Program This program is meant to be your first basic Object-Oriented Program (OOP). You will write 2 of your own classes, each of which represent a 2D shape and inherit some methods from a base class. You will then tie the program together with a Main class which has been partially written and provided to you. See the Example Output for examples of program runs.

1. First you will have to create your own project and add the provided source files to your project. We have not provided any project files in this assignment. You must create your own project. However, you must keep all source files in the folder called src which is one directory in from the root of your repository (i.e. YourRepo/src). Also, do not commit multiple copies of the same name file to your repository. All of this is important for grading purposes
2. Next, take a good look at the 2 files that have been provided already, Main.java and Shape.java. The Main class has some comments that will guide you through coding that file. The Shape class will serve as a base class for the 2 new classes you will create, Rectangle and Circle. Do not make any changes to the file Shape.java. In Main.java, notice that an array of Shape objects has already been provided, so you do not need to change that either. However, Shape itself is an abstract class (meaning we cannot make instance objects of it). This should be an indication to you that you will need to create your own classes that extend (inherit from) Shape and then you will make instance objects of those classes and put them into the Shape array provided. For example, when you create a class called Rectangle that extends Shape, then you can make instance objects of type Rectangle and put them in the Shape array because objects of type Rectangle are also of type Shape. As you’ve learned in lecture and in the book, this is called polymorphism.

(

c) Start by creating your Rectangle class in its own file, Rectangle.java. It should inherit from Shape. It should contain 4 private member variables, all of type double, representing: the top-left coordinate x value, the top-left coordinate y value, the width of the rectangle, and the height of the rectangle. It should also have a single constructor that takes 4 parameters of type double that should be assigned to the appropriate private member variables. If the width or the height passed in to the constructor is less than 1, then make it 1, because users of this class shouldn’t be able to create Rectangle objects with a width or height of less than 1. By doing this, you are encapsulating the private member variables and protecting their values. Though we have also learned about accessor and mutator methods when talking about encapsulation, you don’t actually need to provide any in this class (unless you want to). However, you must override the methods area() and perimeter() so that they compute and return the proper computation per shape object. By overriding these base class methods from Shape in the child class Rectangle, you ensure that objects of this child class provide specific functionality per type, another example of polymorphism. See Shape for the method signatures. Finally, override the toString() method so it returns a String that describes the Rectangle, such as the word “Rectangle” plus the values of the top-left coordinate and the width and height. If you do not override toString(), then when you print a Rectangle object from Main, the output will not look like what you expect.

1. Next, create your Circle class in the same fashion
2. Once the classes Rectangle and Circle are defined properly, you can work on Main. Complete the fillShapes() method and call it from main() passing the array of shapes already created. This method should fill in the array that was passed to it. By fill in, we mean, if the specified they wanted 3 shapes (code already provided in main()), then you need to actually create 3 object instances and put them in the array at indices 0, 1, and 2. Since the fillShapes() method is technically a procedure, you will fill the array in the method, not in main(). Iterate through the array parameter passed into the method, and for each index of the array, ask the user if it should be a Circle or not. If so, ask them for the x, y and radius, and then create a Circle object and assign it into the array at the current index. If not, then do the same but with a Rectangle object. See the Example Output for an example of the input/output.
3. Next, complete the printShapes() method and call it from main() passing the array of shapes. This method simply iterates through the array passed in and prints out each index followed by the Shape object itself. If you have overridden the toString() method for each class properly, then the objects should print out as you expect
4. Now finish the rest of the code in Main. Call the menu() method already provided (once). Then complete the code in the while loop. Each loop iteration, ask the user their menu selection, read in their choice, and handle it appropriately. If they indicate to EXIT, then break out of the loop. Otherwise ask the user which shape they are interested in, and read in the index as they specify. The user’s input here is an integer representing the index of the shape they are interested in. Use a switch statement to handle the remaining choices, Area or Perimeter, each of which should call the area() or perimeter() methods of the shape at the specified index, respectively, and print out the result.

Rubric Task Percentage Following submission instructions 10% General attempt at solving the problem 40% Program compiles and runs 10% Properly created Rectangle and Circle classes 20% Properly created and called fillShapes and printShapes methods 20% Total 100%

Diagram

Description automatically generated

A picture containing graphical user interface

Description automatically generated

A picture containing text

Description automatically generated