OOP Spring 2023 Solutions
HW #1

(1) Write a Rectangle class. The class has the following properties.

(a) There are two data members: length and width.

(b) Write a constructor for the class.

(c) Write member functions that compute the area and perimeter of the Rectangle.

(d) Write setter and getter functions for the length and width.

(e) Write a member function to print the length and width.

(f) Write a test program to test your class.

(g) You **must** write the class and test program in all of the following languages: C++, C#, Groovy, Java, and Python 3.

SOLUTION

C++

```
#include <iostream>
#include <stdexcept>
using namespace std;

class Rectangle {
	int length, width;
public:
	Rectangle(int l, int w): length(l), width(w) {
		if (length < 0 || width < 0)
			throw invalid_argument("Length and width must be positive");
	}

	int area() const { return length * width; }
	int perimeter() const { return 2*(length+width); }

	int getLength() const { return length; }
	int getWidth() const { return width; }
	void setLength(int l) {
		if (l < 0)
			throw invalid_argument("Length must be positive");
		else
			length = l;
	}
```

```
		void setWidth(int w) {
			if (w < 0)
				throw invalid_argument("Width must be positive");
			else
				width = w;
		}
		void print() const {
			cout << "Rectangle length = " << length << " Width = " << width << endl;
		}
};

int main() {
		Rectangle r(3,4);

		r.print();
		cout << "Area = " << r.area() << endl;
		cout << "Perimeter = " << r.perimeter() << endl;

		r.setLength(10);
		r.setWidth(5);
		r.print();

		cout << "Length = " << r.getLength() << endl;
		cout << "Width = " << r.getWidth() << endl;

		try {
			Rectangle s(-3,-4);
		}
		catch(const invalid_argument &ex) {
			cout << "Caught error: " << ex.what() << endl;
		}


		return 0;
}
```

C#

```
using System;

namespace ConsoleApplication1
{
    class Program
    {
```

```
    static void Main(string[] args)
    {
        Rectangle r;

        r = new Rectangle(8,12);
        Console.WriteLine(r.print());
        Console.WriteLine("Area = "+r.area()+" Perimeter = "+r.perimeter());

        Rectangle r2;

        r2 = new Rectangle(10,5);
        Console.WriteLine(r2.print());
        Console.WriteLine("Area = "+r2.area()+" Perimeter = "+r2.perimeter());

        r2.Length = 22;
        r2.Width = 4;
        Console.WriteLine(r2.print());
        Console.WriteLine("Area = "+r2.area()+" Perimeter = "+r2.perimeter());

                        try
                        {
                                Rectangle s;
                                s = new Rectangle(-3,-4);
                        }
                        catch(Exception e)
                        {
                                Console.WriteLine(e);
                        }
        Console.ReadLine();

    }
}

class Rectangle
{
    private int length, width;

    public Rectangle(int length, int width)
    {
         if (length < 0 || width < 0) {
         throw new Exception("Length and width must be positive");
        }
        else {
          this.length = length;
```

```
            this.width = width;
          }
        }

        public int area()
        {
            return length*width;
        }

        public int perimeter()
        {
            return 2*(length+width);
        }

        public string print()
        {
          return "Rectangle length " + Length + " Rectangle width " + Width;
        }

        public int Length
        {
          get { return length; }
          set { length = value; }
        }

        public int Width
        {
            get{ return width; }
            set{ width = value; }
        }
      }
}
```

Java

```
import java.io.*;
class Rectangle {

  int length, width;

  public Rectangle(int length, int width) {
      this.length = length;
      this.width = width;
  }
  public void setLength(int l) {
        length = l;
  }

  public void setWidth(int w) {
          width = w;
  }

  public int getLength() {
     return length;
  }

  public int getWidth() {
     return width;
  }

  public void print() {
     System.out.println("Length:"+ length );
     System.out.println("Width:" + width );
  }

  public int area() {
    return length*width;
  }

  public int perimeter() {
    return 2*(length+width);
  }

}

public class EmployeeTest {

  public static void main(String args[]) {
    Rectangle r1 = new Rectangle(5,3);
    Rectangle r2 = new Rectangle(4,10);

    r1.print();
    r2.print();

    System.out.println("Area = "+r1.area());
    System.out.println("Perimeter = "+r1.perimeter());
```

```
      r2.setLength(10);
      r2.setWidth(5);
      r2.print();

      System.out.println("Length = "+r2.getLength());
      System.out.println("Width = "+r2.getWidth());

   }
}
```

Groovy

```
class Rectangle {
     def int length
     def int width
     int area() {
              return length*width
     }
     int perimeter() {
              return 2*(length+width)
     }
     void print() {
              println("Rectangle length = "+length+" Rectangle width = "+width)
     }
 }

 def rect = new Rectangle()
 rect.length = 10
 rect.width = 6
 rect.print()

 println("Rectangle area = "+rect.area())
 println("Rectangle perimeter = "+rect.perimeter())
 println("New length = "+(rect.length = 12))

 try {
              rect.length = -3
              if (rect.length < 0) {
                           throw new Exception("Length and width must be positive")
              }
 }
 catch(Exception e) {
                  println("Exception: ${e}")
 }
```

Python

```
class Rectangle:
	def __init__(self,length,width):
		self._length = length
		self._width = width
		if self._length < 0 or self._width < 0:
			raise Exception("Length and width must be positive")
	def area(self):
		return self._length*self._width
	def perimeter(self):
		return 2*(self._length+self._width)
	def print_Rectangle(self):
		print("Rectangle length = ", r1._length, " Rectangle width = ", r1._width)
	@property
	def length(self): #getter for length
		return self._length
	@length.setter
	def length(self,value): #setter for length
		self._length = value
	@property
	def width(self): #getter for width
		return self._width
	@width.setter
	def width(self,value): #setter for width
		self._width = value



r1 = Rectangle(12,7)
r1.print_Rectangle()
print("Rectangle area = ", r1.area())
print("Rectangle perimeter = ", r1.perimeter())

r1.length = 23
r1.width = 13
print("New rectangle length = ", r1.length, " New rectangle width = ", r1.width)

try:
	r2 = Rectangle(-2,5)
except Exception as e:
	print("Error: ", e)
```

The following are short answer questions.

(2) Which of the following C++ mechanisms is used to create polymorphism?

Virtual functions ______X__________ Templates ________X__________

Virtual functions exemplify **inclusion polymorphism.** A reference or pointer to a parent object may be converted to an object of any inherited type. This means that determining which method is being called is a run-time decision.

Templates implement the concept of **parametric polymorphism.** Code works correctly for a variety of datatypes. Functional languages such as ML, Haskell, Lisp typically have parametric polymorphism.

(3) Which of the following conversions is generally not legal?
(a) Converting a base object to a derived object?
Ans: Not legal (the base object is not a derived object).

(b) Converting a derived object to a base object?
Ans: Legal (the derived object is a base object).

(4) True or false: it is illegal for a destructor to throw an exception in C++.

Ans: False, but it is generally a bad idea.

(5) What is the difference between function overloading and function overriding?

Ans: Overloading involves functions of the same name having different signatures. Overriding occurs when a function in a derived class replaces a function of the same name in the base class.

(6) What is a predicate in C++?
Ans: A function or function object that returns a boolean.

(7) What is a function object?

Ans: A function object is an object that may be called like a function. In C++ this is achieved by overloading the () operator.

(8) How does a function object differ from a function?

Ans A function object has state. It can be initialized through a constructor and retains its state between invocations.

(9) Explain the difference between overloading and polymorphism.

Ans: Overloading creates many functions of the same name that work on different types. Polymorphism creates one function that works on a variety of datatypes.

(10) Why are namespaces used in C++?

Ans. Namespaces provide a method for preventing name conflicts in large projects. Objects declared inside a namespace block are placed in a named scope that prevents them from being mistaken for identically-named symbols in other scopes.

(11) True or false: Only object-oriented programming languages can exhibit polymorphism.

Ans. False, many languages exhibit polymorphism. In Lisp, car and cons are examples of polymorphic functions.