Міністерство освіти і науки України
Державний вищий навчальний заклад
«Донецький національний технічний університет»

Н. М. Дацун

Об’єктно-орієнтоване програмування

Навчальний посібник для студентів
спеціальності «Програмна інженерія»

Донецьк
2014

Д21
УДК 004.43(075)

Рецензенти: К.М. Довбня, доктор фізико-математичних наук, професор кафедри прикладної механіки і комп'ютерних технологій Донецького національного університету

О.А. Дмитрієва, доктор технічних наук, професор кафедри прикладної математики та інформатики Донецького національного технічного університету

Рекомендовано Вченою радою Донецького національного технічного університету як навчальний посібник для студентів спеціальності «Програмна інженерія» ДонНТУ

Протокол №6 від 20.06.2014

Д21 Дацун Н.М.
Об’єктно-орієнтоване програмування: навчальний посібник для студентів спеціальності «Програмна інженерія». – Донецк: ДонНТУ, 2014. – 205 с.

Книга являє собою навчальний посібник для освоєння технологій об'єктно-орієнтованого програмування студентами напряму підготовки «Програмна інженерія» рівня підготовки бакалавр. У ній розглядаються основні поняття аналізу, проектування і програмування об'єктно-орієнтованих систем. Структура посібника відповідає змістовним модулям чинного галузевого стандарту «Освітньо-професійна програма» напряму підготовки 050103 «Програмна інженерія». Детально розкриті технологічні етапи об'єктно-орієнтованого аналізу, об'єктно-орієнтованого проектування мовою UML, а також створення програмного коду на базі основоположних конценції інкапсуляції, успадкування та поліморфізму. У ролі об'єктно-орієнтованої мови програмування в посібнику використовується C++.

Посібник призначений для студентів напряму підготовки 050103 «Програмна інженерія» рівня підготовки бакалавр. Може бути рекомендований студентам напрямів підготовки «Комп'ютерні науки» та «Комп'ютерна інженерія», які вивчають основи об'єктно-орієнтованого програмування.

ЗМІСТ

РОЗДІЛ 9. РОЗПОДІЛ ПОВЕДІНКИ ТА РЕАЛІЗАЦІЇ

Мета: розглянути приклад створення програмної системи з використанням інкапсуляції, розподілу поведінки та реалізації.

9.1 Інтерфейс і реалізація модуля

Програміст знає, як використати компоненту, написану іншим програмістом, але не повинен знати, як вона реалізована.

Говорять, що компонента інкапсулює поводження об'єкта, якщо вона вміє виконувати деякі дії, але подробиці, як саме вона це робить, сховані.

Це приводить до двох подань про програмну систему.

- вид з боку інтерфейсу - це лицьова сторона. в інтерфейсній частини описується, що вміє робити компонента;
- вид з боку реалізації - це виворіт, вона визначає, як компонента - виконує завдання.

9.2 Приклад рішення завдання: проект TIME

Визначити клас для роботи з об'єктом час. Знайти час, що наступає за вхідним (на одну секунду вперед).

1. Створюємо файл інтерфейсу TIME.HPP, в який поміщаємо опис класу CTime.
2. Створюємо файл реалізації TIME.CPP, в який поміщаємо тексти всіх функцій-членів класу CTime з його протоколу.

В файл TIME.CPP підключаємо файл інтерфейсу TIME.HPP для доступу до опису класу CTime.

3. Створюємо файл використання MAIN1.CPP, в який поміщаємо текст головної функції з створеним об'єктом класу CTime.

В файл MAIN1.CPP підключаємо файл інтерфейсу TIME.HPP для доступу до опису класу CTime.

4. Створюємо файл проекту (або рішення, Solution) TIME.PRJ, в який поміщаємо інформацію про файли TIME.CPP та MAIN1.CPP для їх об'єднання в єдину програму.

9.3 Файл інтерфейсу TIME.HPP

```
class CTime
{     int m_hh;     // дані-члени класу
      int m_mm;
      int m_ss;
public:
      CTime();      // конструктор
      ~CTime();     // деструктор
```

```
// функції-члени “ВЗЯТИ”
        int  GetHh ();
        int  GetMm();
        int  GetSs  ();
        void GetTime (char *buf);
// функції-члени “ПРИЗНАЧИТИ”
        int SetHh (int);
        int SetMm(int);
        int SetSs  (int);
        int SetTime (int AHh,int AMm,int ASs);
        int SetTime (char *buf);
//  функція-член ДІЯ
        void AddSec();
  };
```

9.4 Тестування алгоритмів методів (помилки в даних)

Перша група тестів алгоритмів методів для випадку помилок в даних:

тести						
1)	m_hh	.	m_mm	.	m_ss	
	09	.	40	.	75	
	недопустимо					
2)	m_hh	.	m_mm	.	m_ss	
	09	.	94	.	40	
	недопустимо					
3)	m_hh	.	m_mm	.	m_ss	
	29	.	40	.	30	
	недопустимо					

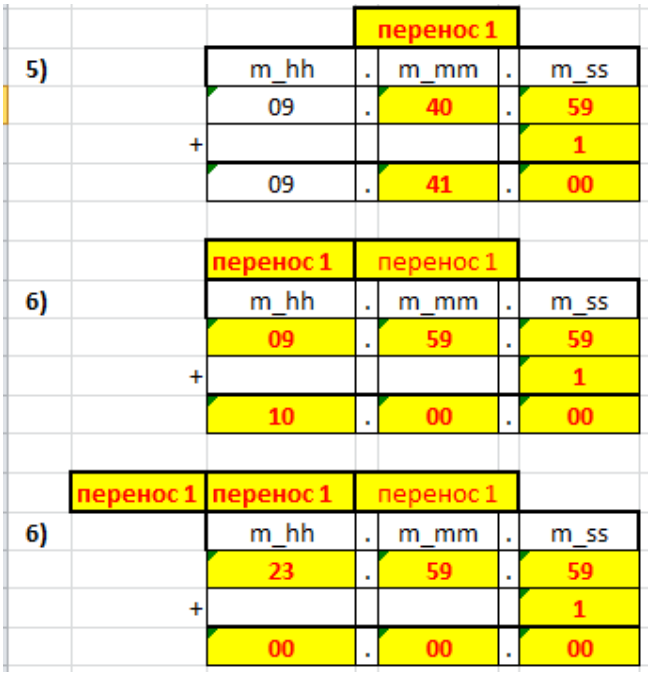
9.5 Тестування алгоритму Action-методу (в межах однієї хвилини)

Цей тест для Action-методу AddSec() для випадку зміни даних в межах однієї хвилини:

4)	m_hh	.	m_mm	.	m_ss
	09	.	40	.	15
+					1
	09	.	40	.	16

9.6 Тестування алгоритму Action-методу (за межами однієї хвилини)

Це тести для Action-методу AddSec() для випадку зміни даних за межами однієї хвилини з перенесенням до "старших розрядів":



9.7 Файл реалізації TIME.CPP

```
#include <stdio.h>
#include "time.hpp"
const KH=24;
const KM=60;
const KS=60;
//      КОНСТРУКТОР
CTime::CTime()
{       m_hh=0;
        m_mm=0;
        m_ss=0;
}
//     ДЕСТРУКТОР
CTime::~CTime()
{}
//    МЕТОДИ
void CTime::AddSec ()
{       if(m_ssCTime::GetTime       (char
*buf)
{sprintf(buf,"%02d. %02d. %02d",
                    m_hh,   m_mm,  m_ss);
}
int  CTime::GetSs  ()
{       return m_ss; }
int  CTime::GetMm ()
{       return m_mm; }
int  CTime::GetHh ()
{       return m_hh; }
```

```
int CTime::SetTime
        (int AHh,int AMm,int ASs)
{       return SetHh(AHh)&&
        SetMm(AMm)&&SetSs(ASs);
}
int CTime::SetTime (char *buf)
{       int h,m,s;
        sscanf(buf,"%d.%d.%d",
        &h,&m,&s);
        return SetTime(h,m,s);
}
int CTime::SetSs  (int ASs)
{       if(ASs>=0&&ASs<=KS-1)
        {       m_ss=ASs; return 1; }
        else
                return 0;
}
int CTime::SetMm (int AMm)
{       if(AMm>=0&&AMm<=KM-1)
        {       m_mm=AMm; return 1; }
        else
                return 0;
}
int CTime::SetHh (int AHh)
{       if(AHh>=0&&AHh<=KH-1)
        {m_hh=AHh; return 1; }
        else            return 0; }
```

9.8 Файл використання MAIN1.CPP

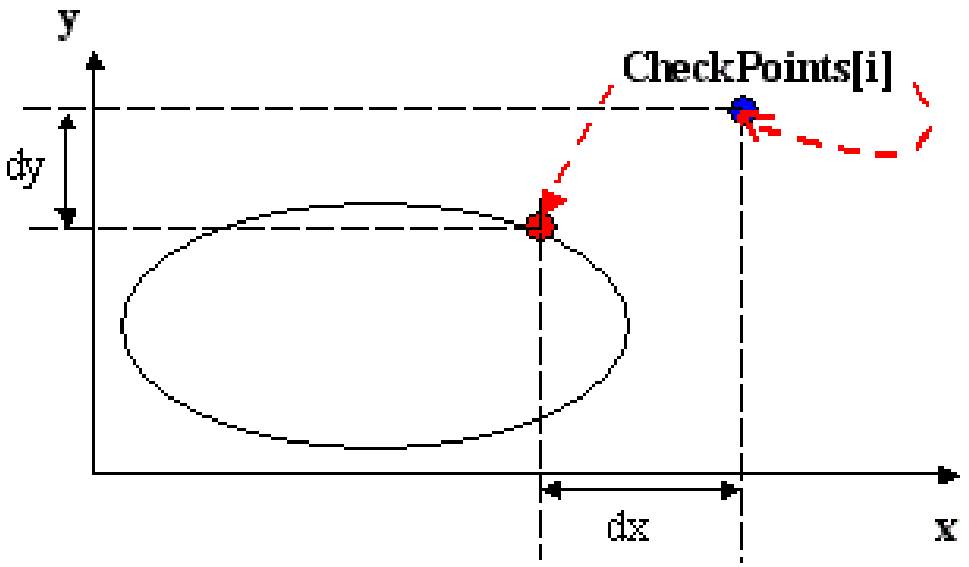
```
#include <stdio.h>
#include <string.h>
#include "time.hpp"
int main ()
{       CTime time;
        char buf[32];

        while(1)
        {
                printf("Please, type a time ('q' to exit) “
                    “[hh.mm.ss]: ");
                scanf("%s",buf);
                if(!strcmp(buf,"q")) break;
                if(!time.SetTime(buf))
                {printf("The TIME You have typed is “
                      “invalid.\n");
                        continue;
                }
                time.AddSec();
                time.GetTime(buf);
                printf("Next time: %s\n",buf);
        }
        return 0;}
```

10.5 Приклад програми з використанням механізму спадкування

Приклад 10.4:

Визначити, чи лежать всі точки заданої множини поза заданим еліпсом (спадкування з використанням ієрархії класів).

Еліпс CheckEllipse – об'єкт класу *Ellipse*.
Множина точок CheckPoints [POINT_COUNT] – масив об'єктів класу *Point*.



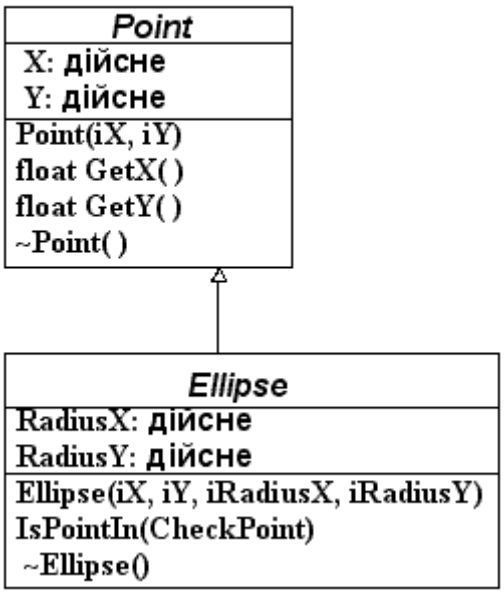
Канонічне рівняння еліпса:

$$\left(\frac{x-xc}{rx}\right)^2+\left(\frac{y-yc}{ry}\right)^2=1$$

(xc, yc) - координати точки центра;
rx, ry - величини напіврадіусів.

Спроектуємо діаграму класів проекту задачі.

Діаграма класів



Текст коду програми мовою С++:

```
// Визначити, чи лежать всі крапки заданої множини поза заданим еліпсом.
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define POINT_COUNT 5
class Point { // базовий клас:
protected:
 float X,Y;
public:
 Point(float iX, float iY);
 float Get();
 float Get();
         ~Point();
         };
// похідний клас:
class Ellipse : public Point {
private:
 float RadiusX, RadiusY;
public:
 Ellipse(float iX, float iY, float iRadiusX, float iRadiusY);
 int IsPointIn(Point * CheckPoint);
 ~Ellipse();
};
```

```
// реалізація базового класу
Point::Point(float iX,float iY)
{
 X=iX;
 Y=iY;
}

float Point::Get()
{
 return X;
}

float Point::Get()
{
 return Y;
}

Point::~Point()
{
}
```

```
// реалізація похідного класу
Ellipse::Ellipse(float iX, float iY, float iRadiusX, float iRadiusY)
        :Point(iX,Yi)
{ if (iRadiusX<=0.0 || iRadiusY<=0.0)
 {  printf("\nнапіврадіуси еліпса повинні бути позитивними\n"); exit(-1);  }
 Radius=iRadiusX;
 Radius=iRadiusY;
}
int Ellipse::IsPointIn(Point * CheckPoint)
{ float d,d;
 d=X-CheckPoint->Get();
 d=Y-CheckPoint->Get();
 return (d*d)/(RadiusX*RadiusX)+(d*d)/(RadiusY*RadiusY)>1.0;
}
Ellipse::~Ellipse() { }
```

```
// використання класів:
int main()
{Point * CheckPoints[POINT_COUNT];
 Ellipse * CheckEllipse;
 int i;
 float x,y,rx,ry;
 int PointInCount;
 for (i=0;i<POINT_COUNT;i++)
 {printf("\nВведіть координати %d-й крапки: ",i+1);
  scanf("%f%f",&x,&y);
  CheckPoints[i]=new Point(x,y);
 }
 printf("\nВведіть координати центра еліпса: ");
 scanf("%f%f",&x,&y);
 printf("\nВведіть радіуси еліпса: ");
 scanf("%f%f",&rx,&ry);
 CheckEllipse=new Ellipse(x,y,rx,ry);
 PointInCount=0;
 for (i=0;i<POINT_COUNT;i++)
  if (CheckEllipse->IsPointIn(CheckPoints[i]))
    PointInCount++;
  if (PointInCount==POINT_COUNT)
    printf("\nВсі крапки множини перебувають поза еліпсом\n");
  else
    printf("\nНе всі крапки множини перебувають поза еліпсом\n");
 printf("\nНажміть будь-яку клавішу для виходу...\n");
 getch();}
```

15.5 Приклад програми з перевантаженням операторів: проект TIME

Приклад 15.1:

Перевантажити в класі CTime оператори: ++ (в класі); +- (в класі); + (поза класом). Зміни внесемо в існуючий клас CTime

15.5.1 Файл інтерфейсу TIME.HPP

```
class CTime
{       int m_hh;    // дані-члени класу
        int m_mm;
        int m_ss;
public:
        CTime();       // конструктор
        ~CTime();     // деструктор
// функції-члени “ВЗЯТИ”
        int  GetHh ();
        int  GetMm();
        int  GetSs  ();
        void GetTime (char *buf);
// функції-члени “ПРИЗНАЧИТИ”
        int SetHh (int);
        int SetMm(int);
        int SetSs  (int);
        int SetTime (int AHh,int AMm,int ASs);
        int SetTime (char *buf);
//  функція-член ДІЯ
 void AddSec();
// ОПЕРАТОРИ перевантажені !!!!
        void operator++(int);
        void operator+=(CTime &);
        friend CTime operator+ (CTime& t1, CTime &t2);
};
```

15.5.2 Файл реалізації TIME.CPP

```
#include <stdio.h>
#include "time.hpp"
const KH=24;
const KM=60;
const KS=60;
//      КОНСТРУКТОР
CTime::CTime()
{       m_hh=0;
        m_mm=0;
        m_ss=0;
}
//     ДЕСТРУКТОР
CTime::~CTime()
{}
//    МЕТОДИ
void CTime::AddSec ()
{       if(m_ssCTime::GetTime       (char
*buf)
{sprintf(buf,"%02d. %02d. %02d",
                m_hh,   m_mm,  m_ss);
}
int  CTime::GetSs  ()
{       return m_ss; }
int  CTime::GetMm ()
{       return m_mm; }
int  CTime::GetHh ()
{       return m_hh; }
int CTime::SetTime
              (int AHh,int AMm,int ASs)
{       return
                SetHh(AHh)&&
                SetMm(AMm)&&
                SetSs(ASs);
}
 int CTime::SetTime (char *buf)
{       int h,m,s;
        sscanf(buf,"%d.                %d.
%d",&h,&m,&s);
        return SetTime(h,m,s);
}
```

```
int CTime::SetSs  (int ASs)
{       if(ASs>=0&&ASs<=KS-1)
        {       m_ss=ASs; return 1; }
        else
                return 0;
}
int CTime::SetMm (int AMm)
{       if(AMm>=0&&AMm<=KM-1)
        {       m_mm=AMm; return 1; }
        else
                return 0;
}
int CTime::SetHh (int AHh)
{       if(AHh>=0&&AHh<=KH-1)
        {m_hh=AHh; return 1; }
        else
                return 0;
}

//  ПЕРЕВАНТАЖЕННЯ:
// ++
void CTime::operator++(int)
{ AddSec(); }

// ДОДАВАННЯ ДВОХ ЧАСІВ
void CTime::operator+=(CTime &t)
{ int h1, m1, s1, h, m, s;
        h1=t.GetHh();
        m1=t.GetMm();
        s1=t.GetSs();
        m_ss+=s1;
        if(m_ss>KS)
                {m_ss-=KS; m_mm++;}
        m_mm+=m1;
        if(m_mm>KM)
                {m_mm-=KM; m_hh++;}
        m_hh+=h1;
        if(m_hh>KH)
                {m_hh-=KH;}
}
```

```
// БІНАРНИЙ ОПЕРАТОР +
 CTime operator+(CTime& t1, CTime &t2)
 {      int h1, m1, s1, h2, m2, s2, h, m, s;
        h1=t1.GetHh();
        m1=t1.GetMm();
        s1=t1.GetSs();
        h2=t2.m_hh;
        m2=t2.m_mm;
        s2=t2.m_ss;
        s=m=h=0;
        s=s1+s2;
        if(s>KS) {s-=KS; m++;}
        m+=(m1+m2);
        if(m>KM)   {m-=KM; h++;}
        h+=(h1+h2);
        if(h>KH)    {h-=KH;}
returnCTime(h, m, s);
 }
```

15.5.3 Файл використання MAIN1.CPP

```
#include <stdio.h>
#include <string.h>
#include "time.hpp"
int main ()
 {CTime time;
  CTime time2;
  CTime time3;
  char buf[32];
  while(1)
     {printf("Please, type a time “
               “('q' to exit)“
           “ [hh.mm.ss]: ");
     scanf("%s",buf);
     if(!strcmp(buf,"q")) break;
     if(!time.SetTime(buf))
         {printf("The TIME You have “
                 “typed is invalid.\n");
          continue;
         }
     time.AddSec();
     time.GetTime(buf);
     printf("\nNext time after AddSec:”
             “ %s\n",buf);
     time++;
     time.GetTime(buf);
     printf("\nNext time after ++: “
             “%s\n",buf);
while(1)
   {printf("Please, type second time “
             “ ('q' to exit) [hh.mm.ss]: ");
  scanf("%s",buf);
  if(!strcmp(buf,"q")) break;
     if(!time2.SetTime(buf))
        {printf("The TIME You “
                “ have typed is invalid.\n");
         continue;     }
     time.GetTime(buf);
     printf("\nTime do +: %s\n",buf);
     time2.GetTime(buf);
     printf("\n Time2 do +: %s\n",buf);
     time+=time2;
     time.GetTime(buf);
     printf("\nTime after +: %s\n",buf);
     time2.GetTime(buf);
     printf("\nTime2 after +: %s\n", buf);
     time3=time+time2;
     time3.GetTime(buf);
     printf("\nTime3 after time+time2: “
             “ %s\n",buf);
      }
  }
  return 0;
}
```

ПРЕДМЕТНИЙ ПОКАЖЧИК

ВИТЯГ З ГАЛУЗЕВОГО СТАНДАРТУ ВИЩОЇ ОСВІТИ УКРАЇНИ
ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА БАКАЛАВР
НАПРЯМУ ПІДГОТОВКИ 050103 “ПРОГРАМНА ІНЖЕНЕРІЯ”
КИЇВ 2013 Р.

Таблиця – Рекомендований перелік навчальних дисциплін і практик

Шифр навчальної дисципліни	Назва навчальної дисципліни	Назва змістового модуля	Шифр змістового модулю
3.05	Об'єк-тно-орієнтоване програмування (у тому числі курсова робота)	Об'єктно-орієнтоване проектування	1.ПФ.Д.04.03.01
		Інкапсуляція та приховання інформації	1.ПФ.Д.04.03.02
		Розподіл поведінки та реалізації	1.ПФ.Д.04.03.03
		Класи та підкласи	1.ПФ.Д.04.03.04
		Успадкування (перевизначення, динамічне зв'язування).	1.ПФ.Д.04.03.05
		Поліморфізм (поліморфізм підтипів і успадкування).	1.ПФ.Д.04.03.06
		Ієрархія класів.	1.ПФ.Д.04.03.07
		Класи колекцій і протоколи ітерації	1.ПФ.Д.04.03.08
		Внутрішнє представлення об'єктів і таблиця методів	1.ПФ.Д.04.03.09

ВИТЯГ З РОБОЧОЇ ПРОГРАМИ ДИСЦИПЛІНИ “ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ” ДЛЯ СТУДЕНТІВ ЗА НАПРЯМОМ ПІДГОТОВКИ 6.050103 “ПРОГРАМНА ІНЖЕНЕРІЯ”

1. Опис навчальної дисципліни

<table>
<tr><td rowspan="2">Найменування показників</td><td rowspan="2">Галузь знань, напрям підготовки, освітньо-кваліфікаційний рівень</td><td>Характеристика навчальної дисципліни</td></tr>
<tr><td>денна форма навчання</td></tr>
<tr><td rowspan="2">Кількість кредитів – 8</td><td>Галузь знань
0501 Інформатика та обчислювальна техніка</td><td rowspan="2">Нормативна</td></tr>
<tr><td>Напрям підготовки
6.050103 “Програмна інженерія”</td></tr>
<tr><td>Модулів – 2</td><td rowspan="5">Спеціальність:
6.05010302 “Інженерія програмного забезпечення”</td><td>Рік підготовки</td></tr>
<tr><td>Змістових модулів – 9</td><td>1-й</td></tr>
<tr><td>Індивідуальне науково-дослідне завдання- не передбачене</td><td>Семестр</td></tr>
<tr><td rowspan="2">Загальна кількість годин – 288</td><td>2-й</td></tr>
<tr><td>Лекції</td></tr>
<tr><td rowspan="13">Тижневих годин для денної форми навчання:
аудиторних – 6
самостійної роботи студента – 5.4</td><td rowspan="13">Освітньо-кваліфікаційний рівень:
бакалавр</td><td>48 год.</td></tr>
<tr><td>Практичні, семінарські</td></tr>
<tr><td>-</td></tr>
<tr><td>Лабораторні</td></tr>
<tr><td>48 год.</td></tr>
<tr><td>Самостійна робота</td></tr>
<tr><td>86 год.</td></tr>
<tr><td>Індивідуальні завдання:</td></tr>
<tr><td>-</td></tr>
<tr><td>Вид контролю:</td></tr>
<tr><td>екз.</td></tr>
</table>