University of Central Punjab

(Incorporated by Ordinance No. XXIV of 2002 promulgated by Government of the Punjab)
FACULTY OF INFORMATION TECHNOLOGY

CSCP2023 - Object Oriented Programming
Spring 2022

Course Term Project Marks: 100	
Topic	1. Classes: getters/setters, default, parameterized and copy constructor 2. All Operators: Math(+, -, *, /), Comparison (==, !=, <=, >=) and Assignment (=), Unary(++, --), Indexing [], Insertion >>, Extraction << 3. Inheritance, Polymorphism, Abstract Class 4. UML class diagram
Objective/ Outcome	Check Students' understanding of Class, Object, UML, Inheritance, Polymorphism and Abstract Class

Instructions:

- Indent your code.
- Comment your code.
- Use meaningful variable names.
- Plan your code carefully on a piece of paper before you implement it.

Requirements:

- Name of each .h and .cpp file for classes should have the same name as that of the class e.g. Person.h (declaration of Class Person) and Person.cpp (Code of the class)
- Main function must be in a separate file: "main.cpp"
- All non-class (global functions) must be declared in "util.h" and coded in "util.cpp"
- **void main()** is not allowed. Use **int main()**
- You are not allowed to use system("pause")

For each sub-problem, please compile and run your code and provide the screen shots of your test runs (20 % marks for each problem).

You are required to follow the naming conventions as follow:

Variables: firstName; (first letter must be lowercase and no underscores allowed)
Function: getName(); (first letter must be lowercase and no underscores allowed)
ClassName: BankAccount (first letter must be UPPERCASE and no underscores allowed)

Shape Drawing System

Objectives

- Check knowledge and skill of writing Operators: Math(+, -, *, /), Comparison (==, !=, <=, >=) and Assignment (=), Indexing Operator [], pre and post ++ & --, unary + & -, extraction << and insertion >>
- Use Abstract Classes with Pure Virtual Functions and Polymorphism

Problem

You will be developing an application that draws the following geometrical shapes on screen using ASCII character printing.

- Vertical Lines e.g. line of length 5 is shown below:

  *
  *
  *
  *
  *

- Horizontal Lines e.g. line of length 4 is shown below:

  *****

- Shaded and Unshaded Rectangle e.g. Shaded rectangle of length 5 and width 3 is shown below:

  *****
  *****
  *****

- And Un-Shaded Rectangle of length 5 and width 3 is shown below:

  *****

- * *

  *****

- Up or down, Up Triangle of base 5 and height 3 is shown below:

  *
  **
  ***
  *****

  And down Triangle of base 5 and height 3 is shown below:

  *****
  ****
  ***
  *

You will provide a menu to the user and based on the menu selection one of the above shapes will be printed on the screen (see the sample output below).

How to Do

1. Define an Abstract Class Shape with the following attributes:
    a. Private Variables:
        i. Length e.g. 5
        ii. Character for drawing the ASCII shape e.g. '*'
    b. Public Functions
        i. Default, Parameterized and Copy Constructor
        ii. Get/Set for all private variables
        iii. Pure Virtual Function "read" that reads shape values from the user
        iv. Pure Virtual Function "display" that displays shape values on the screen
        v. Pure Virtual Function "render" that renders (draws) the shape on the screen
    c. Operators
        i. Assignment =
        ii. Comparison == and !=
        iii. Math + that adds number of characters
        iv. Insertion >> that reads shape values from the user
        v. Extraction << that displays shape values on the screen
2. Define a Class Line that inherits from Shape with the following attributes:
    a. Private Variables:
        i. Flag to indicate Orientation of Line "horizontal" or "vertical"
    b. Public Functions
        i. Default, Parameterized and Copy Constructor
        ii. Get/Set for all private variables
        iii. Function "read" that reads Line values from the user
        iv. Function "display" that displays Line values on the screen
        v. Function "render" that renders (draws) the Line on the screen
    c. Operators
        i. Assignment =
        ii. Comparison == and !=
        iii. Pre and post ++ & -- that increments or decrements length (from parent Shape)
        iv. Insertion >> that reads shape values from the user
        v. Extraction << that displays shape values on the screen
3. Define a Class Rectangle that inherits from Shape with the following attributes:
    a. Private Variables:
        i. Width e.g. 4
        ii. Flag to indicate shaded or not (number of points must be adjusted accordingly)
    b. Public Functions

i. Default, Parameterized and Copy Constructor
ii. Get/Set for all private variables
iii. Function “read” that reads Rectangle values from the user
iv. Function “display” that displays Rectangle values on the screen

c. Function “render” that renders (draws) the Rectangle on the screen
d. Operators
    i. Assignment =
    ii. Comparison == and !=
    iii. Insertion >> that reads shape values from the user
    iv. Extraction << that displays shape values on the screen
    v. Math + that adds length (from parent Shape) and width of two lines
    vi. Pre and post ++ and - - that increments and decrements length (from parent Shape) and width

4. Define a Class Triangle that inherits from Shape with the following attributes:
    a. Private Variables:
        i. Height e.g. 4
        ii. Flag to indicate “up” or ‘down” triangle
    b. Public Functions
        i. Default, Parameterized and Copy Constructor
        ii. Get/Set for all private variables
        iii. Function “read” that reads Triangle values from the user
        iv. Function “display” that displays Triangle values on the screen
        v. Function “render” that renders (draws) the Triangle on the screen
    c. Operators
        i. Assignment =
        ii. Comparison == and !=
        iii. Insertion >> that reads Triangle values from the user
        iv. Extraction << that displays Triangle values on the screen
        v. Math + that adds base (length from parent Shape) and height
        vi. Pre and post ++ and - - that increments and decrements base (length from parent Shape) and height

Constraints

1. Must use the provided main function in the "main.cpp"
2. Sample output must match

Main Function	Sample Output

`int main()`
`{`
`  const int NUMBER = 6;`
`  Shape * shapes[NUMBER];`
`  shapes[0] = new Line(10, '*', true);`
`  shapes[1] = new Line(10, '+', false);`
`  shapes[2] = new Rectangle(10, '&',5,false);`
`  shapes[3] = new Triangle(9,'%', 6, false);`
`  shapes[4] = new Rectangle(10,'@', 5, true);`
`  shapes[5] = new Triangle(9, '$', 6, true);`
 	
`  cout << "** Line **\n";`
`  Line* pl = (Line*)shapes[0];`
`  Line l1 = *pl;`
`  cout << "Line one:" << l1;`
`  pl = (Line*)shapes[1];`
`  Line l2 = *pl;`
`  cout << "Line two:" << l2;`
`  Line L = l1 + l2;`
`  cout << " Line one + Line two = : " << L;`
 	
`  L = l1++;`
`  cout << " L = l1++ = : " << L;`
`  L = --l1;`
`  cout << " L = --l1 = : " << L;`
 	
`  cout << "l1 == l2: " << (l1 == l2) << endl;`
`  cout << "l1 != l2: " << (l1 != l2) << endl;`
 	
`  cout << "** Triangle **\n";`
`  Triangle* pt = (Triangle*)shapes[3];`
`  Triangle t1 = *pt;`
`  cout << "Triangle one:" << t1;`
`  pt = (Triangle*)shapes[5];`
`  Triangle t2 = *pt;`
`  cout << "Triangle two:" << t1;`
`  Triangle T = t1 + t2;`
	** Line **
	Line one:<Shape: 10 */>
	<Line: 1/>
	Line two:<Shape: 10 +/>
	<Line: 0/>
	Line one + Line two = : <Shape: 20 */>
	<Line: 1/>
	L = l1++ = : <Shape: 10 */>
	<Line: 1/>
	L = --l1 = : <Shape: 10 */>
	<Line: 1/>
	l1 == l2: 0
	l1 != l2: 1
	** Triangle **
	Triangle one:<Shape: 9 %/>
	<Triangle: 6 0/>
	Triangle two:<Shape: 9 %/>
	<Triangle: 6 0/>
	Triangle one + two:<Shape: 18 */>
	<Triangle: 12 0/>
	t1++ = <Shape: 9 */>
	<Triangle: 6 0/>
	--t1 = <Shape: 9 */>
	<Triangle: 6 0/>
	t1 == t2: 0
	t1 != t2: 1
	** Triangle **
	Rectangle one:<Shape: 10 &/>
	<Rectangle: 5 0/>
	Rectangle two:<Shape: 10 @/>
	<Rectangle: 5 1/>
	Rectangle one + two = <Shape: 20 */>
	<Rectangle: 10 0/>
	R = r1++: <Shape: 10 */>
	<Rectangle: 5 0/>
	R = --r1: <Shape: 10 */>
	<Rectangle: 5 0/>
	r1 == r2: 0
	r1 != r2: 1

```
	cout << "Triangle one + two:" << T;

	T = t1++;
	cout << "t1++ = " << T;
	T = --t1;
	cout << "--t1 = " << T;
	cout << "t1 == t2: " << (t1 == t2)
<< endl;
	cout << "t1 != t2: " << (t1 != t2)
<< endl;

	cout << "** Triangle **\n";
	Rectangle* pr =
(Rectangle*)shapes[2];
	Rectangle r1 = *pr;
	cout << "Rectangle one:" << r1;
	pr = (Rectangle*)shapes[4];
	Rectangle r2 = *pr;
	cout << "Rectangle two:" << r2;
	Rectangle R = r1 + r2;
	cout << " Rectangle one + two = "
<<R;
	R = r1++;
	cout << "R = r1++: " << R;
	R = --r1;
	cout << "R = --r1: " << R;
	cout << "r1 == r2: " << (r1 == r2)
<< endl;
	cout << "r1 != r2: " << (r1 != r2)
<< endl;
	// cin on Line, Rectangle and
Triangle
	cin >> L >> R >> T;
	// cout on Line, Rectangle and
Triangle
	cout << L << R << T;
	// polymorphism
	cout << "Invoking through
polymorphism: \n";
	for (int i = 0; i < NUMBER; i++)
	{
		shapes[i]->display();
		shapes[i]->render();
	}

	return 0;
}
```

```
Enter Line Data:
Enter Shape Data:
Enter length and character: 11 *
Enter 0 if Line is Horizontal or 1: 0
Enter Rectangle Data:
Enter Shape Data:
Enter length and character: 13 &
Enter width: 9
Enter 1 for shadded and 0 for not shadded: 0
Enter Triangle Data:
Enter Shape Data:
Enter length and character: 15 %
Enter height: 9
Enter 1 for upsided and 0 for downsided: 1
<Shape: 11 */>
<Line: 0/>
<Shape: 13 &/>
<Rectangle: 9 0/>
<Shape: 15 %/>
<Triangle: 9 1/>
Invoking through polymorphism:
<Shape: 10 */>
<Line: 1/>
**********
<Shape: 10 +/>
<Line: 0/>
+
+
+
+
+
+
+
+
+
+

<Shape: 10 &/>
<Rectangle: 5 0/>
&&&&&&&&&&
&         &
&         &
&         &
&&&&&&&&&&
```

	<Shape: 9 %/> <Triangle: 6 0/> %%%% %%% %% % <Shape: 10 @/> <Rectangle: 5 1/> @@@@@@@@@@ @@@@@@@@@@ @@@@@@@@@@ @@@@@@@@@@ @@@@@@@@@@ <Shape: 9 $/> <Triangle: 6 1/> $ $$ $$$ $$$$ $$$$$ $$$$$$

What to Submit?

1. **Class declaration (.h file) and definition (.cpp file) for each class**
2. **Screen shots of all test runs.**
3. **PNG and XML files for UML class diagram (Use draw.io OR MS Vision only to draw the diagram)**

(Compress all files in a single compressed file and upload it)

Where to Submit?

UCP Portal