Sinai University

Faculty of Information Technology

Computer Programming (2) **Quiz #2**

16-11-2019 **Time allowed: One Hour**

Model Answer

Answer the following Questions [15 marks]

Question 1:

Show the output of the following program:

```
class Circle {
   protected:
      int x, y, r;
   public:
     Circle(int xVal, int yVal, int rVal)
     { x=xVal;  y=yVal; r=rVal;}
     void display()
     {cout << x << "," << y << "," << r; }
     virtual float getArea()
      { return 3.14 * r * r; }
};
class Cylinder  : public Circle {
   private:
     int height;
   public:
     Cylinder(int xVal, int yVal, int rVal, int h)
                    : Circle(xVal, yVal, rVal)
     { height = h; }
     void display()
     {  cout << x << ","  << y << ","
           << r << "," << height; }
     float getArea()
     { return 2 * 3.14 * r * height; }
};
void Show(Circle* shape)
{ cout << typeid(*shape).name() << ": "; }

void main () {
  Circle* shape[4];
  shape[0]=new Circle(3, 5, 10);
  shape[1]=new Cylinder(6, 8, 5, 20);
  shape[2]=new Circle(7, 9, 15);
  shape[3]=new Cylinder(1, 3, 7, 5);
  for (int i=0; i<4;i++)
  {
    Show(shape[i]);
    shape[i]->display();
    cout<<"\nArea:" << shape[i] -> getArea();
  }
}
```

3, 5,10
Area:314
6, 8,5
Area:628
7, 9,15
Area:706.5
1, 3,7
Area:219.8

Question 2:

Declare and implement the following class hierarchy that includes 4 classes, each one has a **constructor** that creates an object of the class and initializes its data members with given values:

A) A base class ***account*** that represents a bank account and has one data member **balance** that represents the current balance, and 3 member functions: **deposit()** that accepts an amount of money, then increases the balance by this amount, **get_balance()** that displays the balance, and **withdraw()** that accepts an amount of money, then decreases the balance by this amount if it does not exceed the balance and returns the amount withdrawn; otherwise, returns 0.

B) A child class ***curnt_acnt*** that represents a current account and has 2 additional data members: **limit** representing a lower limit for check cashing free of charge, and **charge** representing charge per check for lower balance, and redefines the member function **withdraw()** such that it accepts an amount of money **amt**, then decreases the balance by **amt**+**charge**, if the balance is less than **limit** and the value of **amt**+**charge** does not exceed the balance, and returns the amount withdrawn, otherwise it performs the withdrawal as in class ***account***.

C) A child class ***sav_acnt*** that represents a savings account and has an additional data member: **intrstRate** representing annual interest rate, and a member function: **compute_int()** that computes the interest, adds it to the balance and returns the interest, where interest = balance x rate.

D) A grandchild class ***special_acnt***, which represents a special account that combines the properties of the current account and the savings account, and has no data members of its own. It redefines the member function **withdraw ()** to perform the withdrawal as in class ***curnt_acnt***, and redefines **compute_int()** to compute the interest, add it to the balance and return the interest, only if the balance exceeds **limit**, otherwise it returns 0.

E) write a main program that performs the following tasks:

(i) Reads the information of accounts from the keyboard, and create ***special_acnt*** objects for these accounts, then

(ii) Asks the user to enter one of the following letters and performs the corresponding operation:

D: Asks the user to enter an amount of money, and deposits this amount in the specified account, then displays the account balance.

W: Asks the user to enter an amount of money, and attempts to withdraw this amount from the specified account, then displays a message indicating whether the withdrawal is done or not.

C: Computes the interest for the account, then displays the account balance.

```
#include <iostream>
#include <cmath>
using namespace std;
class account
{
protected:
	double balance;
public:
	account(double b)
	{
		balance = b;
	}
	void deposit(double amt)
	{
		balance = balance + amt;
	}
	void getbalance()
	{
		cout << balance;
	}
	double withdraw(double amt)
	{
		if (amt <= balance)
		{
			balance = balance - amt;
			return amt;
		}
		else
			return 0.0;
	}
};
class curnt_acnt :virtual public account
{
protected:
	double limit, charge;
public:
	curnt_acnt(double b, double lim, double ch) :account(b)
	{
		limit = lim;
		charge = ch;
	}
	double withdraw(double amt)
	{
		if (balance<limit && amt + charge>balance)
		{
			balance = balance - (amt + charge);
			return amt;
		}
		else
			return account::withdraw(amt);
	}
};
```

```
class sav_acnt :virtual public account
{
protected:
	double interstRate;
public:
	sav_acnt(double b, double rate) :account(b)
	{
		interstRate = rate;
	}
	double compute_int()
	{
		double interest = balance * interstRate;
		balance = balance + interest;
		return interest;
	}
};
class special_acnt : public curnt_acnt, public sav_acnt
{
public:
	special_acnt(double b, double lim, double ch, double rate)
		:account(b),curnt_acnt(b, lim, ch), sav_acnt(b, rate)
	{}
	double withdraw(double amt)
	{
		return curnt_acnt::withdraw(amt);
	}
	double compute_int()
	{
		if (balance > limit)
		{
			return sav_acnt::compute_int();
		}
		else
			return 0.0;
	}
};
```

```
int main()
{
	double b, ch, lim, rate,amt;
	char choice;
	cin >> b >> ch >> lim >> rate;
	special_acnt s1(b, lim, ch, rate);
	cout << "enter your choice (D,W,C)";
	cin >> choice;
	if (choice == 'D')
	{
		cout << "enter the amount of money";
		cin >> amt;
		s1.deposit(amt);
		s1.getbalance();
	}
	else if (choice == 'W')
	{
		cout << "enter the amount of money";
		cin >> amt;
		double w = s1.withdraw(amt);
		if (w == 0)
			cout << "the operation failed";
		else
			cout << "the operation succesful";
	}
	else if (choice == 'C')
	{
		double out=s1.compute_int();
		s1.getbalance();

	}
}
```

With my best wishes