Goals
The goals of this lab are to learn how to override the toString() method and to be able to read in both numeric and String data, taking care to avoid the pitfalls that may occur with newline characters.

Problems

1. **Console Output**: In this problem we’ll model voting precincts. Each precinct has a name (i.e., "Worcester12") an address where voters can vote (i.e., "130 Winter Street") and a total number of voters in the precinct (i.e., 1673). Create a class Precinct, with three fields: name, address, and population.

   Create another class named PrecinctProgram, and define a main method for the PrecinctProgram.

2. Now let’s send some output to the console window. As you know, Java pre-defines an object called System.out that provides a mechanism to send output (as a String) to the window. In fact, with println you can output the values of almost any kind of object or primitive. If you want to output more than one thing, place an addition sign between the things you want to output (+ is the concatenation operator for Strings):

   ```java
   int day;
   day = 9;
   System.out.println("Today is Wed, Nov " + day + ".");
   ```

   We said that System.out.println can output the value of any object. In the main method of your PrecinctProgram class, create a Precinct:

   ```java
   Precinct worcester12;
   worcester12 = new Precinct("Worcester12",
   "130 Winter Street",
   1673);
   ```

   Use System.out.println to display the value of worcester12. What’s going on here?

3. Java has a class called Object from which all other classes are derived. The Object class provides method definitions for a number of methods which should be overridden whenever you create a new class. One of the methods that is provided by the Object class is a method called toString(). toString() returns a String representation of an object, and it is the value
returned by an object’s toString() method that is displayed when that object is supplied as an argument to System.out.println. Unfortunately, the Object class definition of toString() returns a string consisting of the name of the class of which the object is an instance, the at-sign character, and the unsigned hexadecimal representation of the hash code of the object (it’s not necessary for us to know what a hash code is at this point). So you usually want to override toString() so that it provides an easily-readable textual representation of an object’s value.

In your Precinct class, create a toString() method. It should have the heading

    public String toString()

It should return the String "Precinct: PREC-NAME, Address: PREC-ADDR, Population: PREC-POPULATION" where each of these capitalized words is replaced with the actual value in the corresponding Precinct field. Run your program that displays the value of worcester12 again.

4. **Keyboard Input:** Java includes a class called Scanner that makes it easy to read input from the keyboard. Import java.util.Scanner into your PrecinctProgram class. Java’s name for the keyboard is System.in, so create a new Scanner object like this:

    Scanner keyboard = new Scanner(System.in);

You can name your Scanner object anything you want, but programmers often name it keyboard when the input that will be read into the program comes from the keyboard. Now you can use the methods of the Scanner class with the object keyboard to input data that the user of the program types in.

5. Add a new method to your Precinct class:

    // Produces precinct with same name and address as this,
    // but with given amount added to the population
    public Precinct grow(int amount) {
        ...
    }

6. Use the Scanner method nextInt() to read an integer value in from the keyboard (int amount = keyboard.nextInt();). Use the value read in as the amount to grow worcester12’s population. Use System.out.println() to display the resulting Precinct. Run your program.

7. You may have been confused when you ran your program, because although the program was waiting for you to type in a number, it didn’t tell you that. Whenever you do keyboard input (interactive input), you should precede each input operation with a prompt: a message
that is displayed on the console that notifies the user what the program is expecting. Use System.out.println() to display a prompt, and run the program again.

By the way, System.out.print() works just like System.out.println(), except that with println, the next output goes on a new line, whereas with print, the next output is placed on the same line (in other words, println outputs a newline after it outputs the String in its argument list). Prompt messages are often written using print rather than println.

8. **Whitespace and Newlines**: You’ve probably noticed that you have to be very specific about the way you space your output when you use print or println. For example,

```java
int day = 9;
System.out.println("Here's today's date:" + "Nov" + day + "," + "2016");
```

produces very different output from

```java
int day = 9;
System.out.println("Here's today's date: " + "Nov " + day + ", " + "2016");
```

(Try it and see.) Whitespace is any sequence of characters such as spaces, tabs, and newlines, that appear as white space when written on paper. Whitespace also acts as a separator when you read in multiple input items from the keyboard. Each activation of the Scanner method nextInt() skips over any intervening whitespace. So, for example, if your program reads values into two int variables num1 and num2:

```java
int num1;
int num2;
num1 = keyboard.nextInt();
um2 = keyboard.nextInt();
```

the input may be entered at the keyboard in a number of ways:

- the two numbers may be entered on the same line, with any number of space and/or tab characters before and/or after each number
- the two numbers may be entered on two separate lines, one number on each line, with any number of space/tab characters before/after each number
- the two numbers may be entered on two separate lines with any number of blank lines before or in between

Scanner has a method called nextLine() that reads the remainder of a line of input starting wherever the previous reading left off, and returns the input read in as a String. nextLine() reads up to the next newline character (denoted in a Java program by \n...this is the
character that is read when the user types the ENTER key). When nextLine() reads a line of input, it reads the newline character (so that the next reading of input will begin on the next line) but the `\n` character itself isn’t returned as part of the String.

Modify PrecinctProgram so that it prompts the user to enter the name, address, and population of a precinct. Your program should read in the user’s input and use the values entered to create a new Precinct object. After creating the new Precinct, display it. (If the output isn't what you expected, read on.)

The fact that nextLine() reads the newline character but doesn’t store it as part of the returned String can be the cause of confusion when both numeric and text input are being read. What would happen if we read in the data for a new Precinct object this way (here we’re entering the population first, followed by the precinct name, and finally the address):

```java
int pop = keyboard.nextInt();
String preName = keyboard.nextLine();
String preAddr = keyboard.nextLine();
```

If the user types in the input on three separate lines like this:

```
1673
Worcester12
130 Winter Street
```

You might expect the value of pop to be 1673, the value of preName to be "Worcester12", and the value of preAddr to be "130 Winter Street". But what you’ll actually get is the value of pop is 1673, the value of preName is "" (the empty string), and the value of preAddr is "Worcester12" (put the code into your program and try it). Make sure you can explain why. Can you think of a way to modify the code so that the values are what we first expected? Modify your code to make it work.

Submission

When you’ve finished the lab you can export the project to a zip file and submit the zip file on InstructAssist. The name of the project in InstructAssist is Lab 3. To export a project, go to File | Export… and choose Archive File from the General folder. Check off the entire project in the top left window and make sure the format is .zip (not .tar). Finally, give the archive file a name and click Finish.