

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». – Донецьк: 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 концепції 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.

© Дацун Н.М., 2014

## ЗМІСТ

Вступ. . . . .	8
Частина 1. Об'єктно-орієнтоване проектування. . . . .	9
Розділ 1. Загальні поняття ООП. . . . .	10
1.1 Природа об'єкта. . . . .	10
1.2 Стан. . . . .	11
1.3 Поведінка. . . . .	13
1.4 Операції. . . . .	13
1.5 Ролі та відповідальності . . . . .	14
1.6 Об'єкти як автомати. . . . .	14
1.7 Ідентичність. . . . .	15
Розділ 2. Співвідношення «клас-об'єкт». Відношення між об'єктами та між класами. . . . .	16
2.1 Співвідношення «клас-об'єкт» . . . . .	16
2.2 Відношення між класами. . . . .	17
2.3 Асоціація. . . . .	18
2.3.1 Множинність асоціації. . . . .	19
2.3.2 Приклади асоціацій з різними типами потужності. . . . .	20
2.3.3 Інші асоціації. . . . .	20
2.4 Агрегація і композиція. . . . .	20
2.4.1 Агрегація. . . . .	21
2.4.2 Композиція. . . . .	22
2.5 Узагальнення. Граф узагальнень. . . . .	23
2.5.1 Успадкування. . . . .	24
2.5.2 Особливі випадки успадкування. . . . .	25
2.5.3 Делегування. . . . .	25
2.6 Рекомендації щодо вибору відношень між класами. . . . .	26
Розділ 3. Об'єктно-орієнтоване проектування програм. Мова UML. . .	27
3.1 Об'єктно-орієнтоване проектування програм. . . . .	27
3.2 Історія виникнення мови UML. . . . .	27
3.3 Характеристика мови UML. . . . .	27
3.4 Словник мови UML. . . . .	28
Розділ 4. Діаграми класів/взаємодій. . . . .	30
4.1 Позначення класів в UML. . . . .	30
4.2 Позначення відношення "Асоціація". . . . .	31
4.3 Позначення відношення "Агрегація". . . . .	32
4.4 Позначення відношення "Узагальнення". . . . .	32
4.5 Діаграми класів. . . . .	34
4.6 Діаграми об'єктів. . . . .	35
Розділ 5. Діаграми станів/послідовностей. . . . .	37
5.1 Виконавці та прецеденти. . . . .	37
5.2 Діаграми прецедентів. . . . .	38
5.3 Потоки подій. . . . .	38
5.4 Аналіз стійкості. . . . .	40

5.5 Повідомлення та дії. . . . .	41
5.5.1 Дія виклику. . . . .	41
5.5.2 Дія повернення. . . . .	42
5.5.3 Дія створення. . . . .	42
5.5.4 Дія знищення. . . . .	42
5.5.5 Дія відправлення. . . . .	43
5.6 Діаграма послідовностей. . . . .	44
5.7 Приклад діаграми послідовностей для предметної області "телефонія". . . . .	45
5.8 Діаграма кооперації. . . . .	45
5.9 Існування об'єктів усередині системи. . . . .	46
5.9.1 Події. . . . .	46
5.9.2 Стани . . . . .	47
5.9.3 Переходи. . . . .	47
5.9.4 Умови. . . . .	48
5.10 Машини та діаграми станів. . . . .	48
5.11 Додаткова інформація на діаграмі станів. . . . .	49
5.12 Приклади діаграм станів для різних предметних областей. . . . .	49
Питання, задачі, тести частини 1. . . . .	50
Частина 2. Інкапсуляція та приховання інформації. . . . .	56
Розділ 6. Поняття класу. Члени класу. Конструктори і деструктори. . . . .	57
6.1 Поняття інкапсуляції. . . . .	57
6.2 Клас . . . . .	59
6.3 Дані-члени і керування доступом до елементів класів. . . . .	60
6.4 Функції-члени. . . . .	61
6.5 Конструктори та деструктори. . . . .	61
6.6 Конструктори і деструктори в мові C++. . . . .	62
6.7 Приклад рішення завдання. . . . .	63
Розділ 7. Семантика копіювання. Дружні класи та дружні функції. . . . .	65
Класи пам'яті об'єктів. . . . .	65
7.1 Конструктор копіювання. . . . .	65
7.2 Приклад створення класу "Рядок". . . . .	66
7.3 Конструктори і масиви об'єктів. . . . .	66
7.4 Дружні функції. . . . .	68
7.5 Дружні класи та приклад їх використання. . . . .	68
7.6 Класи пам'яті об'єктів . . . . .	69
Питання, задачі, тести частини 2. . . . .	70
Частина 3. Розподіл поведінки та реалізації. . . . .	73
Розділ 8. Принцип Парнаса. . . . .	74
8.1 Принцип Парнаса. . . . .	74
8.2 Принцип Парнаса в термінах об'єктів. . . . .	74
8.3 Приклади використання принципу Парнаса. . . . .	75
Розділ 9. Розподіл поведінки та реалізації. . . . .	77
9.1 Інтерфейс і реалізація модуля. . . . .	77

9.2 Приклад рішення завдання: проект TIME. ....	77
9.3 Файл інтерфейсу TIME.HPP. ....	77
9.4 Тестування алгоритмів методів (помилки в даних) ....	78
9.5 Тестування алгоритме Action-методу (в межах однієї хвилини) ...	78
9.6 Тестування алгоритму Action-методу (за межами однієї хвилини).	78
9.7 Файл реалізації TIME.CPP. ....	79
9.8 Файл використання MAIN1.CPP. ....	80
9.9 Файл проекту TIME.PRJ. ....	80
Питання, задачі, тести тести частини 3. ....	81
Частина 4. Класи та підкласи. ....	83
Розділ 10. Класи та підкласи. Успадкування класів. ....	84
10.1 Класи та підкласи. ....	84
10.2 Успадкування класів. ....	85
10.3 Конструктори похідних класів. ....	87
10.4 Деструктори похідних класів. ....	87
10.5 Приклад програми з використанням механізму спадкування. ....	88
Розділ 11. Контейнерні класи. Порядок визову конструкторів та деструкторів при успадкуванні. ....	91
11.1 Способи включення об'єктів. ....	91
11.2 Приклад програми, написаної з використанням механізму включення. ....	91
11.3 Контейнерні класи. ....	94
11.4 Контейнерний клас "Вектор". ....	95
11.5 Контейнерний клас "Список". ....	96
11.6 Порівняння механізмів включення та успадкування. ....	97
11.7 Порядок визову конструкторів та деструкторів при успадкуванні	98
11.8 Використання контейнерних класів. ....	98
11.9 Використання контейнерних класів. Закрите успадкування. ....	99
11.10 Правила доступу для друзів класів і похідних класів. ....	101
11.11 Дружність і успадкування. ....	102
Питання, задачі, тести тести частини 4. ....	104
Частина 5. Успадкування (перевизначення, динамічне зв'язування). ..	107
Розділ 12. Перевизначення. Динамічне зв'язування. ....	108
12.1 Типізовані перетворення та видимість. ....	108
12.2 Перевизначення. ....	109
12.3 Ієрархія класів з перевизначенням. ....	109
12.4 Ієрархія об'єктів та перевизначення. ....	110
Розділ 13. Методи доступу до членів класу при успадкуванні. ....	111
13.1 Методи доступу до членів з закритої частини класу. ....	111
13.2 Методи доступу до членів з захищеної частини класу. ....	112
13.3 Методи доступу до членів з відкритої частини класу. ....	113
Питання, задачі, тести тести частини 5. ....	114
Частина 6. Поліморфізм (поліморфізм підтипів і успадкування) ....	120
Розділ 14. Принцип абстракції та поліморфізму. Перевантаження	

функцій та конструкторів. . . . .	121
14.1 Принцип абстракції. . . . .	121
14.2 Принцип поліморфізму. . . . .	121
14.3. Види поліморфізму. Поліморфізм імен. . . . .	123
14.4 Функції, що перевантажені (overload) . . . . .	124
14.5 Вибір конкретної функції. . . . .	126
14.6 Перевантаження методів і конструкторів. . . . .	128
Розділ 15. Перевантаження операторів. . . . .	129
15.1 Поняття “перевантаження операторів”. . . . .	129
15.2 Способи перевантаження операторів. Синтаксис функції-оператора. . . . .	129
15.3 Перевантаження унарних операторів. . . . .	129
15.4 Перевантаження бінарних операторів. . . . .	130
15.5 Приклад програми з перевантаженням операторів: проект TIME. . . . .	130
15.5.1 Файл інтерфейсу TIME.HPP. . . . .	130
15.5.2 Файл реалізації TIME.CPP. . . . .	131
15.5.3 Файл використання MAIN1.CPP. . . . .	132
15.5.4 Файл проекту TIME.PRJ. . . . .	132
15.6 Погодженість перевантажених операторів. . . . .	133
15.7 Операторна і функціональна форми використання перевантажених операторів. . . . .	133
15.8 Перевантаження оператора присвоювання = . . . . .	133
15.9 Перевантаження оператора вирізки [ ] . . . . .	135
15.10 Перевантаження операторів new, delete, -> . . . . .	136
Розділ 16. Статичний та динамічний поліморфізм. Поліморфізм підтипів і успадкування. Раннє та пізнє зв'язування. . . . .	138
16.1 "Раннє" і "пізнє" зв'язування. . . . .	138
16.2 Віртуальні функції і поліморфічні кластери. . . . .	138
16.3 Механізм створення поліморфічного кластера. . . . .	140
16.4 Особливості віртуальної функції. . . . .	141
Питання, задачі, тести частини 6. . . . .	143
Частина 7. Ієрархія класів. . . . .	148
Розділ 17. Ієрархія класів. Множинне успадкування. Визови конструкторів та деструкторів при множинному успадкуванні. . . . .	149
17.1 Ієрархія множинного успадкування. . . . .	149
17.2 Конфлікт імен даних-членів класу. . . . .	151
17.3 Конфлікт імен функцій-членів класу. . . . .	152
17.4 Порядок виклику конструкторів при множинному успадкуванні. . . . .	152
Питання, задачі, тести частини 7. . . . .	155
Частина 8. Класи колекцій і протоколи ітерації. . . . .	157
Розділ 18. Узагальнене програмування. Шаблони функцій. Параметризовані функції і класи. Шаблони класів. . . . .	158
18.1 Узагальнене програмування. . . . .	158
18.1.1 Загальний механізм узагальненого програмування. . . . .	158

18.1.2 Способи реалізації. . . . .	159
18.2 Шаблони функцій. . . . .	160
18.3 Параметризовані функції і класи. . . . .	162
18.4 Шаблони класів. . . . .	163
Розділ 19. Обробка виключних ситуацій. . . . .	165
19.1 Помилки та їх обробка. . . . .	165
19.2. Генерація та перехоплення виключних ситуацій. . . . .	166
19.3 Обробка виключних ситуацій та класи. . . . .	168
Розділ 20. Стандартна бібліотека шаблонів. Класи колекцій і протоколи ітерації. . . . .	170
20.1 Стандартна бібліотека шаблонів. . . . .	170
20.2 Класи колекцій і протоколи ітерації. . . . .	170
20.2.1 Контейнери. . . . .	170
20.2.2 Розподільники пам'яті. . . . .	171
20.2.3 Ітератори. . . . .	171
20.2.4 Алгоритми. . . . .	173
Питання розділа 8. . . . .	175
Частина 9. Внутрішнє представлення об'єктів і таблиця методів. . . . .	176
Розділ 21. Внутрішнє представлення об'єктів і таблиця методів. . . . .	177
Розділ 22. Вказівник this. Статичні методи та статичні дані-члени класу. . . . .	179
22.1 Ключове слово this. . . . .	179
22.2 Статичні члени класу. . . . .	180
22.3 Статичні поля класу. . . . .	180
22.4 Статичні методи класу. . . . .	180
Питання, задачі, тести тести частини 9. . . . .	182
Бібліографічний список. . . . .	184
Предметний покажчик. . . . .	185
Відповіді на тести. . . . .	194
Витяг з галузевого стандарту вищої освіти України «Освітньо-професійна програма бакалавр напряму підготовки 050103 “Програмна інженерія”», Київ 2013 р. . . . .	196
Витяг з робочої програми дисципліни “Об’єктно-орієнтоване програмування” для студентів за напрямом підготовки 6.050103 “Програмна інженерія”. . . . .	197
Довідковий матеріал. Відмінності мов Сі та С++. Потоки в С++. . . . .	198

## 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 поведження 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":

				перенос 1	
5)	m_hh	.	m_mm	.	m_ss
	09	.	40	.	59
					1
	+				
	09	.	41	.	00
			перенос 1	перенос 1	
6)	m_hh	.	m_mm	.	m_ss
	09	.	59	.	59
					1
	+				
	10	.	00	.	00
			перенос 1	перенос 1	перенос 1
6)	m_hh	.	m_mm	.	m_ss
	23	.	59	.	59
					1
	+				
	00	.	00	.	00

### 9.7 Файл реалізації TIME.CPP

<pre> #include &lt;stdio.h&gt; #include "time.hpp" const KH=24; const KM=60; const KS=60; //    КОНСТРУКТОР CTime::CTime() {     m_hh=0;     m_mm=0;     m_ss=0; } //    ДЕКТРУКТОР 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; } </pre>	<pre> int CTime::SetTime     (int AHh,int AMm,int ASs) {     return SetHh(AHh)&amp;&amp;     SetMm(AMm)&amp;&amp;SetSs(ASs); } int CTime::SetTime (char *buf) {     int h,m,s;     sscanf(buf,"%d.%d.%d",     &amp;h,&amp;m,&amp;s);     return SetTime(h,m,s); } int CTime::SetSs (int ASs) {     if(ASs&gt;=0&amp;&amp;ASs&lt;=KS-1)     {         m_ss=ASs; return 1; }     else         return 0; } int CTime::SetMm (int AMm) {     if(AMm&gt;=0&amp;&amp;AMm&lt;=KM-1)     {         m_mm=AMm; return 1; }     else         return 0; } int CTime::SetHh (int AHh) {     if(AHh&gt;=0&amp;&amp;AHh&lt;=KH-1)     {m_hh=AHh; return 1; }     else         return 0; } </pre>
---	---

## 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;}

```

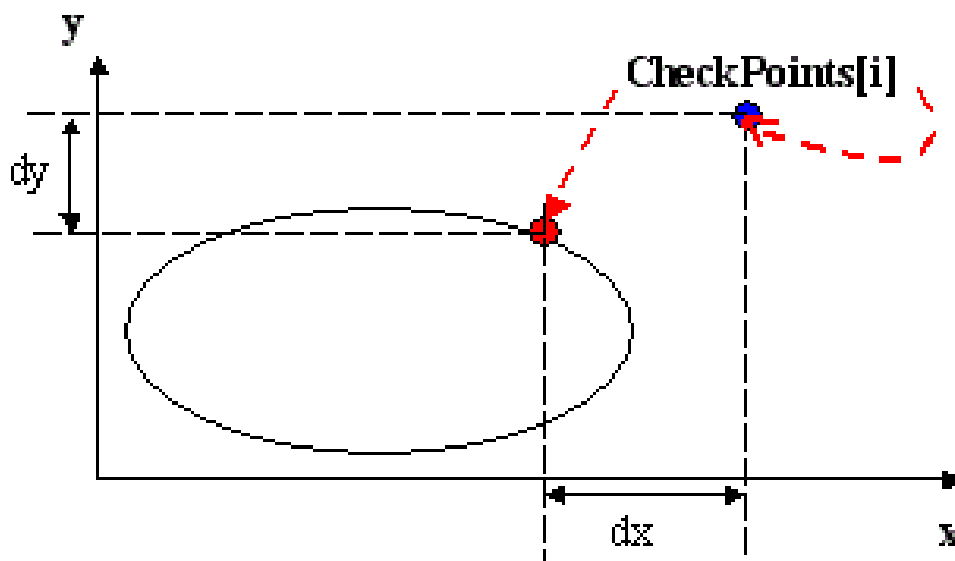
## 10.5 Приклад програми з використанням механізму спадкування

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:

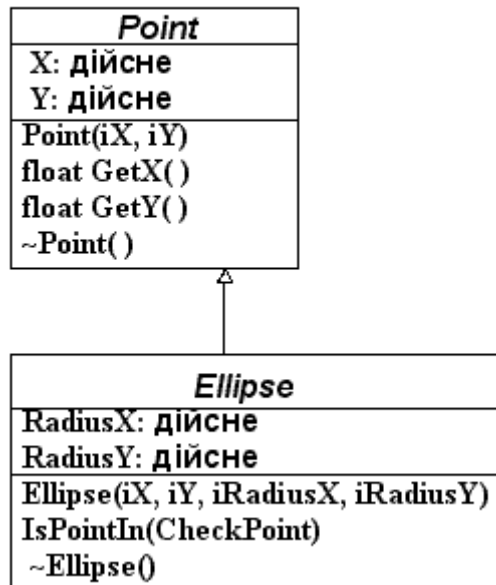
$$\left(\frac{x-x_c}{r_x}\right)^2 + \left(\frac{y-y_c}{r_y}\right)^2 = 1$$

$(x_c, y_c)$  - координати точки центра;

$r_x, r_y$  - величини напіврадіусів.

Спроекуємо діаграму класів проекту задачі.

### Діаграма класів



Текст коду програми мовою C++:

```
// 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 GetX();
    float GetY();
    ~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::GetX()
{
    return X;
}

float Point::GetY()
{
    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("\nрадіуси 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_ss>KS) CTime::GetTime (char
*buf)
{printf(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;}
}

```



<pre>// БІНАРНИЙ ОПЕРАТОР + CTime operator+(CTime&amp; t1, CTime &amp;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;</pre>	<pre>s=m=h=0; s=s1+s2; if(s&gt;KS) {s-=KS; m++;} m+=(m1+m2); if(m&gt;KM) {m-=KM; h++;} h+=(h1+h2); if(h&gt;KH) {h-=KH;} returnCTime(h, m, s); }</pre>
---	---

### 15.5.3 Файл використання MAIN1.CPP

<pre>#include &lt;stdio.h&gt; #include &lt;string.h&gt; #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);</pre>	<pre>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; }</pre>
---	---

## ПРЕДМЕТНИЙ ПОКАЖЧИК

абстракція, 28, 120  
абстрактний тип даних (АТД), 58, 74, 128  
автоматичний об'єкт, 69  
аналіз стійкості, 40  
агрегація, 20, 21, 25, 26, 32  
агрегування, 21  
активний об'єкт, 15  
алгоритми, 169, 172  
асоціація, 17, 18, 29, 31  
асоціативні контейнери, 94  
атрибут, 22, 30, 33, 36, 41

багаторівневий список, 45  
багато-до багатьох, 19  
батьківський клас, 32, 85  
бінарна асоціація, 19  
базовий клас, 60, 84  
батьківський клас, 60

варіант використання, 28  
взаємодія, 28  
взаємодія асинхронна, 43, 46  
взаємодія синхронна, 41, 46  
взаємно-однозначна залежність, 19, 20  
вид діяльності, 41  
видимість, 108  
виклик методу, 41  
виключення, 43  
виключна ситуація, 164, 165  
виключна асоціація, 20  
виконавець, 37, 38  
винятковий потік подій, 38, 39  
відкладена подія, 49  
відкрите успадкування, 98  
відкритий розділ, 60  
візуальне моделювання, 28  
відношення між класами, 17, 26  
відношення спадкування, 25  
відповідальності, 14  
віртуальна функція, 137, 140  
віртуальний метод, 60  
включення за посиланням, 22

властивість об'єкта, 12, 22  
внутрішнє представлення об'єктів, 176  
вузол, 23, 28

генерація виключних ситуацій, 165  
граф узагальнень, 23, 24

дані, 57  
дані-члени, 60, 61, 64  
делегування, 24, 25  
дерево, 147  
деструктор, 13, 61, 62, 67  
деструктор базового класу, 88  
динамічне зв'язування, 108, 123  
деструктор похідного класу, 87  
діаграма, 29  
діаграма взаємодії, 29  
діаграма діяльностей, 29  
діаграма класів, 34, 35, 88, 91  
діаграма компонент, 29  
діаграма кооперації, 45  
діаграма об'єктів, 29, 35, 36  
діаграма послідовностей, 29, 37, 44, 45, 46  
діаграма прецедентів, 38  
діаграма розгортання, 29  
діаграма станів, 29, 37, 48, 49  
діаграма стійкості, 40, 46  
дія, 41  
дія виклику, 41  
дія відправлення, 41, 43  
дія знищення, 41, 42, 43  
дія повернення, 41, 42  
дія при входженні, 49  
дія при виході, 49  
дія створення, 41, 42  
дозвіл перевантаження, 124  
дружні класи, 68, 101, 102  
дружні функції, 68, 128  
дружній метод, 60

екземпляр класу, 22

загальний клас, 32  
закон Деметера, 26

закрите успадкування, 91, 98, 99  
залежність, 29  
залежність "один-множина", 19, 20  
залежність "множина-множина", 19, 20  
замінюємість, 32  
захищений розділ, 60  
змінна this, 61, 178

ідентичність, 15  
ієрархія «є», 24  
ієрархія класів, 84  
ієрархія об'єктів, 91, 110  
ієрархія простого успадкування, 147  
ініціалізація, 61, 67  
ім'я асоціації, 31  
ім'я виконавця, 37  
ім'я деструктора, 62  
ім'я класа, 16, 35  
ім'я конструктора, 61  
ім'я об'єкта, 15, 35  
ім'я події, 48  
інкапсуляція, 26, 57  
інкрементальність, 28, 34  
інтерфейс класу, 28, 74, 77  
ітератор, 13  
ітератори, 169, 170  
ітеративність, 28, 34

керуючий клас, 40  
кінцевий стан, 47, 48  
клас, 16, 22, 26, 28, 30, 59, 64, 74, 84  
клас аналізу, 40  
клас-спадкоємець, 24, 25  
клас-сутність, 40  
клас-уточнення, 32  
класи пам'яті, 69  
ключове слово, 42  
коментар, 28  
композиція, 20, 21, 22  
компонент, 28  
константне поле, 63  
конструктор, 13, 61, 62, 66  
конструктор базового класу, 87, 108  
конструктор без параметрів, 66

конструктор за замовчуванням, 65  
конструктор копіювання, 65  
конструктор похідного класу, 87  
конструктор породженого класу, 87, 108  
контейнер, 94, 169  
контейнерні класи, 91, 94, 98, 99  
конфлікт імен, 148, 150, 151  
кооперація, 28  
кратність, 31, 32

лінія життя, 41, 42, 43, 44

масив об'єктів, 66  
машина станів, 48  
метод, 13, 26, 57  
методи доступу до членів класу при успадкуванні, 111, 112, 113  
механізм віртуальних функцій, 123  
механізм включення, 91, 97  
механізм спадкування, 88  
механізм сполучення імен, 123  
множинне спадкування, 24, 26, 33, 148  
множинне узагальнення, 24  
множинність асоціації, 19  
мова програмування, 28  
мова UML, 27  
модифікатор, 13

нащадок, 23, 24, 32  
нумерація повідомлень, 45

об'єкт, 10, 16, 44, 45, 59, 74  
об'єкт анонімний, 35  
об'єкт у вільній пам'яті, 69  
об'єкт-відправник, 41, 43  
об'єкт-одержувач, 41, 42  
об'єкт член, 69  
об'єктно-орієнтоване моделювання, 27  
об'єктно-орієнтоване проектування, 27, 40  
об'єктно-орієнтований аналіз, 16, 40  
обов'язки, 30  
одиначне успадкування, 24, 33  
один-до багатьох, 19, 20  
один-до одного, 19  
основний потік подій, 38, 39

оператор виведення <<, 149  
операторна форма використання перевантажених операторів, 132  
операція області видимості ::, 86, 150, 151  
операція, 13, 16, 30, 33

пакет, 28  
параметризовані класи, 161, 162  
параметризовані функції, 161  
параметричний поліморфізм, 123  
пасивний об'єкт, 15  
перевантажена функція, 60, 123  
перевантаження імен, 123  
перевантаження конструкторів, 127  
перевантаження методів, 127  
перевантаження бінарних операторів, 128, 129  
перевантаження оператора вирізки, 134  
перевантаження оператора присвоювання, 132  
перевантаження оператора delete, 135  
перевантаження оператора new, 135  
перевантаження оператора ->, 135  
перевантаження операторів, 128  
перевантаження унарних операторів, 128  
перевизначення функцій-членів класу, 108, 109, 110  
перехід, 47, 48  
перехід з охоронною умовою, 48  
перехоплення виключних ситуацій, 165  
підклас, 24, 32, 84, 85  
пізні зв'язування, 123, 137  
поведінка, 13, 25  
повідомлення, 26, 41, 44, 45, 46  
повторення, 46  
подія, 46, 47, 48,  
подія виклику, 46  
показчик на базовий клас, 109  
показчик на породжений клас, 109  
поліморфізм, 32, 33, 120  
поліморфізм імен, 122, 123  
поліморфічний кластер, 137, 139  
поток подій, 38  
породжений клас, 85  
порядок виклику деструкторів при успадкуванні, 98  
порядок виклику конструкторів при успадкуванні, 98  
порядок виклику конструкторів при множинному успадкуванні, 151  
посилання на базовий клас, 108

посилання на породжений клас, 108  
послідовні контейнери, 94  
похідний клас, 84, 101  
початковий стан, 47, 48  
правила доступу, 101  
прецедент, 28, 37, 38, 40  
приватний розділ, 59  
прикордонний клас, 40  
принцип абстракції, 120  
принцип Парнаса, 74, 75  
принцип поліморфізму, 120  
принципи узагальнення, 32  
просте успадкування, 24  
протокол, 14  
прямий ациклічний граф, 147

раннє зв'язування, 123, 137  
реакція об'єкта, 46, 48, 49, 77  
реалізація класу, 73  
ребро, 23  
розподіл інтерфейсу й реалізації, 74  
розподільники пам'яті, 169, 170  
роль, 14, 31, 32, 37

самоперехід, 47  
секція, 30, 43  
селектор, 13  
сигнал, 43, 46  
сильне агрегування, 22  
словник мови, 28  
спадкування, 32, 60  
співвідношення «клас-об'єкт», 16  
способи включення об'єктів, 91  
стан, 11, 28, 46, 47, 48  
Стандартна Бібліотека Шаблонів, 94  
статичне зв'язування, 108, 123  
статичний клас пам'яті, 60  
статичний об'єкт, 69  
статичні поля, 61, 179  
статичні методи класу, 179,  
статичні члени класу, 179, 180  
стрілка, 28, 38, 41  
суперклас, 24, 32  
сутність, 28, 37

таблиця віртуальних функцій, 176  
таксон, 23  
таксономічне відношення, 23  
таксономія, 23  
тернарна асоціація, 20  
тестування алгоритмів, 78  
типізовані перетворення, 108  
тривіальні перетворення, 125

узагальнене програмування, 157  
"узагальнення / спеціалізація", 17, 18, 24, 25  
узагальнення, 23, 29, 32, 33  
умова, 48  
уніфікована мова моделювання, 27  
успадкування, 24, 84, 97, 102  
уточнення перевантаження, 124  
участь у діяльності, 49

фізичне включення, 21  
фокус керування, 44  
функціональна форма використання перевантажених операторів, 132  
функції-члени, 61, 64, 128  
функція-оператор, 128

характерні значення, 36

цикл, 46  
циклічність графа, 23  
"ціле / частина", 17, 18, 21, 32  
цільовий стан, 47

шаблон, 157  
шаблони класів, 162  
шаблони функцій, 159, 160

abstract data types, 58  
Action-метод, 64  
Ada, 157  
aggregation, 21  
ARIS Toolset, 27  
association relationship, 17

base class, 85



binary association, 19

catch, 165

CASE-засоби, 27

child class, 85

Class, 59

composition, 22

const, 61, 63, 180

containment by reference, 22

containment by value, 21

CLU, 157

C++, 28, 60, 88, 92, 157

data hiding, 57

delegation, 25

delete, 63

derived class, 85

friend, 60, 68

generalization relationship, 17, 22

generalization / specialization, 17

Get-метод, 64

“has-a”, 98

Hewlett-Packard, 27

inheritance, 84

implementation, 74

inline, 61

interface, 74

“is a”, 17, 24, 26, 98

Java, 28

“like-a”, 98

many- to- many, 19

Microsoft, 27

Microsoft Visual Modeler, 27

Microsoft Visio, 27

map, 94

multimap, 94

multiset, 94

OMG, 27  
Object Management Group, 27  
Object - Oriented Software Engineering, 27  
one-to-one, 19  
Oracle Designer, 27  
overload, 60, 120, 123

Paradigm Plus, 27  
parent class, 85  
part of, 17  
polymorphism, 120  
private, 59, 68  
protected, 59, 60  
public, 59, 60, 68

Rational Software, 27  
resolution, 124

set, 94  
Set-метод, 64  
Silverrun, 27  
Standard Template Library, 94  
static, 60, 63  
STL, 94, 157, 169  
Sybase, 27  
System Architect, 27

taxon, 23  
throw, 165  
try, 165

virtual, 60, 63, 180  
virtualTable, 176  
VisualBasic, 28  
void, 62  
void-конструктор, 149  
volatile, 63

UML, 27, 28, 34, 41, 42  
Unified Modeling Language, 27

whole / part, 17

xor – association, 20



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**

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