#pragma once

/\*class CRectangle

{

int \*width, \*height;

public:

CRectangle(int, int);

~CRectangle();

int area()

{

return (\*width \* \*height);

}

};

// Base Class Shape

class Shape

{

public:

void setWidth(int w)

{

width = w;

}

void setHeight(int h)

{

height = h;

}

protected:

int width;

int height;

};

// Base class PaintCost

class PaintCost

{

public:

int getCost(int area)

{

return area \* 70;

}

};

// Derived class

class Rectangle : public Shape, public PaintCost

{

public:

int getArea()

{

return (width \* height);

}

};

// Specifications file for the Rectangle class

#ifndef RECTANGLE\_H

#define RECTANGLE\_H

class Rectangle

{

private:

double width; // the rectangle's width

double length; // the rectangle's length

public:

// Exception class

class NegativeSize

{}; // Empty class declaration

// Default constructor

Rectangle()

{

width = 0.0; length = 0.0;

}

// Mutator functions, defined in Rectangle.cpp

void setWidth(double);

void setLength(double);

// Accessor functions

double getWidth() const

{

return width;

}

double getLength() const

{

return length;

}

double getArea() const

{

return width \* length;

}

};

#endif // !RECTANGLE\_H\*/

// Virtual members

#include <iostream>

using namespace std;

class Polygon

{

protected:

int width, height;

public:

void set\_values(int a, int b)

{

width = a; height = b;

}

virtual int area()

{

return 0;

}

};

class Rectangle : public Polygon

{

public:

int area()

{

return width \* height;

}

};

class Triangle : public Polygon

{

public:

int area()

{

return (width \* height / 2);

}

};

/\*#include <iostream>

#include "Header.h"

using namespace std;

int main()

{

CRectangle rect(3, 4), rectb(5, 6);

cout << "rect area: " << rect.area() << endl;

cout << "rectb area: " << rectb.area() << endl;

return 0;

}

CRectangle::CRectangle(int a, int b)

{

width = new int;

height = new int;

\*width = a;

\*height = b;

}

CRectangle::~CRectangle()

{

delete width;

delete height;

}\*/

/\*#include <iostream>

#include "Header.h"

using namespace std;

int main()

{

Rectangle Rect;

int area;

Rect.setWidth(5);

Rect.setHeight(7);

area = Rect.getArea();

// Print the area of the object

cout << "Total area: " << Rect.getArea() << endl;

// Print the total cost of painting

cout << "Total paint cost: $" << Rect.getCost(area) << endl;

return 0;

}

#include <iostream>

using namespace std;

// Function prototype

double divide(int, int);

int main()

{

int num1, num2; // To hold two numbers

double quotient; // To hold the quotient of the numbers

// Get two numbers

cout << "Enter two numbers: ";

cin >> num1 >> num2;

// Divide num1 by num2 and catch any potential exceptions

try

{

quotient = divide(num1, num2);

cout << "The quotient is " << quotient << endl;

}

catch (char \*exceptionString)

{

cout << exceptionString;

}

cout << "End of the program.\n";

return 0;

}

// The divide function divides numerator by denominator. If denominator is zero,

// the function throws an exception.

double divide(int numerator, int denominator)

{

if (denominator == 0)

{

throw "ERROR: Cannot divide by zero.\n";

}

return static\_cast<double>(numerator) / denominator;

}

// Implementation file for the Rectangle class

#include "Header.h"

#include <iostream>

using namespace std;

// setWidth set the value of the member variable width

void Rectangle::setWidth(double w)

{

if (w >= 0)

width = w;

else

throw NegativeSize();

}

// setLength set the value of the member variable length

void Rectangle::setLength(double len)

{

if (len >= 0)

length = len;

else

throw NegativeSize();

}

// This program demonstrates Rectangle class exceptions

int main()

{

int width;

int length;

// Create a Rectangle object

Rectangle myRectangle;

// Get the width and length

cout << "Enter the rectangle's width: ";

cin >> width;

cout << "Enter the rectangle's length: ";

cin >> length;

// Store these values in the Rectangle object

try

{

myRectangle.setWidth(width);

myRectangle.setLength(length);

cout << "The area of the rectangle is "

<< myRectangle.getArea() << endl;

}

catch (Rectangle::NegativeSize)

{

cout << "Error: A negative value was entered.\n";

}

cout << "End of the program.\n";

return 0;

}\*/

#include <iostream>

#include "Header.h"

using namespace std;

int main()

{

Rectangle rect;

Triangle trgl;

Polygon poly;

Polygon \* ppoly1 = &rect;

Polygon \* ppoly2 = &trgl;

Polygon \* ppoly3 = &poly;

ppoly1->set\_values(4, 5);

ppoly2->set\_values(4, 5);

ppoly3->set\_values(4, 5);

cout << ppoly1->area() << '\n';

cout << ppoly2->area() << '\n';

cout << ppoly3->area() << '\n';

return 0;

}

#pragma once

class Account {

protected:

int deposit, withdrawl;

double interest, charge, bal;

public:

void setAmmt(double a, double b, double c)

{

bal = a;

interest = b;

charge = c;

}

void setVals(int a, int b)

{

deposit = a;

withdrawl = b;

}

virtual double bankDeposit(double ammt)

{

bal = bal + ammt;

deposit++;

return bal;

}

virtual double bankWithdrawl(double ammt)

{

bal = bal - ammt;

withdrawl++;

return bal;

}

virtual void calcInt()

{

double MIR = interest / 12;

double MI = bal \* MIR;

bal = bal + MI;

}

virtual double monthlyProc()

{

bal = bal - charge;

calcInt();

setVals(0, 0);

return bal;

};

};

class Savings: public Account {

private:

bool active;

public:

void status()

{

if (bal < 25)

active = false;

}

double saveDeposit(double ammt)

{

if (active)

{

bal = bal + ammt;

deposit++;

return bal;

}

}

double saveWithdrawl(double ammt)

{

if (active)

{

bal = bal - ammt;

withdrawl++;

return bal;

}

}

double monthlyProc()

{

if (active)

{

bal = bal - charge;

calcInt();

return bal;

}

}

};

class Checking: public Account {

double checkWithdrawl(double ammt)

{

if (bal <= 0) {

bal = bal - 15.00;

}

else {

bal = bal - ammt;

withdrawl++;

return bal;

}

}

double monthlyProc()

{

charge = 5.00 \* 12 + (.10\*withdrawl);

bal = bal - charge;

calcInt();

return bal;

}

};

int main()

{

Account acct;

Savings save;

Checking check;

Account \* acct1 = &acct;

double balance = 5000.00; // balance of the savings account

double charge = 10.00; // new balance of the savings account

double interest = 12.5;

int dep = 5, with = 5;

acct1->setAmmt(balance, interest, charge);

cout << "Enter number of deposits, then withdrawls" << endl;

cin >> dep >> with;

acct1->setVals(dep, with);

cout << "You have $" << balance << endl;

cout << "You have deposited " << dep << " times. \nYou've withdrawn " << with << " times" << endl;

cout << "Your total service charges are $" << charge << endl;

cout << "You now have $" << acct1->bankDeposit(100.00) << endl;

cout << acct1->monthlyProc() << endl;

return 0;

}