

## 1 Before Getting Started on the Exercises

Review the class examples we discussed this past week and make sure you understand what each command and operation does. Specifically, make sure you understand the construction of Objects.

For each of the following create a new project with an appropriate name and then write a program that solves the given problem. Remember to do the Shift+Ctrl+F to format the program and to put the standard comments at the top. For each program, you will be submitting all of the java code files through MyClasses. That is, the main program and the java files for each of the objects you created.

I also want either a Microsoft Word doc file (or LibreOffice Writer odt) or a text file (which you can create with NotePad++) of the output of at least one run of the program that tests all of the program features. This doc (or odt) or text file is to be uploaded to MyClasses as well. You can copy and paste output from the Eclipse console area to the word or text program. Each program must include header comments with at least your name, date, and short description of the program.

## 2 Exercises

1. This exercise is to create another class structure (Object). Create a new project and a new main class as usual. Now create another class called `Rectangle`.

In the `Rectangle` class:

- (a) Have two private data members `width` and `height`, both doubles.
- (b) Create a constructor that brings in as parameters doubles for the height and width and sets the two data members to the parameter values. You do not need to worry about data validity checking.
- (c) Create sets and gets accessor methods that will set and return the values of the height and width. Four methods total here. Again, you do not need to worry about data validity checking.
- (d) Create a method `Perimeter` that will calculate and return the value of the perimeter of the rectangle.
- (e) Create a method `Area` that will calculate and return the value of the area of the rectangle.
- (f) Create a method `isSquare` that will return true if the rectangle is a square and false if it is not.
- (g) Create a method `toString` that will return a string with the height and width information. Specifically, if we have a rectangle object called `rectangle1` with width 7 and height 2 the code line

```
System.out.println(rectangle1);
```

would produce the output of

```
Rectangle Data: Width = 7.0    Height = 2.0
```

In the main file:

- (a) Create the following method. Note that this method brings in a `Rectangle` object as a parameter. As we discussed in class, these objects are new data types you create and can be used like any other data type. So sending a `Triangle` or `Rectangle` to a method is just as easy as sending an `int` or a `double`.

```

public static void PrintRectangleInformation(Rectangle r) {
    System.out.println("Height: " + r.getHeight());
    System.out.println("Width: " + r.getWidth());
    System.out.println("Area: " + r.Area());
    System.out.println("Perimeter: " + r.Perimeter());
    if (r.isSquare())
        System.out.println("The rectangle is a square.");
    else
        System.out.println("The rectangle is not a square.");
}

```

(b) Insert the following code into the main itself,

```

Scanner kb = new Scanner(System.in);

System.out.print("Input height and width of Rectangle #1: ");
double h = kb.nextDouble();
double w = kb.nextDouble();

Rectangle rectangle1 = new Rectangle(h, w);

System.out.print("Input height and width of Rectangle #2: ");
h = kb.nextDouble();
w = kb.nextDouble();

Rectangle rectangle2 = new Rectangle(h, w);

System.out.println();
System.out.println(rectangle1);
System.out.println(rectangle2);

System.out.println();
System.out.println("Rectangle 1");
PrintRectangleInformation(rectangle1);

System.out.println();
System.out.println("Rectangle 2");
PrintRectangleInformation(rectangle2);

rectangle1.setHeight(17);
rectangle1.setWidth(17);

System.out.println();
System.out.println("Rectangle 1");
PrintRectangleInformation(rectangle1);

```

Run the program and you should get output like the following. You may need to debug your Rectangle class if you have any errors.

```

Input height and width of Rectangle #1: 5 7
Input height and width of Rectangle #2: 6 6

Rectangle Data: Width = 7.0    Height = 5.0
Rectangle Data: Width = 6.0    Height = 6.0

Rectangle 1
Height: 5.0
Width: 7.0
Area: 35.0
Perimeter: 24.0
The rectangle is not a square.

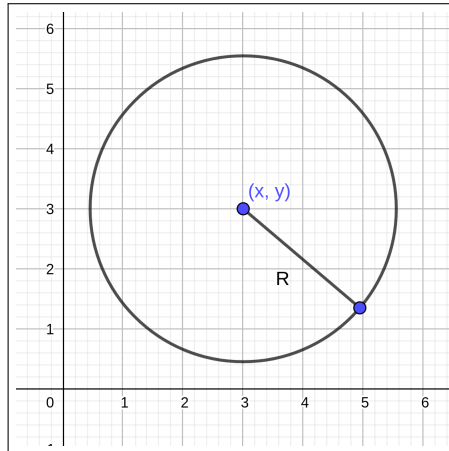
Rectangle 2
Height: 6.0
Width: 6.0
Area: 36.0
Perimeter: 24.0
The rectangle is a square.

Rectangle 1
Height: 17.0
Width: 17.0
Area: 289.0

```

```
Perimeter: 68.0  
The rectangle is a square.
```

2. In this exercise you will be creating a Circle object (class) that will be similar to the triangle object we created in class. The data that is being stored in the Circle object are the  $x$  and  $y$  coordinates for the center and the radius of the circle, all doubles.



All data members must be declared private. The methods for the class are as follows.

- (a) One constructor is simply a default constructor that sets the numeric data members to 0.

```
public Circle()
```

- (b) The second constructor sets the values of the data members to the parameter values.

```
public Circle(double cx, double cy, double rad)
```

- (c) An accessor method for setting the  $x$  and  $y$  coordinates of the circle.

```
public void setCenter(double cx, double cy)
```

- (d) An accessor method for setting the radius of the circle.

```
public void setRadius(double rad)
```

- (e) An accessor method for getting the  $x$  coordinates of the circle.

```
public double getCenterX()
```

- (f) An accessor method for getting the  $y$  coordinates of the circle.

```
public double getCenterY()
```

- (g) An accessor method for getting the radius of the circle.

```
public double getRadius()
```

- (h) A method to calculate and return the circumference of the circle.

```
public double Circumference()
```

- (i) A method to calculate and return the area of the circle.

```
public double Area()
```

- (j) A method that will determine if two circles collide.

```
public boolean collide(Circle c)
```

We will talk about this one a little. Notice that the parameter is a circle type, so in the method you will be working with two circles at one time, the parameter circle *c* and the “calling” circle. Let’s say that you declared two circles in the main, *c1* and *c2*. Then to determine if the two circles collide you would have the command,

```
c1.collide(c2)
```

If the value of this is true then the two circles overlap and false if they do not. In this call the circle *c2* would be loaded into the parameter *c* in the method. If you have a data member of radius in the class structure, then the value of *radius* is the radius of *c1* and the value of *c.radius* is the radius of *c2*.

To determine if two circles collide you calculate the distance between the two centers of the circles and if that distance is less than the sum of the two radii then the circles are colliding, think about that geometrically. Recall that the distance between two points  $(x_1, y_1)$  and  $(x_2, y_2)$  is  $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ .

In the main program file you will also write a method called `printProperties` that will take as a parameter a circle object and print out its attributes. Your main program is to be the following.

```
public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);

    System.out.print("Input the X coordinate of the center: ");
    double cx = keyboard.nextDouble();
    System.out.print("Input the Y coordinate of the center: ");
    double cy = keyboard.nextDouble();
    System.out.print("Input the radius: ");
    double rad = keyboard.nextDouble();

    Circle circle1 = new Circle(cx, cy, rad);
    printProperties(circle1);

    Circle circle2 = new Circle(2, 3, 2);
    printProperties(circle2);
    Circle circle3 = new Circle(4, 5, 1);
    printProperties(circle3);
    Circle circle4 = new Circle(0, 0, 3);
    printProperties(circle4);

    System.out.println();
    System.out.println(circle2.collide(circle3));
    System.out.println(circle2.collide(circle4));
    System.out.println(circle3.collide(circle4));
}
```

**Program Run:** The following is a run of the program.

```
Input the X coordinate of the center: 4
Input the Y coordinate of the center: 7
Input the radius: 5
```

```
Properties
Center = (4.0, 7.0)
Radius = 5.0
Circumference = 31.41592653589793
```

```
Area = 78.53981633974483

Properties
Center = (2.0, 3.0)
Radius = 2.0
Circumference = 12.566370614359172
Area = 12.566370614359172

Properties
Center = (4.0, 5.0)
Radius = 1.0
Circumference = 6.283185307179586
Area = 3.141592653589793

Properties
Center = (0.0, 0.0)
Radius = 3.0
Circumference = 18.84955592153876
Area = 28.274333882308138

true
true
false
```

3. In this exercise you will be creating a Student object (class) that will hold some student information, such as, name, id, and exam scores. The class will also have methods for updating and retrieving student information. The main program will act as a grade book, for a class of a single student.

The data that is being stored in the student object is the first and last names of the student, the student's ID number (as an integer), three exam scores and a final exam score, all doubles. All data members must be declared private. The methods for the class are as follows.

- (a) The constructor is simply a default constructor that sets the strings to the empty string and the numeric data members to 0.

```
public Student()
```

- (b) The accessor method setName will update the first and last names.

```
public void setName(String first, String last)
```

- (c) The accessor method getName will return the student's full name in informal form.

```
public String getName()
```

- (d) The accessor method getFormalName will return the student's full name in formal form.

```
public String getFormalName()
```

- (e) The accessor method setID will update the student's ID number.

```
public void setID(int id)
```

- (f) The accessor method getID will return the student's ID number.

```
public int getID()
```

- (g) The accessor method `setExamScore` will store the score for the given exam number. If the exam number is not 1, 2, or 3 then the score cannot be stored. In addition, the exam scores are to be between 0 and 100, and can be doubles. If the exam score is less than 0 store 0 and if the score is greater than 100 store 100.

```
public void setExamScore(int examNumber, double score)
```

- (h) The accessor method `getExamScore` will return the score for the given exam number. If the exam number is not 1, 2, or 3 then return 0.

```
public double getExamScore(int examNumber)
```

- (i) The accessor method `setFinalExamScore` will store the score for the final exam. The exam score is to be between 0 and 200, and can be double. If the exam score is less than 0 store 0 and if the score is greater than 200 store 200.

```
public void setFinalExamScore(double score)
```

- (j) The accessor method `getFinalExamScore` will return the score for the final exam.

```
public double getFinalExamScore()
```

- (k) The method `ExamAverage` will calculate and return the student's exam average for the three 100 point exams. It will not include the final exam.

```
public double ExamAverage()
```

- (l) The method `Average` will calculate and return the student's exam average. This one does include the final exam average.

```
public double Average()
```

- (m) The method `FinalProjection` will calculate and return the score needed on the final (out of 200) that the student needs to get the target average for the course.

```
public double FinalProjection(double target)
```

- (n) The method `LetterGrade` will calculate the student's average and then return the student's current letter grade on the 90-80-70-60 point scale.

```
public String LetterGrade()
```

The main program will handle all input from the keyboard and all printing to the console window. There will be no printing from the Student class or keyboard input into the Student class.

When the program is run the user will see the following menu.

Please select from the following menu:

1. Input Student Name
2. Input Student ID
3. Input Exam Score
4. Input Final Exam Score
5. Print Exam Average without the Final Exam
6. Calculate Final Exam Projection
7. Print Student Report
8. Quit

Selection:

The first four items will take input from the user and then the main will call the appropriate method to update the student information. Items 5 and 6 will print out the requested information and item 7 will print out a complete student report in the form.

```
Student Progress Report: Spickler, Don   (12345)
Exam Scores
Exam 1 = 84.0
Exam 2 = 78.0
Exam 3 = 70.0
Final = 174.0
Average = 81.2
Letter Grade = B
```

This report is to be printed in a method (in the main program file) that takes in a single parameter of Student type. The menu and the menu selection are to be done in a separate method as well. This method will have no parameters but it will return an integer that designates the user's selection. Also, this method should check the user's input and if it is a number out of range the method should ask the user to make another selection.

One idiosyncrasy of the scanner is the difference between the `nextLine` reads and the numeric reads. When reading a numeric input the enter key that is at the end of the input stays in the input buffer but when a `nextLine` is done the enter key is removed from the buffer. What this means is that if you were to read in a numeric value and then a string (using `nextLine`), the string read would take the enter key without stopping for user input. For example, in the following block of code, the `nextLine` read does not stop and wait for user input. The enter is taken in and the `str` variable is the empty string.

```
System.out.print("Input a number: ");
double num = keyboard.nextDouble();
System.out.print("Input a string: ");
String str = keyboard.nextLine();
System.out.println(str + "    " + num);
```

The output of the above block of code follows.

```
Input a number: 12.34
Input a string:    12.34
```

If we add in a single line, `String clearBuffer = keyboard.nextLine();` after the numeric read then the enter key is removed and the next call to `nextLine` will stop and wait for user input.

```
System.out.print("Input a number: ");
double num = keyboard.nextDouble();
String clearBuffer = keyboard.nextLine();
System.out.print("Input a string: ");
String str = keyboard.nextLine();
System.out.println(str + "    " + num);
```

The output of the above block of code follows.

```
Input a number: 12.34
Input a string: This is my String
This is my String    12.34
```

So if in your code you are calling `nextLine` after a numeric read you may wish to use the `clear` buffer line as we did above. For example, if you select to update the student's id number and then select to update the student's name, this could easily happen.

All programming languages have idiosyncrasies, the job of programming is to make the program do what you want it to do given the tools of the language. Sometimes this means coming up with a work-around to some things.

**Program Run:** The following is a complete run of the program.

```
Please select from the following menu:
1. Input Student Name
2. Input Student ID
3. Input Exam Score
4. Input Final Exam Score
5. Print Exam Average without the Final Exam
6. Calculate Final Exam Projection
7. Print Student Report
8. Quit
```

```
Selection: 1
```

```
Input Student's First Name: Don
Input Student's Last Name: Spickler
```

```
Please select from the following menu:
1. Input Student Name
2. Input Student ID
3. Input Exam Score
4. Input Final Exam Score
5. Print Exam Average without the Final Exam
6. Calculate Final Exam Projection
7. Print Student Report
8. Quit
```

```
Selection: 2
```

```
Input Student's ID: 12345
```

```
Please select from the following menu:
1. Input Student Name
2. Input Student ID
3. Input Exam Score
4. Input Final Exam Score
5. Print Exam Average without the Final Exam
6. Calculate Final Exam Projection
7. Print Student Report
8. Quit
```

```
Selection: 3
```

```
Exam Number: 1
Score: 84
```

```
Please select from the following menu:
1. Input Student Name
2. Input Student ID
3. Input Exam Score
4. Input Final Exam Score
5. Print Exam Average without the Final Exam
6. Calculate Final Exam Projection
7. Print Student Report
8. Quit
```

```
Selection: 3
```

```
Exam Number: 2
```



Score: 78

Please select from the following menu:

1. Input Student Name
2. Input Student ID
3. Input Exam Score
4. Input Final Exam Score
5. Print Exam Average without the Final Exam
6. Calculate Final Exam Projection
7. Print Student Report
8. Quit

Selection: 3

Exam Number: 3

Score: 70

Please select from the following menu:

1. Input Student Name
2. Input Student ID
3. Input Exam Score
4. Input Final Exam Score
5. Print Exam Average without the Final Exam
6. Calculate Final Exam Projection
7. Print Student Report
8. Quit

Selection: 5

Exam Average = 77.33333333333333

Please select from the following menu:

1. Input Student Name
2. Input Student ID
3. Input Exam Score
4. Input Final Exam Score
5. Print Exam Average without the Final Exam
6. Calculate Final Exam Projection
7. Print Student Report
8. Quit

Selection: 6

Target Average: 90

Final exam score needed to have a 90.0%  
at the end of the course is 218.0 out of 200.

Please select from the following menu:

1. Input Student Name
2. Input Student ID
3. Input Exam Score
4. Input Final Exam Score
5. Print Exam Average without the Final Exam
6. Calculate Final Exam Projection
7. Print Student Report
8. Quit

Selection: 6

Target Average: 80

Final exam score needed to have a 80.0%  
at the end of the course is 168.0 out of 200.

Please select from the following menu:

1. Input Student Name
2. Input Student ID
3. Input Exam Score
4. Input Final Exam Score

5. Print Exam Average without the Final Exam
6. Calculate Final Exam Projection
7. Print Student Report
8. Quit

Selection: 6

Target Average: 70  
Final exam score needed to have a 70.0%  
at the end of the course is 118.0 out of 200.

Please select from the following menu:

1. Input Student Name
2. Input Student ID
3. Input Exam Score
4. Input Final Exam Score
5. Print Exam Average without the Final Exam
6. Calculate Final Exam Projection
7. Print Student Report
8. Quit

Selection: 6

Target Average: 60  
Final exam score needed to have a 60.0%  
at the end of the course is 68.0 out of 200.

Please select from the following menu:

1. Input Student Name
2. Input Student ID
3. Input Exam Score
4. Input Final Exam Score
5. Print Exam Average without the Final Exam
6. Calculate Final Exam Projection
7. Print Student Report
8. Quit

Selection: 6

Target Average: 40  
You already have enough points for a 40.0%.

Please select from the following menu:

1. Input Student Name
2. Input Student ID
3. Input Exam Score
4. Input Final Exam Score
5. Print Exam Average without the Final Exam
6. Calculate Final Exam Projection
7. Print Student Report
8. Quit

Selection: 4

Final Exam Score: 174

Please select from the following menu:

1. Input Student Name
2. Input Student ID
3. Input Exam Score
4. Input Final Exam Score
5. Print Exam Average without the Final Exam
6. Calculate Final Exam Projection
7. Print Student Report
8. Quit

Selection: 7

```
Student Progress Report: Spickler, Don (12345)
Exam Scores
Exam 1 = 84.0
Exam 2 = 78.0
Exam 3 = 70.0
Final = 174.0
Average = 81.2
Letter Grade = B
```

Please select from the following menu:

1. Input Student Name
2. Input Student ID
3. Input Exam Score
4. Input Final Exam Score
5. Print Exam Average without the Final Exam
6. Calculate Final Exam Projection
7. Print Student Report
8. Quit

Selection: 8