University of Technology, Jamaica
School of Computing and Information Technology

OOP Principles Lab Practical Five - Overriding and Overloading

Objective:
The objectives of this lab practical are to allow students to be able to:

- obtain more practice implementing inheritance relationships in code using C++ and Java
- work with overloaded methods in a class
- override methods in an inheritance hierarchy

This lab demonstrates how to override and overload methods in a class. Methods will be overloaded in one class. A method in the base class will be overridden in the derived classes.

Exercise One

Implement the classes depicted in the UML class diagrams below, using C++ and/or Java. A Driver file with main() is also to be implemented. Complete the implementation of the other classes by yourself, then create objects and invoke their methods from main( ). By now, you should know how to implement inheritance in an OOP language – that is, by using the extends keyword in Java and by using : plus the access specifier in C++. For example:

C++
```
class Square : public Shape
{

};
```

Java
```
public class Square extends Shape
{

}
```

In this lab we will look at overriding and overloading (see lecture notes five). To override a method you create another version of the method (in a child class) that has the same method name, signature and return type. To overload a method, you create another version of the method (for example in the same class or in a child class) where the overloaded version has the same method name but either a different signature or return type.

In this lab, we will override the Area( ) method from the parent class by providing other versions in the child classes. The area method will take arguments depending on the type of shape we are handling and display the colour and area of the shape, for example "the area of the purple square is 14.7". We will overload the perimeter function in the triangle class by providing a version that works with equilateral triangles only (P = 3 times any side), and another version that works will all other triangles (P = a + b + c).

Design using UML

Shape
- Color : string
+ Shape() + Area() : void

Circle
- Radius : double
+ Circle(String, double) + Area() : void

Triangle
- Base : double - Height : double - Hypotenuse : double
+ Triangle(String, double, double, double) + Perimeter(double) : void + Perimeter(double, double, double) : void + Area() : void

Square
- Length : double
+ Square(String, double) + Area() : void

Formulas:

Area of an arbiturary shape
– not defined

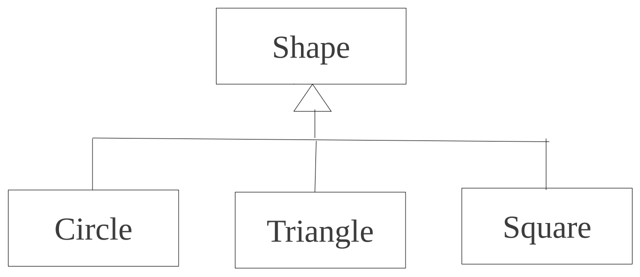
Area of a circle
(A = 3.142 * Radius * Radius)

Area of a Triangle
(A = 0.5 * Base * Height)

Area of a Square
(A = Length * Length)

Perimeter of a Equilateral Triangle
(P = 3 * Side)

Perimeter of any Triangle
(P = a + b + c)



Exercise Two

a) Overload the Perimeter method in the Triangle class so that your overloaded version takes no argument, but uses the attributes of the class (base, height and hypotenuse) to calculate and display the perimeter of the triangle. The method should print something like this – “the perimeter of this blue triangle is 9.6” (if the base=1.2, height=4.5 and hypotenuse=3.9).

b) Invoke your overloaded Perimeter method from main( ).

Exercise Three

Parallelogram
- Base : double - Height : double
+ Parallelogram(String, double, double) + Area() : void

Assume the Parallelogram class above is also a sub class of the Shape class.

a) Add code to your program to implement the Parallelogram class. Override the Area( ) method in the parent class with a version specific for the Parallelogram class. Your overridden version of the Area( ) method in the Parallelogram class should calculate the area and display a message similar to the following - “the area of the yellow parallelogram is 14.8”. The formula for calculating calculating the area of a parallelogram is A = base * height.

b) Add code to your main() in the Driver file to create a new Parallelogram object, then invoke the overridden Area( ) method.

c) Modify your overridden version of the Area( ) method in the Parallelogram class so that it first calls the parent class version of Area( ) before it does anything else (see lecture notes five).

Exercise Four – homework

Modify your Driver file containing main( ) so that it takes information from the user for each shape (example Length for square) then use pass this information entered by the user to the class. Display the area for each shape in main( ) using the information entered. Do NOT accept any input from the user inside of the class methods, instead accept all user input in main( ) and pass it to the class object.