Programmazione 2
e Laboratorio di Programmazione

Corso di Laurea in

Informatica

Università degli Studi di Napoli "Parthenope"

Anno Accademico 2023-2024

Prof. Luigi Catuogno

1

Esercizi svolti

2

Esercizio: *Simulazione di un C/C bancario #3*

Modifichiamo la classe MiniCCB per aggiungere le seguenti funzionalità:

Un metodo che cambi il PIN

```
bool cambiaPIN(string oldPIN, string newPIN)
```

Il PIN deve essere di almeno 5 caratteri

3

Esercizio: *Simulazione di un C/C bancario #3*

Modifichiamo la classe MiniCCB per aggiungere le seguenti funzionalità:

Un codice PUK che possa essere utilizzato per sbloccare un C/C dopo tre errori di PIN consecutivi (e cambia il PIN).

```
bool sblocca(string mioPUK, string newPIN)
```

Il PUK deve essere di almeno 10 caratteri

10 errori di PUK consecutivi bloccano il C/C

4

Esercizio: *Simulazione di un C/C bancario #3*

```
 4  class MiniCCB {
 5  private:
 6          double dare;
 7          double avere;
 8          int tentativiPIN;
 9          int tentativiPUK;
10          bool blocco;
11          string PIN;
12          string PUK;
 …  …
```

5

Esercizio: *Simulazione di un C/C bancario #3*

```
13          bool CheckPIN(string p)
14          {
 …                  …
26          }
27
28          bool CheckPUK(string p)
29          {
30                  if (tentativiPUK>10)
31                          return false;
32                  if(p==PUK){
33                          tentativiPUK=0;
34                          return true;
35                  }
36                  tentativiPUK++;
37                  return false;
38          }
```

6

Esercizio: *Simulazione di un C/C bancario #3*

```
39  public:
40          MiniCCB() {
41                  dare=avere=0;
42                  tentativiPIN=tentativiPUK=0;
43                  PIN="11111";
44                  PUK="1111111111";
44                  blocco=false;
46          };
47
48          MiniCCB(string p, string q) {
49                  dare=avere=0;
50                  tentativiPIN=tentativiPUK=0;
52                  PIN=p;
52                  PUK=q;
53                  blocco=false;
54          };
```

7

Esercizio: *Simulazione di un C/C bancario #3*

```
55          bool Saldo(double &importo, string p)
56          {
…                   …
62          };
63          bool Deposito(double importo, string p)
64          {
…                   …
70          };
71          bool Prelievo(double importo, string p)
72          {
…                   …
78          };
79          bool Bloccato()
80          {
81                  return blocco;
82          }
```

8

Esercizio: *Simulazione di un C/C bancario #3*

```
83          bool CambiaPIN(string oldp, string newp)
84          {
85                  if(blocco || !CheckPIN(oldp))
86                          return false;
87                  if(newp.size()<5)
88                          return false;
89                  PIN=newp;
90                  return true;
91          }
92
```

9

Esercizio: *Simulazione di un C/C bancario #3*

```
93          bool Sblocca(string pk, string newp)
94          {
95                  if(!CheckPUK(pk))
96                          return false;
97                  if(!blocco||newp.size()<5)
98                          return false;
99                  PIN=newp;
100                 tentativiPIN=0;
101                 blocco=false;
102                 return true;
103         }
104 };
```

10

Esercizio: *Simulazione di un C/C bancario #3*

```
105  int main()
106  {
107          double saldo=0, importo=0;
108          bool ancora=true;
109          int scelta=0;
110          MiniCCB Conto("12345","abcdefghil");
111          string mioPIN,nuovoPIN,mioPUK;
 …           …
```

11

Esercizio: *Simulazione di un C/C bancario #3*

```
164          case 4:
165                  do {
166                          cout << "Inserisci il nuovo PIN (min. 5 caratteri): ";
167                          cin >> nuovoPIN;
168                  } while (nuovoPIN.size()<5);
169                  cout << "Inserisci il PIN: ";
170                  cin >> mioPIN;
171                  if(!Conto.CambiaPIN(mioPIN,nuovoPIN)){
172                          cout<<"L'operazione non è andata a buon fine!" << endl;
173                          cout<<"Verificare che il PIN sia corretto o …"<<endl;
174                  }
175                  break;
```

12

Esercizio: *Simulazione di un C/C bancario #3*

```
176         case 5:
177                 do {
178                         cout << "Inserisci il nuovo PIN (min. 5 caratteri): ";
179                         cin >> nuovoPIN;
180                 } while (nuovoPIN.size()<5);
181                 cout << "Inserisci il PUK: ";
182                 cin >> mioPUK;
183                 if(!Conto.Sblocca(mioPUK,nuovoPIN)){
184                         cout<< "L'operazione non è andata a buon fine!" << endl;
185                         cout << "Verificare che il PUK sia … la banca"<<endl;
186                 }
187                 break;
188         } // end switch
  …      …
```

13

Esercizio: *sugli angoli e i gradi…*

Scrivere una classe **`tellAngle()`** con la seguente interfaccia:

```
tellAngle(int gradi, int primi, int secondi)
```

Il costruttore che imposta i tre attributi indicati. Se i parametri passati risultassero superiori risp. a 360, 60 e 60, imposta gli stessi valori *modulo* la rispettiva soglia.

```
void show()
```

visualizza i tre valori nel formato **`ggg:pp:ss`**

14

Esercizio: *sugli angoli e i gradi…*

```
int getGr();
int getPr();
int getSc();
```

restituisce il valore dell'attributo corrispondente

```
void setGr(int g);
void setPr(int p);
void setSc(int s);
```

imposta il valore dell'attributo corrispondente (modulo risp. 360, 60 e 60)

15

Esercizio: *sugli angoli e i gradi…*

```
10  class tellAngle {
11  private:
12          int gradi;
13          int primi;
14          int secondi;
15  public:
16          tellAngle(int g, int p, int s)
17          {
18                  gradi=g%360;
19                  primi=p%60;
20                  secondi=s%60;
21          }
 …  …
```

16

Esercizio: *sugli angoli e i gradi…*

```
22          void show()
23          {
24                  cout << setw(3)<<setfill('0')<<right<<gradi<<":";
25                  cout << setw(2)<<setfill('0')<<right<<primi<<":";
26                  cout << setw(2)<<setfill('0')<<right<<secondi<<endl;
27          }
28          int getGr() { return gradi; }
29          void setGr(int g) { gradi=g%360; }
30          // etc. …
31  };      // fine def. classe
32  int main()
33  {
34          tellAngle x(12,6,35);
35          x.show();
36  }
```

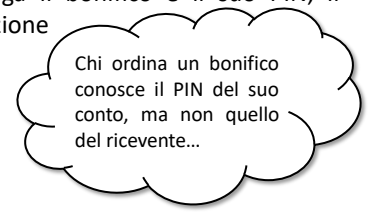
17

Esercizio: *Simulazione di un C/C bancario #4*

Si implementi una funzione esterna alla classe MiniCCB che permetta di effettuare un bonifico da un conto all'altro:

```
bool bonifico( MiniCCB &mioCC,
               string mioPIN,
               MiniCCB &suoCC, double importo )
```

La funzione prende il riferimento al conto che eroga il bonifico e il suo PIN, il riferimento al conto ricevente e l'importo della transazione



18

Esempio: *Simulazione di un C/C bancario #4*

```
10  class MiniCCB {
11          friend bool bonifico(MiniCCB &, string, MiniCCB &, double);
12  private:
13          double dare;
14          double avere;
16          int tentativiPIN;
16          int tentativiPUK;
17          bool blocco;
18          string PIN;
19          string PUK;
20          …
```

bonifico è una funzione *friend* della classe **MiniCCB**.

19

Esempio: *Simulazione di un C/C bancario #4*

```
118 }; // end class MiniCCB
119
120 bool bonifico(MiniCCB &mioCC, string mioPIN, MiniCCB &suoCC, double importo)
121 {
122         if(!mioCC.Prelievo(importo,mioPIN))      {
123                 cout << "L'operazione non è andata       fine!" << endl;
124                 cout << "Verificare che il PIN sia c
125                 return false;
126         }
127         suoCC.avere+=importo;
128         return true;
129 }
130
131 int main()
132 {
  …
136         MiniCCB Conto("12345","abcdefghil"), AltroConto("54321","ciao!ciao!");
```

Per l'addebbito non è necessario implementare nulla di nuovo, basta utilizzare il metodo **Prelievo**

Effettua l'accredito incrementando direttamente l'attributo **importo** (che è privato) di **suoCC**

20

Esempio: *Simulazione di un C/C bancario #4*

```
211         case 6:
212                 cout << "Inserisci l'importo del bonifico: ";
213                 cin >> importo;
214                 cout << "Inserisci il PIN: ";
215                 cin >> mioPIN;
216                 if(!bonifico(Conto,mioPIN,AltroConto,importo)) {
217                         cout<< "L'operazione non è andata a buon fine!" << endl;
218                         cout << "Verificare che il PIN sia corretto …"<<endl;
219                 }
220                 else
221                  {
222                         AltroConto.Saldo(importo,"54321");
223                         cout << "saldo altro conto: "<< importo << endl;
224                 }
225                 break;
  …
…
```

21