C++程序设计入门（上）

第六周编程作业

1.题目内容：（难度：中）

基于第五单元的作业成果，创建一个基类

本作业基于第 5 单元作业【4】的代码修改。

保留 MyScreen 类的声明及实现，不做任何改动

修改 MyRectangle 类及 MyCircle 类

1）将 MyRectangle 类的声明及实现均注释掉，使之不再生效

2）将 MyCircle 类的声明及实现均注释掉，使之不再生效

增加 MyShape 类：

1）颜色处理

a. 将 MyRectangle 与 MyCircle 类中表示颜色的数据域成员挪到 MyShape 类的 ***private* 区域**

b. 将 setColor(int R, int G, int B)函数挪到 MyShape 类中；

c. 将颜色数据成员的 getter 函数挪到 MyShape 类中；如果原 MyRectangle 和 MyCircle 中没有这些 getter 函数，则在 MyShape 类中添加。

2）屏幕处理

a. 将 MyRectangle 与 MyCircle 类中的 Screen*类型的成员挪到 MyShape 类中

b. 将 setScreen 函数也挪到 MyShape 类中

3）MyShape 类构造函数：

a. 根据需要，添加必要的构造函数

b. 所有构造函数均应将颜色初始化为白色，也就是 RGB 三个颜色分量的值均为 255

4）添加一个 string 类型的**【私有】**数据成员 type_

a. 在 MyShape 的构造函数初始化列表中将其初始化为字符串 "myshape"

b. 由你自行决定 type_的访问控制属性，只要它不是 public 访问属性即可

5）在 MyShape 类中添加函数 Draw()，在 Draw()函数中仅仅实现（不要画蛇添足）：

a. 首先输出屏幕的宽和高，中间以**大写的字母"X"**连接，并且放置在一对方括号中，形如：[640X480]

b. 紧随之后输出数据成员 type_的值；

c. 紧随之后再输出使用【半角逗号】分隔的三个颜色。这三个颜色放到一对【半角小括号】中，形如：(255,155,55)，其中不包含任何空格

d. 最后使用 std::endl 换行。

6）根据需要，在 MyShape 类中添加其它函数，例如 getter/setter

输入格式:

空格分隔的两个整数，代表屏幕的宽和高

输出格式:

参见样例。不同输入会导致不同的输出信息

输入样例:

560 120

输出样例:

enter screen

[560X120]myshape(255,255,255)

[560X120]myshape(0,0,255)

leave screen

注意：上面的样例输出一共有 5 行，最后一行为空行

主函数：（不可做任何修改）

```
int main() {
  int width, height;
  cin >> width >> height;

  Screen *screen = Screen::getInstance(width, height);

  MyShape shape1(screen);
  MyShape* shape2 = new MyShape();
```

```
  shape2->setScreen(*screen);
  shape2->setColor(0, 0, 0xff);

  shape1.Draw();
  shape2->Draw();

  delete shape2;
  screen->deleteInstance();

#ifdef DEBUG
  std::cin.get();
#endif
  return 0;
}
```

第五单元作业示例：

以下代码为第五单元作业的参考代码（仅提供了两个图形类的代码）。

建议使用你自己的代码作为本单元作业的基础，尽量不要用本题目所给的代码

```
class MyRectangle {
private:
    int x1_, y1_, x2_, y2_;
    int R_, G_, B_;

    Screen* screen_;

    int getWidth() {
        return x2_ - x1_;
    }

    int getHeight() {
```

```
        return y2_ - y1_;
    }

public:
    MyRectangle (int x1, int y1, int x2, int y2, Screen* screen) {
        x1_ = x1;
        y1_ = y1;
        x2_ = x2;
        y2_ = y2;
        R_ = 255; G_ = 255, B_ = 255;
        screen_ = screen;

        cout << "myrectangle" << endl;
    }

    MyRectangle () {
        x1_ = y1_ = 10;
        x2_ = y2_ = 100;
        R_ = 255; G_ = 255, B_ = 255;

        screen_ = 0;

        cout << "myrectangle" << endl;
    }

    void setCoordinations(int x1, int y1, int x2, int y2) {
        x1_ = x1;
        y1_ = y1;
        x2_ = x2;
        y2_ = y2;
    }
```

```
    void setScreen(Screen& screen) {
        screen_ = &screen;
    }

    void setColor(int R, int G, int B) {
        R_ = R; G_ = G, B_ = B;
    }


    void Draw () {
        cout << x1_ << " " << y1_ << " " <<
                this->getWidth() << " " <<
                this->getHeight() << endl;
        cout << R_ << " " << G_ << " "  << B_ << endl;
    }

    void showScreen() {
        cout << screen_->getWidth() << " " << screen_->getHeight() << endl;
    }

};
class MyCircle {
private:
    int x_, y_, radius_;
    int R_, G_, B_;
    Screen* screen_;

public:
    MyCircle (int x, int y, int radius, Screen* screen) {
        x_ = x;
```

```
        y_ = y;
        radius_ = radius;
        R_ = 255; G_ = 255, B_ = 255;
        screen_ = screen;

        cout << "mycircle" << endl;
    }

    MyCircle () {
        x_ = y_ = 200;
        radius_ = 100;
        R_ = 255; G_ = 255, B_ = 255;
        screen_ = 0;

        cout << "mycircle" << endl;
    }

    MyCircle(const MyCircle& aCircle) {
        x_ = aCircle.x_;
        y_ = aCircle.y_;
        radius_ = aCircle.radius_;
        R_ = aCircle.R_;
        G_ = aCircle.G_;
        B_ = aCircle.B_;
        screen_=aCircle.screen_;
        cout << "copy mycircle" << endl;
    }

    void setCenter(int x, int y) {
        x_ = x;
        y_ = y;
```

```
    }

    void setRadius(int radius) {
        radius_ = radius;
    }

    void setColor(int R, int G, int B) {
        R_ = R; G_ = G, B_ = B;
    }

    void setScreen(Screen& screen) {
        screen_ = &screen;
    }

    void Draw () {
        cout << x_ << " " << y_ << " " << radius_ << endl;
        cout << R_ << " " << G_ << " "  << B_ << endl;
    }

    void showScreen() {
        cout << screen_->getWidth() << " " << screen_->getHeight() << endl;
    }

};
```

2.题目内容：（难度：中）

以 MyShape 作为基类，修改 MyRectangle 类和 MyCircle 类从 MyShape 派生

基于本单元作业【1】的代码，以及第 5 单元作业【4】的 MyRectangle 类和 MyCircle 类进行修改

修改 MyShape 类

1）添加一个成员 showShape()，将 Draw()函数体中的所有代码都移动到 showShape()函数中

2）修改 Draw()函数的声明，使之成为纯虚函数

3）依据下面 MyRectangle 类以及 MyCircle 类的需要，修改 MyShape 的构造函数

修改 MyRectangle 类：

1）以公有方式继承 MyShape 类

2）将所有已经移动到 MyShape 类中的数据成员和函数成员都删除

3）修改所有的构造函数

a. 使之能够将基类中的私有成员 type_的值设置为字符串“myrectangle”

b. 保留对矩形坐标初始化的代码

c. 必要的话，通过基类构造函数和基类成员函数对已经移动到基类中的数据成员进行初始化

d. 删除构造函数中的所有输出语句

4）修改 MyRectangle 类的 **Draw()函数，使之仅仅做如下动作：**

a. 首先调用基类成员函数 showShape()

b. 然后输出矩形左上角顶点以及矩形宽和高并换行。此处保留原有的代码即可，输出格式同第五单元作业【4】

5）删除 showScreen()函数

6）如有必要，则增加其他数据成员及函数成员

7）不要输出任何未要求输出的信息，不然会导致系统扣分。

修改 MyCircle 类：

1）以公有方式继承 MyShape 类

2）将所有已经移动到 MyShape 类中的数据成员和函数成员都删除

3）修改所有的构造函数

a. 使之能够将基类中的私有成员 type_的值设置为字符串“mycircle”

b. 保留对圆心坐标及半径初始化的代码

c. 必要的话，通过基类构造函数和基类成员函数对已经移动到基类中的数据成员进行

初始化

d. **删除构造函数中的所有输出语句**

4）修改 MyCircle 类的 **Draw()函数，使之仅仅做如下动作：**

a. 首先调用基类成员函数 showShape()

b. 然后输出圆心坐标、半径并换行。此处保留原有的代码即可，输出格式同第五单元作业【4】，保持不变

5）删除 showScreen()函数

6）如有必要，则增加其他数据成员及函数成员

7）不要输出任何未要求输出的信息，不然会导致系统扣分。

main() 函数：

需使用如下 main()函数及辅助类 Helper（不得更改）

```
template <typename T>
struct Helper
{
typedef char SmallType;
typedef int LargeType;

template <typename U>
static char Test(U(*)[1]);
template <typename U>
static int Test(...);

const static bool Result = sizeof(Test<T>(NULL)) == sizeof(LargeType);
};

int main() {
  int width, height;
  cin >> width >> height;

  Screen *screen = Screen::getInstance(width, height);
```

```
  if (!Helper<MyShape>::Result) cout << endl;
  MyShape *s1, *s2;

  s1 = new MyRectangle();
  s1->setScreen(*screen);
  s1->setColor(0, 0, 0xff);

  s2 = new MyCircle(100,110,50, screen);

  s1->Draw();
  s2->Draw();

  delete s1, s2;
  screen->deleteInstance();

#ifdef DEBUG
  std::cin.get();
#endif
  return 0;
}
```

输入格式:

空格分隔的整数，代表屏幕宽和高

输出格式:

字符串

输入样例:

```
600 400
```

输出样例：

enter screen

[600X400]myrectangle(0,0,255)

10 10 90 90

[600X400]mycircle(255,255,255)

100 110 50

leave screen

注意：输出样例共有 7 行，最后一行为空行

EGE 绘图练习补充说明

当你做完本次作业，这一季的课程就要结束了，即将迎来期末考试。

不过，我希望你能继续将这次在线编程的代码在课后增加 EGE 的绘图函数，让这些图形真正在屏幕上绘制出来。

你一定会遇到各种困难。但是，成功只属于那些能够克服困难的人。