In this exercise you will write a program which estimates the cost of a telephone call, and also implement unit tests for the program.

The phone company applies the following rules to a phone call to calculate the charge:

* The minimum before-tax charge of 59.400 cents applies to all calls to any destination up to 50km away and 89.000 cents for any destination further than 50km away.
* Calls are charged on a per-second basis at 0.759 cents per second (<= 50km) and 1.761 cents per second (> 50km)
* Off-peak seconds (from 19:00:00 to 06:59:59 the next day) are given a discount of 40% off (<= 50km) and 50% off (> 50km) off the above rate
* If the type of call was share-call AND the destination is more than 50km away, there is a discount of 50% off after any off-peak discount (minimum charge still applies). However, share-calls over shorter distances are not discounted.
* Finally, VAT of 14% is added to give the final cost.

Your program should ask for the following input:

* The starting time of the call (to be split up into hours, minutes and seconds)
* The duration of the call (to be split up into minutes and seconds)
* Whether the duration was more than 50km away
* Whether the call was share-call

Hint: you can prompt the user to input hours, minutes and seconds at once by asking for a format likeHH:MM:SS and splitting the resulting string by the delimiter. You may assume that the user will enter valid input, and that no call will exceed 59 minutes and 59 seconds.

Your program should output the following information:

* The basic cost
* The off-peak discount
* The share-call discount
* The net cost
* The VAT
* The total cost

1. Before you write the program, identify the equivalence classes and boundaries that you will need to use in equivalence testing and boundary analysis when writing black-box tests.
2. Write the program. Remember that you will need to write unit tests for this program – the calculation that you need to test should be placed in some kind of unit, like a function, which can be imported from outside of the program and used independently of the rest of the code (like the user input)!

Now implement the black-box tests by writing a unit test module for your program. Run all the tests, and make sure that they pass!

#include "stdafx.h"

#include<iostream>

#include<conio.h>

#include<stdlib.h>

#include<string>

#include<ctime>

#include<sstream>

using namespace std;

class phoneBill {

private:

double basicCost=0.0;

double callCharge=0.0;

double VAT = 0.14;

double offPeak=0.0;

double sharedDiscount=0.0;

double netCost=0.0;

double totalCost=0.0;

public:

phoneBill(char distanceCall,char sharedCall) {

if (distanceCall=='y') {

this->basicCost = 89.001;

this->callCharge = 1.761;

if (sharedCall == 'y') {

this->sharedDiscount = 0.5;

}

else

{

this->sharedDiscount = 0.0;

}

}

else

{

this->basicCost = 59.4;

this->callCharge = 0.759;

}

}

double getBasicCost() {

return this->basicCost;

}

double getCallCharge() {

return this->callCharge;

}

double getOffPeak() {

return this->offPeak;

}

double getSharedDiscount() {

return this->sharedDiscount;

}

double getNetCost() {

return this->netCost;

}

double getTotalCost() {

return this->totalCost;

}

double getVAT() {

return this->VAT;

}

int getHours(string time) {

int hours;

stringstream greek(time.substr(0, 2));

greek >> hours;

if (hours < 24) {

return hours;

}

else

{

throw string("Invalid Hour Exception");

}

}

int getMin(string time) {

int min;

stringstream greek(time.substr(3, 2));

greek >> min;

if (min < 60) {

return min;

}

else {

throw string("Invalid Minute Exception");

}

}

int getSec(string time) {

int sec;

stringstream greek(time.substr(6, 2));

greek >> sec;

if (sec < 60) {

return sec;

}

else

{

throw string("Invalid Second Exception");

}

}

void calculateOffPeakDiscount(string startingTime, char distanceCall) {

int hours,min,sec;

hours = getHours(startingTime);

min = getMin(startingTime);

sec = getSec(startingTime);

if ((hours >= 19 && hours <= 24) || (hours >= 0 && hours<7)) {

if (distanceCall == 'y') {

this->offPeak = 0.5;

}

else

{

this->offPeak = 0.4;

}

}

}

void calculateNetCost(string durationTime) {

int hours,min,sec;

hours = getHours(durationTime)\*3600;

min = getMin(durationTime)\*60;

sec = getSec(durationTime);

this->netCost = (hours + min + sec)\*this->callCharge;

this->netCost = (this->netCost\*(1 - this->offPeak));

this->netCost = (this->netCost\*(1 - this->sharedDiscount));

if (this->netCost < this->basicCost) {

this->netCost = this->basicCost;

}

}

void calculateTotalCost() {

this->totalCost = this->netCost + (this->netCost\*this->VAT);

}

};

int main()

{

string startingTime, durationTime;

char distanceCall, sharedCall;

cout << "Enter the strating time of the call(HH:MM:SS):\t";

cin >> startingTime;

cout << "Enter the duration of the call(HH:MM:SS):\t";

cin >> durationTime;

cout << "Was the duration was more than 50km(y/n):\t";

cin >> distanceCall;

cout << "Is the call a shared call(y/n):\t\t\t";

cin >> sharedCall;

try {

phoneBill\* phone = new phoneBill(distanceCall, sharedCall);

phone->calculateOffPeakDiscount(startingTime, distanceCall);

phone->calculateNetCost(durationTime);

phone->calculateTotalCost();

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Telephone Bill\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\tBasic Cost\t\t=" << phone->getBasicCost() << endl;

cout << "\tOff-Peak Discount\t=" << (phone->getOffPeak())\*100 << "%" << endl;

cout << "\tShared Call Discount\t=" << (phone->getSharedDiscount())\*100 << "%" << endl;

cout << "\tNet Cost\t\t=" << phone->getNetCost() << endl;

cout << "\tVAT\t\t\t=" << (phone->getVAT())\*100 << "%" << endl;

cout << "\tTotalCost\t\t=" << phone->getTotalCost() << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

}

catch (string e) {

cout << e;

}

cin.clear();

cin.ignore(1000,'\n');

cin.get();

return 0;

}