Lab Assignment 9

Exercise 1: **(About inheritance, dynamic binding, and polymorphism)** Write a base class `Shape` and derived classes `Triangle`, `Circle`, and `Rectangle`.

The base class `Shape` has two virtual functions:

```
double getArea();
string getClassName();
```

The << operator should also be overloaded for the class `Shape` only as the second operand.

Please complete the necessary data members, constructors, and functions so that the following main function can produce the corresponding output.

Use the following client code to test your program.

```
int main(){
vector<Shape*> vs;

    Rectangle r(10,20);
    Circle c(10);
    Triangle t(18,30,24);

    vs.push_back(&r);
    vs.push_back(&c);
    vs.push_back(&t);

    for(auto s : vs)
        cout << s;

    return 0;
}
```

The outputs are as follows:

```
Rectangle's area is 200
Circle's area is 314.15
Triangle's area is 216
```

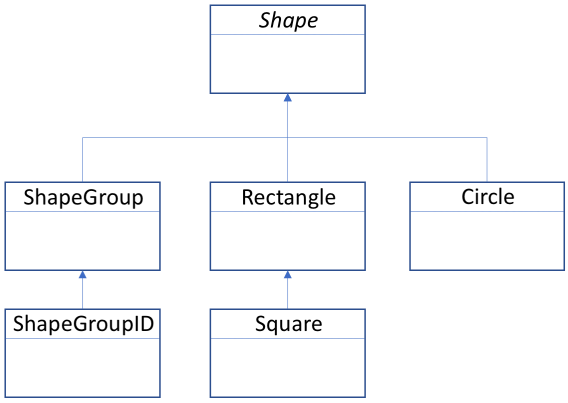
Exercise 2: **(About the rule of three, inheritance, dynamic binding, and polymorphism)**

Object-Oriented Programming Language

05/21/2020

Write a base class `Shape` and derived classes: `ShapeGroup`, `ShapeGroupID`, `Triangle`, `Circle`, and `Rectangle`.

The outline of the inheritance structure is as shown in the following figure:



The `ShapeGroup` and `ShapeGroupID` classes each use a dynamic array[1] to store:

1. pointer to specific array of `Shape*`s and
2. corresponding integer IDs for each stored `Shape` pointers.

To add members into the classes `ShapeGroup` and `ShapeGroupID`, the following function is used:

```
virtual void insert(Shape* s);
```

Make sure that the parent class `Shape` is an Abstract Base Class.

The base class `Shape` has two virtual functions:

```
void printArea();
string getClassName();
```

Please complete the necessary data members, constructors (including the copy constructors), destructors and functions so that the following main function can produce the corresponding output.

Use the following client code to test your program.

```
int main(){
srand(time(0));
    ShapeGroupID sgID;
```

[1]For example a dynamic array storing pointers to `int` IDs. See sample code in the back of this problem.

```
    Rectangle* r1 = new Rectangle(10,20);
    Circle* c1 = new Circle(10);
    Square* s1 = new Square(10);
    sgID.insert(r1);
    sgID.insert(c1);
    sgID.insert(s1);

    ShapeGroupID sgID2;
    Circle* c2 = new Circle(5);
    Square* s2 = new Square(5);
    sgID2.insert(c2);
    sgID2.insert(s2);

    ShapeGroupID sgID3(sgID);
    Circle* c3 = new Circle(20);
    Square* s3 = new Square(20);
    sgID3.insert(c3);
    sgID3.insert(s3);
    sgID3.insert(&sgID2);

    sgID3.printArea();

    delete r1; delete c1; delete s1; delete c2; delete s2;
    delete c3; delete s3;
    return 0;
}
```

The outputs are as follows:

```
ShapeGroup::Copy Constructor
ShapeGroupID::Copy Constructor
Rectangle of id 68's area:
200
Circle of id 6's area:
314.15
Square of id 31's area:
100
Circle of id 62's area:
1256.6
Square of id 34's area:
400
****ShapeGroupID of id 51's area:
Circle of id 88's area:
78.5375
Square of id 75's area:
25

ShapeGroupID::Destructor
ShapeGroup::Destructor
ShapeGroupID::Destructor
ShapeGroup::Destructor
ShapeGroupID::Destructor
ShapeGroup::Destructor
```

Example code on dynamic arrays for pointer storing.

```
#include <iostream>

using namespace std;

int main(){
  int** intDyPtr;
  intDyPtr = new int*[10];

  for(int i = 0; i < 10; ++i)
    intDyPtr[i] = new int(i);

  cout << "Example code for dynamic array storing pointers! " <<
endl;

  // clean up for int elements
  for(int i = 0; i < 10; ++i)
    delete intDyPtr[i];

  // clean up for the pointers
  delete[] intDyPtr;
}
```