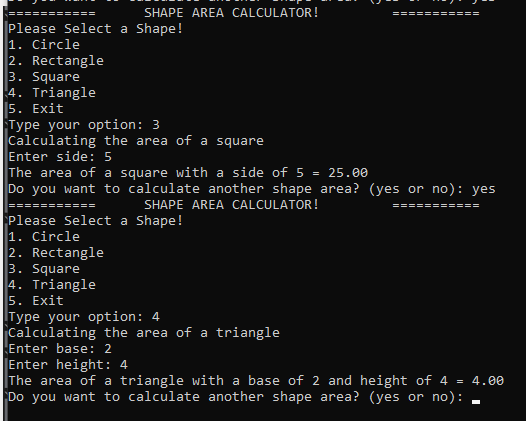
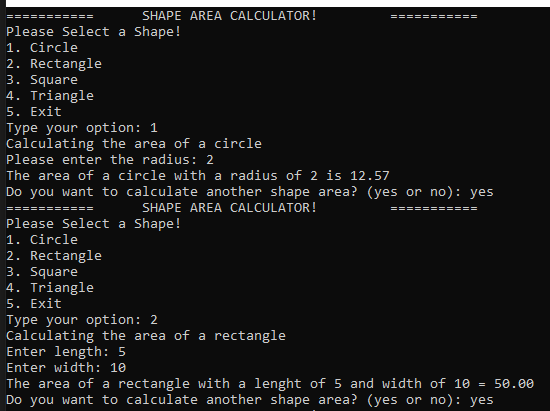
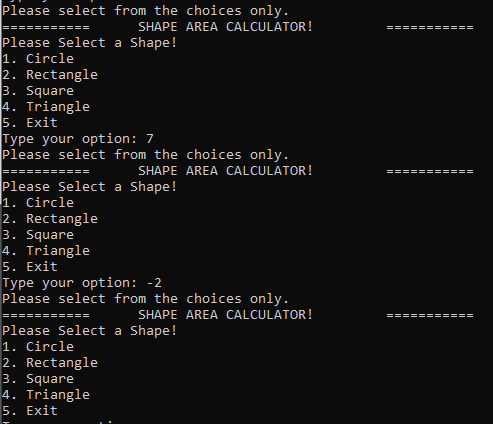
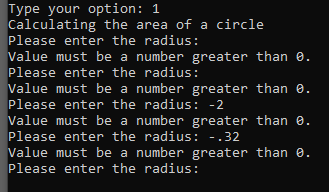


**OUTPUTS**

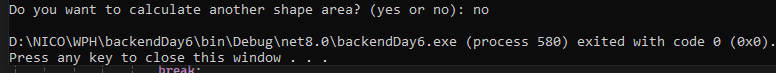
****

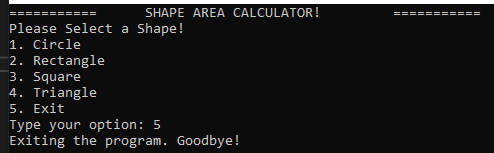
**ERROR HANDLING**

****

****

**IF CHOICE = NO**

****

****

**REFLECTION**

In this activity, I used polymorphism by creating a base class called Shape that has a method named CalculateArea().I created separate classes for each shape like Circle, Rectangle, Square, and Triangle, and used the override keyword in each one to create their own version of CalculateArea() using the correct formula. This allows me to use a Shape variable in the Main method and still call the correct method depending on the user’s choice. That is how I applied polymorphism by using the same method name but with different behaviors depending on the object.

For user input, I made a static class called InputHelper with a method called ReadInput(). It uses a do while loop that keeps asking the user until they enter a valid number that is greater than zero. This helped make sure the program does not crash or accept wrong input. I also used variables like operation to store the user’s selected shape, and inside each shape class I stored the needed values like radius, length, width, base, or height. After the user chooses a shape from the menu, I used a switch statement to create the right shape object and store it in a Shape variable. Then outside the switch, I called shape.CalculateArea() which automatically runs the correct version of the method depending on what shape was created.

One thing I learned from this activity is how useful polymorphism is. At first, I thought I had to call each shape method separately, but now I understand that one method call can do all the work if the setup is correct. I did get a bit confused at first when I called the method inside each case, but after I moved the method call outside the switch, the code became much cleaner. I also found it helpful to have a separate class just for input so I did not have to repeat code every time I needed user input. This activity really helped me understand how object-oriented programming makes code more flexible, organized, and easier to manage.