**תרגיל 4**

**פונקציות חברות, סטטיות וטיפול בחריגות**

בתרגיל זה עלייך לכתוב תכנית לניהול כספומט. בצע את השלבים הבאים, עד לתכנית המלאה.

בכל מצבי השגיאה האפשריים יש לזרוק חריגה מתאימה ולהמשיך לפעולה הבאה.

רשימת ההודעות לחריגה מפורטת בהמשך לגבי כל אחד מחלקי התכנית.

1. הגדר מחלקה בשם Clock לייצוג שעה.

השדות במחלקה יהיו:

* + hour - השעה (0 עד 23)
  + minute - מספר הדקות (0 עד 59)
  + second - מספר השניות (0 עד 59)

הוסף למחלקה את הפונקציות הבאות:

* + constructors:
    1. default -שקובע את השעה 00:00:00.
    2. קונסטרקטור עם פרמטרים - מקבל ערכים ומציב בשדות, במידה והערכים שגויים השעה תיקבע ל- 00:00:00.
    3. copy constructor
  + set/get עבור כל שדה במחלקה
  + אופרטור += שמקבל כפרמטר מספר שניות ומקדם את השעות בהתאם.
  + אופרטור פלט >> להדפסת השעה בפורמט hh:mm:ss (שימו לב שגם כאשר הערכים קטנים מ-10 יש להדפיס בפורמט של 2 ספרות).
  + ואופרטור קלט >> עבור קלט גם כן מהפורמט.

במקרה שהקלט אינו תקין תשלח הודעת חריגה ובאובייקט יוצב הערך 00:00:00.

בכל מקרה של חריגה תשלח הודעה מתאימה לפי הניסוח בפירוט הבא:

Wrong time format.

Invalid time - more than 60 seconds.

Invalid time - more than 60 minutes.

Invalid time - more than 24 hours.

Invalid time - negative number of seconds.

Invalid time - negative number of minutes.

Invalid time - negative number of hours.

1. הגדר מחלקה בשם Account לייצוג חשבון בנק.

השדות במחלקה יהיו:

* accountNumber - מספר החשבון.
* code - קוד סודי בן 4 ספרות (הספרה השמאלית ביותר אינה 0).
* balance – היתרה בחשבון.

הוסף למחלקה את הפונקציות הבאות:

* + constructors:
    1. default -שמציב 0 בכל השדות.
    2. קונסטרקטור עם פרמטרים - מקבל ערכים ומציב אותם בשדות.
  + set/get לפי הצורך (אין לאפשר לשנות את הקוד הסודי ואת מספר החשבון)
  + אופרטור << עבור קלט של נתוני החשבון ההתחלתיים - מספר חשבון וקוד סודי. הלקוח אינו מזין יתרה. היתרה הראשונית הינה 0.
  + withdraw(int nis) – פונקציה למשיכת מזומנים בסכום של nis ש"ח. בפעולה אחת ניתן למשוך מזומנים עד לסכום של 2500 ₪, עם מסגרת אשראי של 6000 ₪ (כלומר תאפשר חריגה של עד 6000 ₪ )
  + deposit(int) – פונקציה המאפשרת להפקיד צ'קים בחשבון בסכום שאינו עולה על 10000 ₪.

בנוסף, המחלקה תכלול את שדות ופונקציות סטטיים:

* sumWithdraw – מונה של סכום כל המשיכות מהבנק (מכל החשבונות).
* sumDeposit – מונה של סכום כל ההפקדות בהמחאה (מכל החשבונות).
  + getSumWithdraw() - פונקציה לברור סכום כל המשיכות מכל החשבונות יחד.
  + getSumDeposit() - פונקציה לברור סכום כל הצ'קים שהופקדו עד עכשיו.

בכל פעם שמתעוררת בעיה יש לזרוק הודעת שגיאה מתאימה.

להלן נוסח ההודעה האפשריות:

ERROR: wrong code!

ERROR: cannot deposit more than 10000 NIS!

ERROR: cannot withdraw more than 2500 NIS!

ERROR: cannot have less than - 6000 NIS!

1. נתונה התוכנית הראשית הבאה למימוש כספומט. התוכנית משתמשת במחלקות שגדרת בשאלות הקודמות, בתוכנית זו מניחים שהבנק כולל 10 חשבונות בלבד.

בשלב ראשון המשתמש נדרש להזין את נתוני כל החשבונות:

* מספר חשבון (המספר חייב להיות ייחודי, כלומר – לא יהיו שני חשבונות עם אותו מספר חשבון),
* וקוד סודי (בן 4 ספרות, הספרה השמאלית אינה 0).

**המשתמש אינו מזין היתרה.** נניח שהיתרה ההתחלתית היא תמיד 0.

החריגות האפשריות בשלב זה

ERROR: code must be of 4 digits!

ERROR: no such account number!

ERROR: account number must be unique!

ולאחר מכן התוכנית מאפשרת בלולאה לבחור בסדרה של פעולות עד לבחירת 0 – stop.

בכל שלב התוכנית אמורה לבצע פעולה אחת ולהדפיס הודעה מתאימה כולל את שעת ביצוע הפעולה. במידה והתקבלה חריגה התוכנית אמורה להדפיס את שעה (לפני ביצוע הפעולה) ואת תוכן החריגה שהתקבלה ותמשיך לבקש את הבחירה הבאה.

נניח כי שעת פתיחת הכספומט היא שמונה בבוקר 08:00:00

בירור יתרה אורך 20 שניות,

משיכת מזומנים אורך 50 שניות,

הפקדת אורכת 30 שניות

הצגת סך ההפקדות או סך כל המשיכות אורכת דקה.

נניח כי כל הפעולות מתבצעות ברצף, ללא הפסקה.

התוכנית עובדת נכון כל עוד אין חריגות...

הוסיפו לתוכנית זו את הטיפול בכל החריגות האפשריות.

#include "Clock.h"

#include "Account.h"

using namespace std;

enum action

{

stop,

balance,

withdraw,

deposit,

sumDeposit,

sumWithdraw

};

action menu()

{

cout<<"enter 1 to get account balance"<<endl;

cout<<"enter 2 to deposit money"<<endl;

cout<<"enter 3 to withdraw money"<<endl;

cout<<"enter 4 to see the sum of all withdraws"<<endl;

cout<<"enter 5 to see the sum of all Deposits"<<endl;

cout<<"enter 0 to stop"<<endl;

int x;

cin>>x;

return (action)x;

}

int findAccount(Account\* bank,int size)

**{**

int number,code;

cout<<"please enter account number: ";

cin>>number;

int i=0;

while (i<size && bank[i].getAccountNumber()!=number)

i++;

cout<<"please enter the code: ";

cin>>code;

if (bank[i].getCode()==code)

return i;

}

void printTransaction(Account a, action ac)

{

Clock c;

cout<<c<<"\t";

;

switch (ac)

{

case balance: cout<<"account #: "<<a.getAccountNumber()<<"\t";

cout<<"balance: "<<a.getBalance()<<endl;

break;

case deposit:

case withdraw: cout<<"account #: "<<a.getAccountNumber()<<"\t";

cout<<"new balance: "<<a.getBalance()<<endl;

break;

case sumDeposit:

cout<<"sum of all deposits: "<<Account::getSumDeposit()<<endl;

break;

case sumWithdraw:

cout<<"sum of all withdraws: "<<Account::getSumWithdraw()<<endl;

break;

}

}

void getBalance(Account\* bank,int size)

{

int i=findAccount(bank,size);

printTransaction(bank[i],balance);

}

void checkDeposit(Account\* bank,int size)

{

int i=findAccount(bank,size);

float amount;

cout<<"enter the amount of the check: ";

cin>>amount;

bank[i].deposit(amount);

printTransaction(bank[i],deposit);

}

void cashWithdraw(Account\* bank,int size)

{

int i=findAccount(bank,size);

float amount;

cout<<"enter the amount of money to withdraw: ";

cin>>amount;

bank[i].withdraw(amount);

printTransaction(bank[i],withdraw);

}

int main()

{

Account bank[10];

cout<<"enter account number and code for 10 accounts:\n";

for(int i=0;i<10;i++)

cin>>bank[i];

action ac=menu();

while(ac)

{

switch (ac)

{

case balance: getBalance(bank,10);

break;

case withdraw:cashWithdraw(bank,10);

break;

case deposit:checkDeposit(bank,10);

break;

case sumDeposit:printTransaction(bank[0],sumDeposit);

break;

case sumWithdraw:printTransaction(bank[0],sumWithdraw);

}

ac=menu();

}

return 0;

}