# EX1

**main1.cpp**

#include <iostream>  
class MyBase1  
{  
public:  
 MyBase1()  
 {  
 std::cout << "…BaseClass1 Object is created!" << std::endl;  
 }  
 ~MyBase1()  
 {  
 std::cout << "…BaseClass1 Object is destroyed!" << std::endl;  
 }  
};  
  
class MyDerived1 : public MyBase1  
{  
public:  
 MyDerived1()  
 {  
 std::cout << "…First layer derived Object is created!" << std::endl;  
 }  
 ~MyDerived1()  
 {  
 std::cout << "…First layer derived Object is Destroyed!" << std::endl;  
 }  
};  
class MyDerived11 : public MyDerived1  
{  
public:  
 MyDerived11()  
 {  
 std::cout << "…Second layer derived Object is created!" << std::endl;  
 }  
 ~MyDerived11()  
 {  
 std::cout << "…Second layer derived Object is destroyed!" << std::endl;  
 }  
};  
int main()  
{  
 MyBase1 a;  
 MyDerived1 b;  
 MyDerived11 c;  
}



**mian2.cpp**

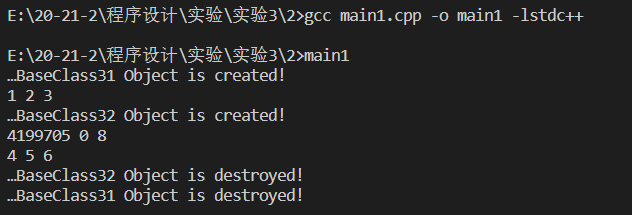
#include <iostream>  
class MyBase1  
{  
public:  
 MyBase1()  
 {  
 std::cout << "…BaseClass1 Object is created!" << std::endl;  
 }  
 ~MyBase1()  
 {  
 std::cout << "…BaseClass1 Object is destroyed!" << std::endl;  
 }  
};  
class MyBase2  
{  
 MyBase1 a1;  
  
public:  
 MyBase2()  
 {  
 std::cout << "…BaseClass2 Object is created!" << std::endl;  
 }  
 ~MyBase2()  
 {  
 std::cout << "…BaseClass2 Object is destroyed!" << std::endl;  
 }  
};  
class MyDerived1 : public MyBase2  
{  
 MyBase1 a1;  
  
public:  
 MyDerived1()  
 {  
 std::cout << "…First layer derived Object is created!" << std::endl;  
 }  
 ~MyDerived1()  
 {  
 std::cout << "…First layer derived Object is Destroyed!" << std::endl;  
 }  
};  
class MyDerived11 : public MyDerived1  
{  
public:  
 MyDerived11()  
 {  
 std::cout << "…Second layer derived Object is created!" << std::endl;  
 }  
 ~MyDerived11()  
 {  
 std::cout << "…Second layer derived Object is destroyed!" << std::endl;  
 }  
};  
int main()  
{  
 MyBase2 a;  
 MyDerived1 b;  
 MyDerived11 c;  
}



# EX2

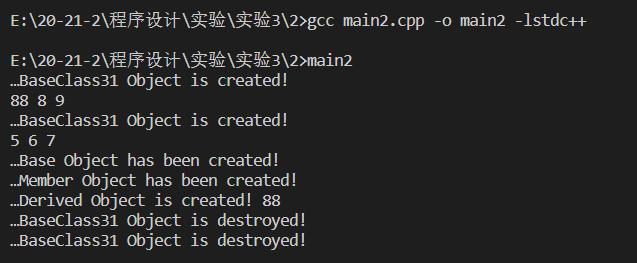
**main1.cpp**

#include <iostream>  
class MyBase31  
{  
 int a, b, c;  
  
public:  
 MyBase31(int x, int y, int z) : a(x), b(y), c(z)  
 {  
 std::cout << "…BaseClass31 Object is created!" << std::endl;  
 std::cout << a << " " << b << " " << c << std::endl;  
 }  
 ~MyBase31()  
 {  
 std::cout << "…BaseClass31 Object is destroyed!" << std::endl;  
 }  
};  
class MyBase32  
{  
 int a, b, c;  
  
public:  
 MyBase32(int x, int y, int z)  
 {  
 std::cout << "…BaseClass32 Object is created!" << std::endl;  
 std::cout << a << " " << b << " " << c << std::endl;  
 a = x, b = y, c = z;  
 std::cout << a << " " << b << " " << c << std::endl;  
 }  
 ~MyBase32()  
 {  
 std::cout << "…BaseClass32 Object is destroyed!" << std::endl;  
 }  
};  
int main()  
{  
 MyBase31 a(1, 2, 3);  
 MyBase32 b(4, 5, 6);  
}



**main2.cpp**(实验手册程序有误，作少量改动)

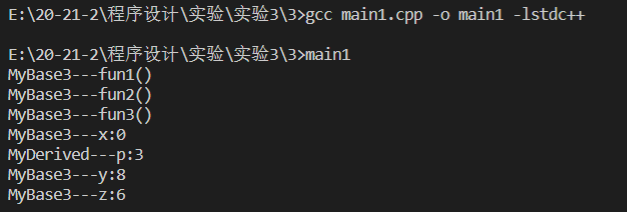
#include <iostream>  
class MyBase31  
{  
 int a, b, c;  
  
public:  
 MyBase31(int x, int y, int z) : a(x), b(y), c(z)  
 {  
 std::cout << "…BaseClass31 Object is created!" << std::endl;  
 std::cout << a << " " << b << " " << c << std::endl;  
 }  
 ~MyBase31()  
 {  
 std::cout << "…BaseClass31 Object is destroyed!" << std::endl;  
 }  
};  
class MyBase32  
{  
 int a, b, c;  
  
public:  
 MyBase32(int x, int y, int z)  
 {  
 std::cout << "…BaseClass32 Object is created!" << std::endl;  
 std::cout << a << " " << b << " " << c << std::endl;  
 a = x, b = y, c = z;  
 std::cout << a << " " << b << " " << c << std::endl;  
 }  
 ~MyBase32()  
 {  
 std::cout << "…BaseClass32 Object is destroyed!" << std::endl;  
 }  
};  
class MyDerived1 : public MyBase31  
{  
 MyBase31 a = MyBase31(5, 6, 7);  
 int c;  
  
public:  
 MyDerived1(int x) : c(x), MyBase31(x, 8, 9)  
 {  
 std::cout << "…Base Object has been created!" << std::endl;  
 std::cout << "…Member Object has been created! " << std::endl;  
 std::cout << "…Derived Object is created! " << c << std::endl;  
 }  
};  
int main()  
{  
 MyDerived1 b(88);  
}



# EX3

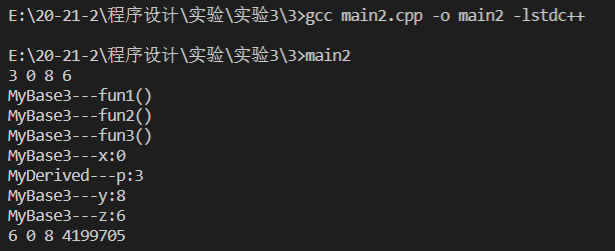
**main1.cpp**

#include <iostream>  
using std::cout;  
using std::endl;  
  
class MyBase3  
{  
  
protected:  
 int y;  
 void fun2()  
 {  
 cout << "MyBase3---fun2()" << endl;  
 }  
  
public:  
 int x;  
 void fun1()  
 {  
 cout << "MyBase3---fun1()" << endl;  
 }  
 int z;  
 void MyBase(int a, int b, int c)  
 {  
 x = a;  
 y = b;  
 z = c;  
 }  
  
 int getX()  
 {  
 cout << "MyBase3---x:";  
 return x;  
 }  
  
 int getY()  
 {  
 cout << "MyBase3---y:";  
 return y;  
 }  
  
 int getZ()  
 {  
 cout << "MyBase3---z:";  
 return z;  
 }  
  
 void fun3()  
 {  
 cout << "MyBase3---fun3()" << endl;  
 }  
};  
  
class MyDerived1 : public MyBase3  
{  
public:  
 int p;  
 MyDerived1(int a) : p(a) {}  
  
 int getP()  
 {  
 cout << "MyDerived---p:";  
 return p;  
 }  
  
 void display()  
 {  
 //cout << p << "" << x << "" << y << "" << z << "" << endl;  
 fun1();  
 fun2();  
 fun3();  
 }  
};  
  
int main()  
{  
 MyDerived1 a(3);  
 a.display();  
 //cout << a.x << "" << a.p << "" << a.y << "" << a.z << endl;  
 cout << a.getX() << "\n"  
 << a.getP() << "\n"  
 << a.getY() << "\n"  
 << a.getZ() << endl;  
}



**main2.cpp**

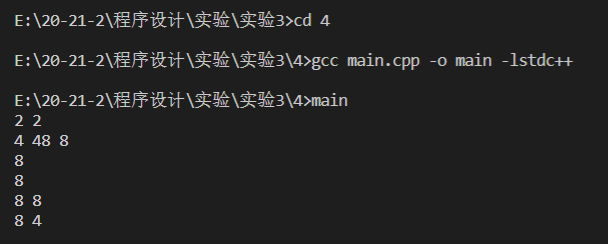
#include <iostream>  
using std::cout;  
using std::endl;  
  
class MyBase3  
{  
  
protected:  
 int y;  
 void fun2()  
 {  
 cout << "MyBase3---fun2()" << endl;  
 }  
  
public:  
 int x;  
 void fun1()  
 {  
 cout << "MyBase3---fun1()" << endl;  
 }  
 int z;  
 void MyBase(int a, int b, int c)  
 {  
 x = a;  
 y = b;  
 z = c;  
 }  
  
 int getX()  
 {  
 cout << "MyBase3---x:";  
 return x;  
 }  
  
 int getY()  
 {  
 cout << "MyBase3---y:";  
 return y;  
 }  
  
 int getZ()  
 {  
 cout << "MyBase3---z:";  
 return z;  
 }  
  
 void fun3()  
 {  
 cout << "MyBase3---fun3()" << endl;  
 }  
};  
  
class MyDerived2 : public MyBase3  
{  
public:  
 int p;  
 MyDerived2(int a) : p(a) {}  
 int getP()  
 {  
 cout << "MyDerived---p:";  
 return p;  
 }  
  
 void display()  
 {  
 cout << p << " " << x << " " << y << " " << z << " " << endl;  
 fun1();  
 fun2();  
 fun3();  
 }  
};  
  
class MyDerived21 : public MyBase3  
{  
public:  
 int p;  
 MyDerived21(int a) : p(a) {}  
 int getP()  
 {  
 cout << "MyDerived21---p:" << endl;  
 return p;  
 }  
 void display1()  
 {  
 cout << p << " " << x << " " << y << " " << z << " " << endl;  
 }  
};  
  
int main()  
{  
 MyDerived2 a(3);  
 MyDerived21 b(6);  
 a.display();  
 //cout << a.x << " " << a.p << " " << a.y << " " << a.z << endl;  
 cout << a.getX() << "\n"  
 << a.getP() << "\n"  
 << a.getY() << "\n"  
 << a.getZ() << endl;  
 b.display1();  
}



# EX4

**main.cpp**

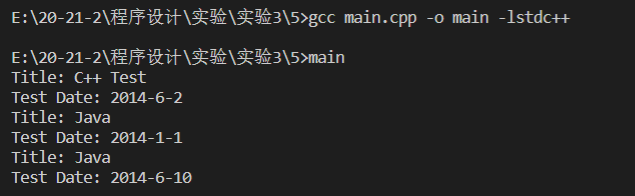
#include <iostream>  
using std::cout;  
using std::endl;  
  
class MyBase  
{  
 int x;  
  
public:  
 MyBase(int a) : x(a){}  
 int getX()  
 {  
 return x;  
 }  
};  
  
class MyDerived : public MyBase  
{  
 int y;  
  
public:  
 MyDerived(int a) : y(a), MyBase(a + 4){}  
 int getY()  
 {  
 return y;  
 }  
};  
  
int main()  
{  
 MyBase a(2), \*p = &a;  
 MyDerived b(4), \*q = &b;  
 MyBase &c = a;  
 MyBase &d = b;  
 cout << a.getX() << " " << p->getX() << endl;  
 cout << b.getY() << " " << q->getY() << b.getX() << " " << q->getX() << endl;  
 a = b;  
 cout << a.getX() << " " << endl; //<< a.getY()  
 p = q;  
 cout << p->getX() << " " << endl; //<< p->getY()  
 cout << c.getX() << " " << d.getX() << " " << endl; //<< d.getY()  
 //b = a;  
 cout << b.getX() << " " << b.getY() << endl;  
}



# EX5

**main.cpp**

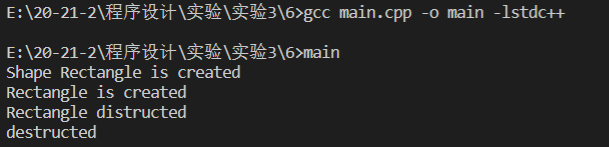
#include <iostream>  
using std::cout;  
using std::endl;  
using std::ostream;  
using std::string;  
  
class Date  
{  
public:  
 Date(int iyear = 2014, int imonth = 1, int iday = 1) : year(iyear), month(imonth), day(iday) {}  
  
 friend ostream &operator<<(ostream &output, Date k)  
 {  
 output << k.year << "-" << k.month << "-" << k.day;  
 return output;  
 }  
  
private:  
 int year;  
 int month;  
 int day;  
};  
  
class FinalTest : public Date  
{  
public:  
 FinalTest(string ititle = "", Date idate = Date()) : title(ititle), date(idate) {}  
  
 void print()  
 {  
 cout << "Title: " << title << endl;  
 cout << "Test Date: " << date << endl;  
 }  
  
 void setDue(Date k)  
 {  
 date = k;  
 }  
  
private:  
 string title;  
 Date date;  
};  
  
int main()  
{  
 FinalTest item1("C++ Test", Date(2014, 6, 2));  
 item1.print();  
 FinalTest item2("Java");  
 item2.print();  
 item2.setDue(Date(2014, 6, 10));  
 item2.print();  
}



# EX6

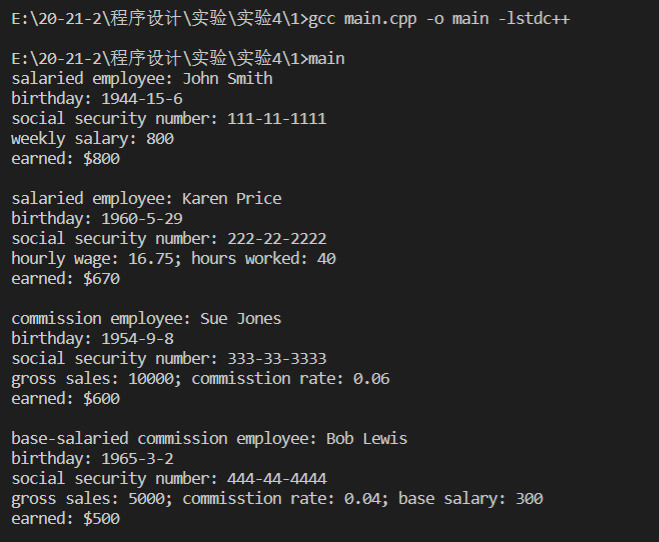
**main.cpp**

#include <iostream>  
#include <string>  
using std::cout;  
using std::endl;  
using std::string;  
  
class Shape  
{  
 public:  
 Shape(string input){  
 id = input;  
 cout << "Shape " << id << " is created" << endl;  
 }  
 ~Shape(){  
 cout << "destructed" << endl;  
 }  
 protected:  
 string id;  
};  
  
class Rectangle: public Shape  
{  
 public:  
 Rectangle(double input\_length, double input\_width= 0,string id = "Rectangle"):length(input\_length),width(input\_width),Shape(id){  
 cout << "Rectangle is created" << endl;  
 }  
 ~Rectangle(){  
 cout << "Rectangle distructed" << endl;  
 }  
 int area(){  
 return length\*width;  
 };  
 protected:  
 int length;  
 int width;  
};  
  
class circle: public Shape  
{  
 public:  
 int area(){  
 return radius\*radius\*3.14;  
 }  
 void print(){  
  
 }  
 protected:  
 int radius;  
};  
  
class square: public Rectangle, public circle  
{  
 protected:  
 circle incribe\_circle;  
};  
  
int main(){  
 Rectangle a(1,2);  
}



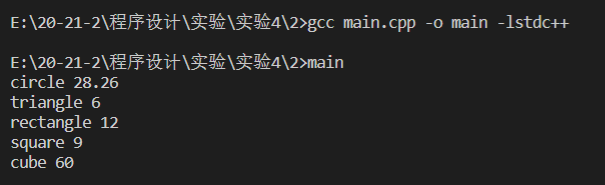
# EX1

#include <iostream>  
#include <windows.h>  
using std::cout;  
using std::endl;  
using std::ostream;  
using std::string;  
  
bool bonus(int month)  
{  
 SYSTEMTIME systm;  
 GetLocalTime(&systm);  
 return (systm.wMonth == month);  
}  
  
class Date  
{  
public:  
 Date(int iyear = 2014, int imonth = 1, int iday = 1) : year(iyear), month(imonth), day(iday) {}  
  
 friend ostream &operator<<(ostream &output, Date k)  
 {  
 output << k.year << "-" << k.month << "-" << k.day;  
 return output;  
 }  
  
 int getMonth()  
 {  
 return month;  
 }  
  
protected:  
 int year;  
 int month;  
 int day;  
};  
  
class Employee  
{  
public:  
 Employee(string iname = "John Smith", string isecuritynumber = "111-11-1111", Date ibirthdate = Date()) : name(iname), securitynumber(isecuritynumber), birthdate(ibirthdate) {}  
  
 virtual void show()  
 {  
 cout << " employee: " << name << endl;  
 cout << "birthday: " << birthdate << endl;  
 cout << "social security number: " << securitynumber << endl;  
 }  
  
protected:  
 string name;  
 Date birthdate;  
 string securitynumber;  
};  
  
class salariedEmployee : public Employee  
{  
public:  
 salariedEmployee(string iname = "John Smith", string isecuritynumber = "111-11-1111", Date ibirthdate = Date(), double isalary = 100) : salary(isalary)  
 {  
 name = iname;  
 securitynumber = isecuritynumber;  
 birthdate = ibirthdate;  
 }  
  
 ~salariedEmployee()  
 {  
 cout << "deleting object of class SalariedEmployee" << endl;  
 }  
  
 void show()  
 {  
 cout << id;  
 Employee::show();  
 cout << "weekly salary: " << salary << endl;  
 cout << (bonus(birthdate.getMonth()) ? "HAPPY BIRTHDAY!\n" : "");  
 cout << "earned: $" << (bonus(birthdate.getMonth()) ? salary + 100 : salary) << endl  
 << endl;  
 }  
  
protected:  
 double salary;  
 string id = "salaried";  
};  
  
class hourlyemployee : public Employee  
{  
public:  
 hourlyemployee(string iname = "John Smith", string isecuritynumber = "111-11-1111", Date ibirthdate = Date(), double iwage = 20, double ihour = 0) : wage(iwage), hour(ihour)  
 {  
 name = iname;  
 securitynumber = isecuritynumber;  
 birthdate = ibirthdate;  
 }  
  
 ~hourlyemployee()  
 {  
 cout << "deleting object of class HourlyEmployee" << endl;  
 }  
 void show()  
 {  
 cout << id;  
 Employee::show();  
 cout << "hourly wage: " << wage << "; ";  
 cout << "hours worked: " << hour << endl;  
 cout << (bonus(birthdate.getMonth()) ? "HAPPY BIRTHDAY!\n" : "");  
 cout << "earned: $" << (bonus(birthdate.getMonth()) ? hour \* wage + 100 : hour \* wage) << endl  
 << endl;  
 }  
  
protected:  
 double wage;  
 double hour;  
 string id = "salaried";  
};  
  
class commissionemployee : public Employee  
{  
public:  
 commissionemployee(string iname = "John Smith", string isecuritynumber = "111-11-1111", Date ibirthdate = Date(), double isale = 20, double icommission = 0) : sale(isale), commission(icommission)  
 {  
 name = iname;  
 securitynumber = isecuritynumber;  
 birthdate = ibirthdate;  
 }  
  
 ~commissionemployee()  
 {  
 cout << "deleting object of class CommissionEmployee" << endl;  
 }  
  
 void show()  
 {  
 cout << id;  
 Employee::show();  
 cout << "gross sales: " << sale << "; ";  
 cout << "commisstion rate: " << commission << endl;  
 cout << (bonus(birthdate.getMonth()) ? "HAPPY BIRTHDAY!\n" : "");  
 cout << "earned: $" << (bonus(birthdate.getMonth()) ? sale \* commission + 100 : sale \* commission) << endl  
 << endl;  
 }  
  
protected:  
 double sale;  
 double commission;  
 string id = "commission";  
};  
  
class basecommissionemployee : public Employee  
{  
public:  
 basecommissionemployee(string iname = "John Smith", string isecuritynumber = "111-11-1111", Date ibirthdate = Date(), double isale = 20, double icommission = 0, double ibase = 0) : sale(isale), commission(icommission), base(ibase)  
 {  
 name = iname;  
 securitynumber = isecuritynumber;  
 birthdate = ibirthdate;  
 }  
  
 void show()  
 {  
 cout << id;  
 Employee::show();  
 cout << "gross sales: " << sale << "; ";  
 cout << "commisstion rate: " << commission << "; ";  
 cout << "base salary: " << base << endl;  
 cout << (bonus(birthdate.getMonth()) ? "HAPPY BIRTHDAY!\n" : "");  
 cout << "earned: $" << (bonus(birthdate.getMonth()) ? sale \* commission + base + 100 : sale \* commission + base) << endl  
 << endl;  
 }  
  
protected:  
 double sale;  
 double commission;  
 double base;  
 string id = "base-salaried commission";  
};  
  
int main()  
{  
 Employee \*employees[4];  
 Date a(1944, 15, 6), b(1960, 5, 29), c(1954, 9, 8), d(1965, 3, 2);  
 employees[0] = new salariedEmployee("John Smith", "111-11-1111", a, 800);  
 employees[1] = new hourlyemployee("Karen Price", "222-22-2222", b, 16.75, 40);  
 employees[2] = new commissionemployee("Sue Jones", "333-33-3333", c, 10000, 0.06);  
 employees[3] = new basecommissionemployee("Bob Lewis", "444-44-4444", d, 5000, 0.04, 300);  
 for (int i = 0; i < 4; i++)  
 {  
 employees[i]->show();  
 }  
}



# EX2

#include <cmath>  
#include <iostream>  
using std::cout;  
using std::endl;  
using std::ostream;  
using std::string;  
  
class shape  
{  
public:  
 shape(){};  
 ~shape(){};  
  
 double x;  
 double y;  
 string name;  
 virtual void print() {}  
};  
  
class circle : public shape  
{  
public:  
 circle(string n = "none", double ix = 0, double iy = 0, double r = 0) : radius(r)  
 {  
 x = ix;  
 y = iy;  
 name = n;  
 }  
  
 ~circle() {}  
  
 double area()  
 {  
 return 3.14 \* radius \* radius;  
 }  
  
 void print()  
 {  
 cout << name << " " << area() << endl;  
 }  
  
protected:  
 double radius;  
};  
  
class triangle : public shape  
{  
public:  
 triangle(string n = "noname", double ix = 0, double iy = 0, double il1 = 0, double il2 = 0, double il3 = 0) : l1(il1), l2(il2), l3(il3)  
 {  
 x = ix;  
 y = iy;  
 name = n;  
 }  
  
 double area()  
 {  
 double p = (l1 + l2 + l3) / 2;  
 return sqrt(p \* (p - l1) \* (p - l2) \* (p - l3));  
 }  
  
 ~triangle() {}  
  
 void print()  
 {  
 cout << name << " " << area() << endl;  
 }  
  
private:  
 double l1;  
 double l2;  
 double l3;  
};  
  
class rectangle : public shape  
{  
public:  
 rectangle(string n = "none", double ix = 0, double iy = 0, double il1 = 0, double il2 = 0) : l1(il1), l2(il2)  
 {  
 x = ix;  
 y = iy;  
 name = n;  
 }  
  
 double area()  
 {  
 return l1 \* l2;  
 }  
  
 ~rectangle() {}  
  
 void print()  
 {  
 cout << name << " " << area() << endl;  
 }  
  
protected:  
 double l1;  
 double l2;  
};  
  
class cube : public shape  
{  
public:  
 cube(string iname = "none", double x = 0, double y = 0, double z = 0)  
 {  
 name = iname;  
 volume = x \* y \* z;  
 }  
  
 ~cube() {}  
  
 void print()  
 {  
 cout << name << " " << volume << endl;  
 }  
  
protected:  
 double volume;  
};  
  
class square : public rectangle  
{  
public:  
 square(string iname = "none", double ix = 0, double iy = 0, double il1 = 0, double il2 = 0)  
 {  
 x = ix;  
 y = iy;  
 name = iname;  
 l1 = il1;  
 l2 = il2;  
 }  
  
 ~square() {}  
  
 double area()  
 {  
 return rectangle::area();  
 }  
  
 void print()  
 {  
 rectangle::print();  
 }  
};  
  
int main()  
{  
 shape \*shapes[5];  
 shapes[0] = new circle("circle", 1, 1, 3);  
 shapes[1] = new triangle("triangle", 1, 1, 3, 4, 5);  
 shapes[2] = new rectangle("rectangle", 1, 1, 3, 4);  
 shapes[3] = new square("square", 1, 1, 3, 3);  
 shapes[4] = new cube("cube", 3, 4, 5);  
 for (int i = 0; i < 5; i++)  
 {  
 shapes[i]->print();  
 }  
}



# EX3

#include <cmath>  
#include <iostream>  
using std::cout;  
using std::endl;  
using std::ostream;  
using std::string;  
  
class Account  
{  
public:  
 virtual void calculate(){};  
  
 void debit(double idebit)  
 {  
 money -= idebit;  
 cout << "Name:" << name << endl;  
 cout << "Money:" << money << endl  
 << endl;  
 }  
  
 void credit(double icredit)  
 {  
 money += icredit;  
 cout << "Name:" << name << endl;  
 cout << "Money:" << money << endl  
 << endl;  
 }  
  
protected:  
 double money;  
 string id;  
 string name;  
};  
  
class SaveingAccount : public Account  
{  
public:  
 SaveingAccount(string iname, double iinterest = 0, double imoney = 0) : interest(iinterest)  
 {  
 id = "Saveing";  
 money = imoney;  
 name = iname;  
 cout << "Account created" << endl;  
 cout << "Name:" << name << endl;  
 cout << "Money:" << money << endl;  
 cout << "Type:" << id << endl  
 << endl;  
 }  
  
 void calculate()  
 {  
 money \*= (1 + interest);  
 cout << "Name:" << name << endl;  
 cout << "Money:" << money << endl  
 << endl;  
 }  
  
protected:  
 double interest;  
};  
  
class CheckingAccount : public Account  
{  
public:  
 CheckingAccount(string iname, double imoney = 0)  
 {  
 money = imoney;  
 id = "Checking";  
 name = iname;  
 cout << "Account created" << endl;  
 cout << "Name:" << name << endl;  
 cout << "Money:" << money << endl;  
 cout << "Type:" << id << endl  
 << endl;  
 }  
  
 void calculate()  
 {  
 cout << "Name:" << name << endl;  
 cout << "Money:" << money << endl;  
 cout << "Warning:This is a checking account" << endl  
 << endl;  
 }  
};  
  
int main()  
{  
 Account \*accounts[2];  
 accounts[0] = new SaveingAccount("John Smith", 0.04, 300);  
 accounts[1] = new CheckingAccount("Karen Price", 400);  
 accounts[0]->debit(20);  
 accounts[1]->debit(40);  
 accounts[0]->credit(20);  
 accounts[1]->credit(40);  
 accounts[0]->calculate();  
 accounts[1]->calculate();  
}

