УДК 378.147:681.3 **З. С. Сейдаметова**

НАВЧАННЯ ІНЖЕНЕРНО-ТЕХНІЧНИХ ФАХІВЦІВ НА ОСНОВІ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПІДХОДУ

Об'єктно-орієнтоване проектування (ООП) – це сучасний підхід у програмуванні, в основу якого покладено принципи інкапсуляції та приховування інформації, розділення інтерфейсу і реалізацію, класи, наслідування, поліморфізм та ієрархію класів. Деякі дослідники зазначають, що ООП є лише даниною моді, якою вважалася у 80-ті роки ідеологія структурного проектування. На відміну від структурного проектування в ООП відсутня будь-яка формалізація, якщо не враховувати Лисков-принцип [1: 3]. В основному проекти для розробки програмних продуктів, що користуються об'єктно-орієнтованими технологіями, як базу певної частини їх систем використовують структури додатків. Прикладом може слугувати MFC (Microsoft Foundation Classes), який становить структуру для програм з віконним інтерфейсом для IBM-сумісних комп'ютерів. Ця структура покладена в основу архітектури інтерактивних візуальних програм та бібліотеки класів, що підтримують розробку додатків [4: 5].

Виділяють шість основних понять об'єктно-орієнтованого проектування:

Об'єкт. Об'єкт – операційна категорія, що інкапсулює конкретні значення даних та програмні коди, які маніпулюють цими значеннями. Наприклад, інформація про конкретний елемент бази даних та операції, які необхідні для маніпулювання цими даними, формує об'єкт. Об'єкти – це базові категорії в ООП, які становлять сукупність об'єктів, що взаємодіють між собою для розв'язання конкретного завдання. У процесі виконання програми завдяки такій взаємодії об'єкти створюються, зазнають зміни в ході програми, до них здійснюється доступ, вони знищуються. У проекті об'єкт є виявленням певної спеціальної категорії в завданні та у його розв'язанні. Об'єкти можуть мати різні стосунки, які відображають взаємовідношення їх аналогів у проблемній галузі.

Повідомлення. Повідомлення – це запит на виконання певної операції конкретним об'єктом. Якщо розглянути поняття "повідомлення" в розрізі мовних проблем, то в термінології C++ повідомлення кваліфікується як виклик функції-члена, а в середовищах Java і Smalltalk – виклик методів. У повідомленні зазначається ім'я операції, а також можуть бути присутні фактичні параметри, які використовуватимуться при виконанні операції. Сукупність об'єктів вступає у взаємодію шляхом обміну повідомленнями. Об'єкт, що ініціює повідомлення, називається відправником, а об'єкт, який одержує повідомлення – одержувачем. У результаті передачі таких повідомлень з'являються відповіді у вигляді повернених значень або винятків, які пересилаються від того, хто одержує повідомлення до відправника.

Інтерфейс. Інтерфейс являє собою сукупність оголошень, що регламентують поведінку. Для визначення дій, що відносяться до конкретного поняття, шаблони поведінки можуть об'єднуватися в єдину групу. Інтерфейс є будівельним блоком для специфікацій, які визначають набір загальнодоступних поведінок класу.

Клас. Клас – це сукупність об'єктів, які спільно використовують загальну концептуальну базу. Можна розглядати клас як шаблон для побудови об'єктів. Якщо об'єкти утворюють базові елементи для виконання об'єктно-орієнтованих програм, то класи – це базові елементи для визначення об'єктно-орієнтованих програм. Загальна для всіх об'єктів класу концептуальна база складається з двох частин: специфікації класу (представлення того, що може робити кожний клас) і реалізації класу (визначення, як кожний з об'єктів робить те, що може робити).

Наслідування. Наслідування становить відношення між класами, яке дозволяє визначати новий клас, за основу якого взято визначення того класу, що існує. Залежність

одного класу від іншого дозволяє повторно використовувати специфікацію і інтерфейс суперкласу. Наслідування використовується лише для моделювання відношення “є“ або “є вид, вигляд“.

Поліморфізм. Поліморфізм дає можливість розглядати об’єкт як те, що належить до більш ніж одного класу. Розрізняють поліморфізм включення, об’єднання і параметричний поліморфізм. Поліморфізм об’єднання – це об’єднання, входження різних форм в один і той же клас. Завдяки тому, що об’єктно-орієнтовані мови програмування підтримують такий поліморфізм, відправник може використовувати об’єкт як параметр, на базі реалізації його інтерфейсу, а не класу в цілому. Параметричний поліморфізм характеризується здатністю визначати тип у термінах одного або більшої кількості параметрів. Так, в C++ шаблони забезпечують можливість породжувати “новий клас“ під час компіляції, тобто такий клас, для якого замість формального параметра у визначенні підставляється фактичний параметр. Ця властивість дозволяє створювати екземпляри нових класів, які широко використовуються в C++ для створення стандартної бібліотеки шаблонів.

Основою для створення програмного забезпечення є структурні моделі програмних систем. Моделями, що широко використовуються, виступають:

− класична модель, що передбачає наявність трьох базових блоків (блок введення, блок обробки даних, блок виведення даних);

− подіє-орієнтована модель. Програма, що будується на цій моделі, складається з інтерфейсу користувача, диспетчера подій, блоку реакції на події;

− відкрита програмна система, що складається з інтерфейсу користувача, блоку обробки даних і засобів конфігурації;

− програмовані алгоритми керування, управління, основу яких становить віртуальна машина;

− взаємозамінні модулі. Програми, побудовані на взаємозамінних модулях, складаються з алгоритмічної заглушки і реалізації алгоритму;

− інтерфейс програмування. Основу програмних систем, що базуються на цій технології, становлять зовнішні модулі, в яких реалізується керуючий алгоритм;

− змішана модель. Ця модель базується на подіє-орієнтованому підході.

Наведені вище моделі широко використовуються в програмуванні складних систем. Вони дозволяють на етапі проектування програмної системи розширювати її функціональні можливості і тонке налаштування на предметну сферу користувача.

Використовування технологій та моделей ООП дозволяють розробникам програмних продуктів вийти на такий рівень виробництва програмного забезпечення, коли стає можливою робота у великій команді з прогнозуванням термінів розробки, гнучкою керованістю проектами.

Міжнародний стандарт підготовки IT-фахівців CC2001 [6: 7] пропонує навчання IT-спеціальностям на основі об’єктного підходу. Навчання фокусується на програмуванні, а на початку навчання акцентується увага на принципах об’єктно-орієнтованого проектування і програмування. Базовими навчальними дисциплінами, що становлять провідні засади такого підходу, є:

- Введення в об’єктно-орієнтоване програмування.
- Структури даних і аналіз алгоритмів.
- Об’єктно-орієнтоване програмування.
- Об’єктно-орієнтоване проектування і методологія.
- Технологія підготовки програмних продуктів.
- Технологія проектування.

У таблиці 1 пропонується послідовність вивчення базових дисциплін об’єктно-орієнтованого підходу підготовки IT-фахівців:

Таблиця 1.

Послідовність курсів з орієнтацією на ООП

Курс	Осінній семестр	Весняний семестр
1 курс	Введення в об'єктно-орієнтоване програмування	Введення в об'єктно-орієнтоване програмування
2 курс	Структури даних і аналіз алгоритмів	Об'єктно-орієнтоване програмування
3 курс	Об'єктно-орієнтоване проектування і методологія	Технологія підготовки програмних продуктів
4 курс	Технологія проектування	Технологія проектування

Основною метою об'єктного підходу є ознайомлення студентів з:

− основними методами ефективного і високопродуктивного створення програмних систем;

− можливостями програмування комп'ютерних мереж, а також інструментами, за допомогою яких реалізуються такі можливості.

У кінці навчання студенти повинні вміти:

− застосовувати основні принципи створення прикладного програмного забезпечення і інтерфейсів, призначених для користувача;

− застосовувати принципи розробки додатків (у тому числі проектування і розробку різних об'єктів) для створення значного програмного продукту, з обґрунтуванням проектного рішення на кожному етапі та враховувати вимоги до якості ПЗ;

− застосовувати основні методи ефективної та високопродуктивної розробки великих програмних систем;

− використовувати можливості програмування мереж, а також інструменти як технічні, так і програмні, за допомогою яких реалізуються такі можливості;

− застосовувати принципи, пов'язані з проектуванням і розвитком додатків, призначених для роботи у всесвітній мережі;

− систематично оцінювати якість інтерфейсів широкого спектру програмних продуктів.

Для пропаганди такого підходу і його впровадження потрібні молоді, енергійні викладачі з досвідом роботи над крупними проектами. На сьогодні в Україні викладацький корпус, можливо, не повністю готовий до сприйняття запропонованого підходу підготовки ІТ-фахівців.

ЛІТЕРАТУРА:

1. Зубинский А. ООП и UML: коротко и без мифов // Компьютерное обозрение. – № 35, 12. – 18 сентября 2001. – http://itc.ua/7431.
2. Гради Буч. Объектно-ориентированный анализ и проектирование с примерами приложений на C++ – второе издание. Rational Санта-Клара, Калифорния, перевод с английского под редакцией И. Романовского и Ф. Андреева.: СПб, 2001. – 560 с.
3. Пол Ирэ. Объектно-ориентированное программирование с использованием C++: Пер. с англ. – Киев: НИПФ "ДиаСофт Лтд", 1995.
4. Макгрегор Дж., Сайкс Д. Тестирование объектно-ориентированного программного обеспечения. Практическое пособие: Пер. с англ. – К.: ООО "ТИД "ДС", 2002. – 432 с.
5. Брауде Э.Д. Технология разработки программного обеспечения – СПб.: Питер, 2003. – 656 с.
6. Computing Curricula 2001. Computer Science. – Final Report (December 15, 2001), The Joint Task Force on Computing Curricula, IEEE CS, ACM – 236 p. – http://www.computer.org/education/cc2001/.
7. Рекомендации по преподаванию информатики в университетах: Пер. с англ. – СПб., 2002. – 372 с. – http://se.math.spbu.ru/cc2001/cc2001.html.