equal to the radius. In your modified code, the ball should bounce back when it touches the surface of the rectangle. You should be able to move the rectangle left or right when pressing on it with mouse and dragging it towards the left or right.

Here is the the code shown in class for the bouncing ball:

www.cs.armstrong.edu/liang/intro11e/html/BallPane.html
www.cs.armstrong.edu/liang/intro11e/html/BounceBallControl.html

Question#2:
For the following class, write a JUnit test class to test the method getTotalRentPayment and whether an exception is thrown by the method setRentAmount.

```
public class RentCalculator {
      private double utilitiesCharge;
      private int numberOfMonths;
      private double rentAmount;
      private java.util.Date loanDate;

      /**
       * Construct a RentCalculator with specified monthly utilities
charge, number of
       * months, and rent amount
       */
```

```
	public RentCalculator(double utilitiesCharge, int numberOfMonths,
double rentAmount) {
		this.utilitiesCharge = utilitiesCharge;
		this.numberOfMonths = numberOfMonths;
		this.rentAmount = rentAmount;
		loanDate = new java.util.Date();
	}

	/** Return annualInterestRate */
	public double getutilitiesChargee() {
		return utilitiesCharge;
	}

	/** Set a new monthly untility charge */
	public void setutilitiesCharge(double utilitiesCharge) {
		this.utilitiesCharge = utilitiesCharge;
	}

	/** Return numberOfMonths */
	public int getNumberOfMonths() {
		return numberOfMonths;
	}

	/** Set a new numberOfMonths */
	public void setNumberOfMonths(int numberOfMonths) {
		this.numberOfMonths = numberOfMonths;
	}

	/** Return loanAmount */
	public double getRentAmount() {
		return rentAmount;
	}

	/** Set a newRentAmount */
	public void setRentAmount(double rentAmount) throws Exception {
		if (rentAmount < 0)
			throw new Exception("Money cannot be negative");
		this.rentAmount = rentAmount;
	}

	/** Find total payment */
	public double getTotalRentPayment() {
		double totalPayment = (rentAmount + utilitiesCharge) *
numberOfMonths;
		return totalPayment;
	}

	/** Return loan date */
	public java.util.Date getLoanDate() {
		return loanDate;
	}
}
```

Question#3:

a) Following the Observer design pattern, write a class called TenantObserver that prints a message to the tenant in its update method when setRentAmount is called in the RentCalculator class from Question#2. Modify the RentCalculator class to allow to register/ unregister observers.
b) Apply the Singleton pattern on RentCalculator to ensure that only one RentCalculator is could be instantiated. Change what is necessary in the RentCalculator class to apply the Singleton pattern.

Question#4: The List interface is defined as follows:

```
interface  List {
        int count();  //return the current number of elements in the list
        Object get(int index);  //return the object at the index in the list
        Object first();  //return the first object in the list
        Object last();   //return the last object in the list
        boolean include(Object obj);  //return true is the object in the list
        void append(Object obj); //append the object to the end of the list
        void prepend(Object obj);  //insert the object to the front of the list
        void delete(Object obj);  //remove the object from the list
        void deleteLast();  //remove the last element of the list
        void deleteFirst();  //remove the first element of the list
        void deleteAll();  //remove all elements of the list
}
```

(a) Write a class adapter to adapt Java ArrayList to the List interface.

(b) Write a main program to test your adapters through List interface.

(c) Same requirement as (a) and (b), but write an object adapter to adapt Java ArrayList to the List interface.

For all these questions, test your code with some inputs. You have to submit the following to the blackboard:

1-Source code.
2-Screenshots of the output or the test runs. Put all the screenshots in a single word document.