СИСТЕМИ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ: З ЧОГО РОЗПОЧИНАТИ

Цибулько М. М.

Анотація. Об'єктно-орієнтоване програмування вивчають в освітніх установах різних рівнів. Однак, для викладачів існують певні труднощі під час викладання об'єктно-орієнтованих концепцій початківцям. Як навчати основам об'єктно-орієнтованого програмування на рівні вступного курсу залишається перманентним предметом для дискусії. У статті представлена оцінка цих проблем й обговорюються деякі підходи щодо покращення якості й успішності таких курсів.

Ключові слова: об'єктно-орієнтована мова програмування, процедурне програмування, абстракція, мова UML, алгоритм, середовище програмування.

Об'єктно-орієнтоване програмування є ключовою технологією як у програмній індустрії, так і в освітній галузі. Компанії, що займаються розробленням програмного забезпечення випустили об'єктно-орієнтовані версії своїх продуктів, а освітні установи на кожному рівні включили об'єктно-орієнтоване програмування до своїх навчальних програм. У більшості курсів з програмування закордонних освітніх закладів Паскаль або С було змінено на такі об'єктно-орієнтовані мови як C++, Smalltalk, Eiffel, а дещо пізніше — Java та C Sharp (C#). Основні причини цієї тенденції пов'язані з такими перевагами об'єктно-орієнтованого програмування як: інкапсуляція, наслідування і поліморфізм. З іншого боку, збільшення в масштабі частки використання об'єктно-орієнтованих методів у практиці програмування пов'язане зі зростаючими розмірами і складністю реальних програмних проектів, оскільки об'єктно-орієнтована парадигма програмування дозволяє значно підвищити рівень технології створення програмних засобів, скоротити затрати на розроблення програм, повторне використання програмного коду. Нині широко розповсюджена думка, що об'єктно-орієнтований стиль програмування пропонує кращий інструментарій (засоби) для навчання і розробки програмного забезпечення.

Перехід від попереднього процедурного стилю програмування до об'єктно-орієнтованого, однак, усе ще залишається складним завданням. Більшість викладачів і студентів набувають свій перший досвід програмування, програмуючи в процедурному стилі. Варто зазначити, що оволодіння стилем об'єктно-орієнтованого програмування є значно важчим після звикання до процедурного програмування. Згідно із спостереженнями, для середньостатистичного студента потрібний достатньо довгий період часу, щоб перейти від процедурного до об'єктно-орієнтованого програмування [11]. У перехідний період має місце так званий процес «відучування». Багаторічний досвід викладання програмування свідчить, що старт із процедурного стилю утруднює вивчення об'єктно-орієнтованого програмування на пізніших стадіях [10]. Це наводить на думку, що вивчення об'єктно-орієнтового програмування слід розпочати якнайшвидше, щоб уникнути вищезгаданого звикання до процедурного стилю.

З іншого боку, навчання об'єктно-орієнтованого програмування «з нуля» не є легким завданням. Не існує педагогічно-обґрунтованих методик, які б скеровували викладачів і освітні установи. Навчальні програми з інформатики та методичні рекомендації щодо вивчення програмування в більшості випадків чітко не регламентують використання програмного забезпечення і покладають добір мови програмування на вчителя [1]. Викладачі ВНЗ все ще експериментують різноманітні стилі і підходи, щоб знайти кращі й більш ефективні шляхи пояснення методології і концепції об'єктно-орієнтованих програм. Важливі питання, які потребують вирішення, такі.

1. Яку об'єктно-орієнтовану мову слід використовувати?

2. Якою має бути послідовність вивчення об'єктно-орієнтованих концепцій і методів?

3. Яким має бути навчальне середовище?

4. Коли і наскільки глибоко мають бути включені в курс елементи моделювання і об'єктно-орієнтованого проектування?

Відповіді на ці та подібні запитання значно впливають на структуру і ефективність курсу об'єктно-орієнтованого програмування.

Досвід з викладання багатьох курсів демонструє, що вибір ефективної навчальної методики і ретельне планування курсів допоможуть подолати ці труднощі [10]. Об'єктно-орієнтовані технології програмування базуються в основному на моделюванні деякого реального процесу (об'єкта). Тому природно розпочинати навчання з концептуальних питань: опису й ідентифікування об'єктів. Отже, якщо ми хочемо навчати об'єктно-орієнтованому програмуванню, об'єкти і їх взаємодії завжди повинні бути в центрі уваги. Моделювання об'єктів і їх взаємозв'язків, з іншого боку, може бути реалізоване включенням елементів проектування об'єктів із самого початку курсу. Однак, моделювання, проектування і подання об'єктів у формі програмного коду мають бути інтегровані в курс належним чином. Сучасні об'єктно-орієнтовані мови програмування і засоби для навчання забезпечують ефективні методи й інструментарій, які сприяють такій інтеграції.

Труднощі, пов'язані з переходом від процедурного до об'єктно-орієнтованого програмування. Традиційні процедурні мови програмування здебільшого побудовані на концепції машини з архітектурою фон Неймана. Принципи їх вивчення ґрунтуються на понятті послідовного виконання команд, розгалуження і циклу. У результаті, вивчення таких популярних мов як, наприклад, ФОРТРАН, Паскаль і Basic базується на ос-

нові проектування (конструювання) і виконання загальних алгоритмів і структур даних. Це краще всього виражає знаменита цитата «Алгоритми + Дані = Програма» [2]. Тут головне завдання — навчити студентів програмувати шляхом реалізації алгоритмічних методів. Ця навчальна модель наголошує на необхідності якнайшвидшого вивчення синтаксису й основних структур мови. У разі ж вивчення об'єктно-орієнтованого програмування у вступному курсі, головна увага повинна акцентуватися на особливостях об'єктно-орієнтованої парадигми, замість синтаксичних і структурних деталей мови. Для того щоб уникнути цього несприятливого впливу, рекомендують розпочати вивчення об'єктно-орієнтованого програмування з так званої «чистої» об'єктно-орієнтованої мови [7].

Важливість абстракції і проектування в об'єктно-орієнтованому програмуванні. Об'єктно-орієнтовані програмні продукти базуються на об'єктах і їх взаємодії. У деякому розумінні, об'єктно-орієнтовані концепції замінили модульну структуру описань алгоритмів традиційними мовами програмування. Тому, об'єктно-орієнтовані програми повинні значно відрізнятися від відповідних їм версій на Паскалі або Basic. Основний акцент має робитися на розв'язанні проблеми, шляхом належного моделювання і проектування об'єктів і класів. Так, програма, результатом виконання якої є поява на екрані речення, на зразок «Hello, World!» — певним чином поганий старт, оскільки тут синтаксис домінує над аспектами, орієнтованими на вирішення завдання програмування. Кращий і здебільшого рекомендований підхід — використовувати абстракцію і проектування із самого початку викладання курсу [8]. Хоча, можливо, вважається, що доволі важко навчити абстрактно мислити багатьох студентів, що зараз стало важливим компонентом під час вивчення об'єктно-орієнтованого програмування. Абстракція, з іншого боку, вимагає введення елементів моделювання і проектування навіть у курси початкового рівня.

Навчання основ об'єктно-орієнтованого програмування у ввідних курсах залишається серйозною педагогічною проблемою. Навіть самі загальні поняття є, по суті, складними для розуміння і важко написати завершену програму без їх використання. Візьмемо, наприклад, мінімальний опис головного методу, який потрібний для кожного Java додатку, або простий оператор друкування, який вимагає звернення до System Class. Ці поняття і структури можуть бути засвоєні тільки після значного досвіду програмування мовою Java [9].

Об'єктно-орієнтоване програмування вимагає деяких планувальних зусиль перед фактичним початком процесу написання програми. Оскільки представлення об'єктів і розподілення задач серед них не є, у звичному сенсі, програмуванням. Робота на цій стадії повинна бути сконцентрована на пошук відповідей на такі запитання:

- Яким буде результат виконання програми після її завершення?
- Які класи і об'єкти будуть потрібні?
- За що будуть відповідати ці класи і об'єкти?
- Яким буде інформаційний вміст і функціональне навантаження кожного об'єкта?
- Як об'єкти будуть взаємодіяти між собою (як відбуватиметься обмін даними)?

Вартий уваги і загальновизнаний підхід у цьому відношенні — «Об'єкти спочатку зі зразками проекту» («Objects first with design patterns») [5]. Основними елементами цього підходу є об'єктна ідентифікація, об'єктне моделювання і об'єктне проектування. Кодування рішень повинно базуватися на реалізаціях цієї послідовності. Об'єктно-орієнтоване моделювання і проектування можна полегшити використанням простих інтегрованих графічних засобів за допомогою Уніфікованої Мови Моделювання (UML), яка підтримує широкий набір елементів графічної системи позначень, або навіть простих позначень з використанням знаків на зразок блок-схем і стрілок. У більшості випадків, використання зробленого власноруч схематичного зображення дуже полегшує розуміння взаємозв'язків між об'єктами і класами (рис. 1).

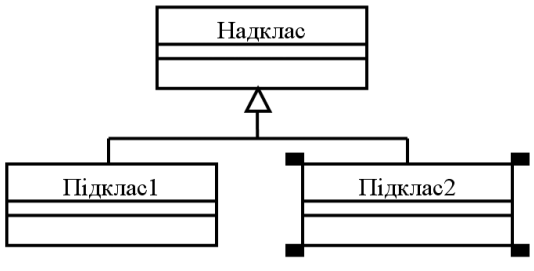

Рис. 1. Приклад зображення наслідування класів у UML

Наголос на моделюванні і проектуванні безпосередньо із самого початку і поєднання кодування (написання) програми з проектуванням є ефективним шляхом навчання об'єктно-орієнтованого програмування [12].

Вибір об'єктно-орієнтованої мови. У даний час C++ і Java — дві найбільш популярні мови для вивчення об'єктно-орієнтованого програмування, C++ здебільшого надають перевагу завдяки її потужності як мови програмування, зворотної сумісності із C і відносної легкості виконання класичних алгоритмів і представлення структур даних. Проте C++ бракує деяких характеристик, якими має володіти мова для вивчення об'єктно-орієнтованого програмування. Серед проблемних галузей — «гібридність» (поєднання процедурного і об'єктно-орієнтованого підходів) мови, занадто складна об'єктна модель і відсутність читабельності [6]. Цих недоліків вдалося уникнути, створюючи нову мову C#, її розробники значно вдосконалили саме синтаксис мови C++. Мова C# досить проста в сприйнятті і надзвичайно зручна як для професійного програмування, так і для навчання його основ. Мова повністю об'єктно-орієнтована. Проте методична підтримка для C# поки що є недостатньою. В Україні не існує жодного рекомендованого МОНМС підручника для її вивчення в школі [1].

З іншого боку, вважається, що Java є «чистою» об'єктно-орієнтованою мовою і не має багатьох суттєвих недоліків C++. Вона є також безумовно самою розповсюдженою мовою для Інтернет-середовищ і мережевих програмних проектів. Ці та інші переваги, зокрема незалежність платформи і відносна простота роблять Java природним вибором для навчальної мови у початкових курсах програмування. Успіх ринку Java є ще одним позитивним чинником для її використання як навчальної мови. М. Коллінг робить висновок, що серед різних альтернатив, Java, здається, є кращим вибором для вивчення об'єктно-орієнтованого програмування [7].

Середовище програмування. Правильно підібране програмне середовище є важливим фактором для ефективності вступного курсу об'єктно-орієнтованого програмування. Традиційні середовища програмування включають текстовий редактор, компілятор мови, редактор зв'язків і бібліотеки функцій. Цього достатньо для простого розроблення прикладної програми. Існує також багато комерційних систем розроблення програм для популярних мов програмування. Використання інтегрованих графічних систем розроблення програм під час вивчення мов програмування стає все більш популярним. Однак ці системи є по суті професійними середовищами розроблення програм і їх відповідність навчальним цілям на вступному етапі є дискусійною. Вони занадто потужні і вдосконалені ніж це потрібно для навчальних цілей і їх багатофункціональність робить їх менш зручними для початкового курсу. Ці види середовищ також критикуються за неспроможність відображати об'єктно-орієнтовані концепції і автоматичне генерування коду. Деякі сучасні середовища можуть автоматично генерувати мало не будь-які фрагменти програм, а то навіть і цілі програми. Автоматичне генерування коду дозволяє, з одного боку, прискорити виконання учнем завдань, продемонструвати ефективність і потужність сучасних засобів програмування. З іншого боку, використання готового коду, до якого слід віднести автоматично генерований код, недостатньо сприяє розумінню механізмів розробки програм.

Як компроміс, було розроблено спеціально спроектовані для вивчення об'єктно-орієнтованих мов середовища. За кордоном популярним прикладом у цьому відношенні є BLUE language (рис. 2). C++ і Java версії BLUE забезпечують дуже просте і легке у використанні середовище для створення і маніпулювання об'єктами і методами. Система заохочує студента розробляти програму в термінах об'єктів, класів і їх взаємодій. Java версія системи BLUEJ зараз доступна як супровідний компонент підручника [4]. Залежно від наступного кроку вивчення об'єктно-орієнтованого програмування, цей вид інтегрованого середовища може бути вибраний як основний засіб навчання.

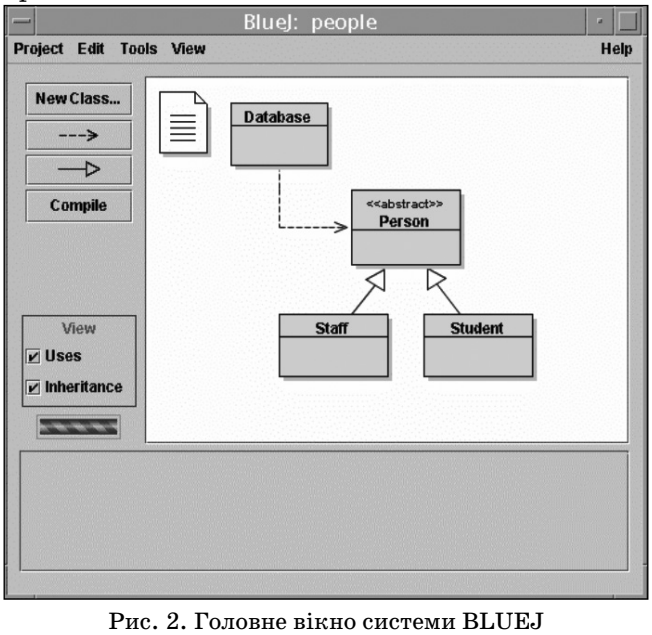

Рис. 2. Головне вікно системи BLUEJ

Серед російських розробок варто відзначити систему програмування Pascalabc.NET (рис. 3). Це мова Object Pascal для платформи Microsoft.NET, що містить всі основні елементи сучасних мов програмування: модулі, класи, передавання операцій, інтерфейси, виключення, узагальнені класи, «збирання» сміття, а також деякі засоби розпаралелювання процесів. Система Pascalabc.NET включає також просте інтегроване середовище, орієнтоване на ефективне навчання програмування і спеціалізовані модулі для навчання. Платформа .NET забезпечує мову Pascalabc.NET величезною кількістю стандартних бібліотек і дозволяє легко поєднувати її з іншими .NET-мовами: C#, Visual Basic.NET, C++, Delphi.NET, Oxygene, Oberon та ін. Крім цього є засоби, що дозволяють створювати компілятори інших мов програмування і вбудовувати їх у середовище за допомогою спеціальних плагінів [3].

Зміст курсу об'єктно-орієнтованого програмування

Іншим важливим фактором, що впливає на успішність вивчення об'єктно-орієнтованого програмування є вибір і порядок тем, які мають бути розглянуті. Те, що навіть найпростіші об'єктно-орієнтовані програми включають дійсно важкі для сприйняття поняття, робить цей вибір особливо важливим. Як ми вже підкреслювали, метою є не тільки вивчення мови програмування, але й інтеграція абстракції й елементів проектування в курс. Певним чином, об'єктно-орієнтоване програмування можна розглядати як побудову і реалізацію абстракцій. Тому головною метою повинно стати подання моделі задачі в термінах об'єктів і їх взаємодії (структурування програми і її проектування через об'єкти і класи), що не виключає попереднього аналізу її у термінах алгоритмічного підходу (метод як складова об'єкта описується як алгоритм). Алгоритмічні деталі і структури даних повинні цілком базуватися на моделюванні і проектуванні.

Структуруючи ввідний курс, потрібно мати на увазі, що об'єктно-орієнтована парадигма об'єднує три концепції: об'єкт, клас і наслідування, що має бути відображено в змісті курсу. Можливий такий пріоритетний перелік тем.

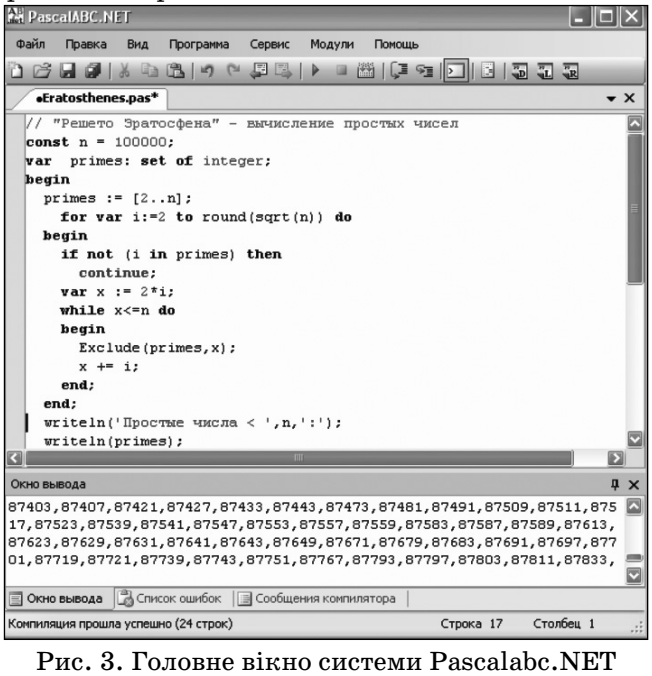

Рис. 3. Головне вікно системи Pascalabc.NET

1. Об'єкт.
2. Клас, підклас, надклас.
3. Дані і змінні.
4. Методи (алгоритмічна складова).
5. Конструктори і метод перезавантаження.
6. Інкапсуляція і модифікатори видимості.
7. Взаємодії між об'єктами.
8. Наслідування, програмна рекурсія.

Отже, ознайомлення з об'єктами і класами має бути представлено із самого початку курсу. Розпочинаючи із самих простих об'єктів і класів, взятих із повсякденного життя, далі мають слідувати основні кроки з моделювання і структурування циклу розроблення програми. Для ілюстрації можуть бути вибрані легкі для сприйняття об'єкти реального світу, такі як людина (person), студент (student), машина (car), будинок (house), школа (school), банк (bank), транспорт (traffic), світло (light) тощо. З проектувальної точки зору, об'єкт може розглядатися як програмний елемент, який наочно сприяє вирішенню проблеми. Після представлення основ об'єктного моделювання і проектування, можна подати формальні означення класів і їх реалізації, використовуючи мінімальну кількість елементів коду. Після цього мають бути представлені базові елементи мови. Основні теми, які мають бути розглянуті, включають: типи даних, змінні, методи виклику процедур і функцій і передавання параметрів.

Далі мають бути розкриті зв'язки між об'єктами і класами з наголосом на використанні бібліотек, пакетів, міток і різних структур даних. Не варто висвітлювати у ввідному курсі повністю поняття наслідування і споріднених концепцій рекурсії і черги. Отже, наголос має бути зроблений на правильний вступ до фундаментальних концепцій, і залишити більш складніші поняття на курси вищого рівня. Потрібно наголосити знову, що у ввідному курсі деталі мови не повинні домінувати, затіняючи фундаментальні об'єктно-орієнтовані концепції.

Висновки. Для покращення якості й ефективності вступного курсу з об'єктно-орієнтованого програмування доцільно використати підхід, який полягає в такому:

- використовувати «чисту» об'єктно-орієнтовану мову;
- розпочинати навчання з понять об'єкта і класу;
- продемонструвати основні поняття, посилаючись на прості об'єкти реального світу;
- включити моделювання і проектування як фундаментальні компоненти на кожній стадії. Використовувати прості графічні позначення;
- не розпочинати написання програми до завершення належного проектування класів і об'єктів;
- постійно слідувати простим моделям проблема-проектування-впровадження (problem-design-implementation);
- викладати пояснення нової проблеми, включаючи максимальну кількість елементів з попереднього завдання;
- уводити поняття об'єктних взаємозв'язків і наслідування якнайраніше, але в розумних межах;
- використовувати прості алгоритми і структури даних, які не вимагають поглиблених знань особливостей мови.

* * *

Цибулько М.М. Системы объектно-ориентированного программирования: с чего начинать

Аннотация. Объектно-ориентированное программирование изучают в различных образовательных учреждениях. Однако, для преподавателей существуют определенные трудности во время преподавания объектно-ориентированных концепций начинающим. Как учить основам объектно-ориентированного программирования на уровне вступительного курса остается перманентным предметом для дискуссии. В статье представлена оценка этих проблем и обсуждаются некоторые подходы для улучшения качества и успешности таких курсов.

Ключевые слова: объектно-ориентированный язык программирования, процедурное программирование, абстракция, язык UML, алгоритм, среда программирования.

* * *

Tsybulko M.M. Object oriented programming systems: a point from which to start

Resume. Object oriented programming is taught at various educational environments. However, teachers usually experience problems when introducing object oriented concepts and programming to beginners. How to teach the fundamentals of object oriented programming at an introductory level course is still a common subject for debate. In the paper, an evaluation of these problems is presented and some possible approaches for improving the quality and success of such courses are discussed.

Key Words: Object oriented language , procedural programming, abstraction, UML, algorithm, programming environment.

Література

1. *Завадський І.О.* Навчальна програма з інформатики для 9–12 класів загальноосвітніх навчальних закладів. Академічний рівень / І. О. Завадський, Ж. В. Потапова, Ю. О. Дорошенко // Інформатика та інформаційні технології в навчальних закладах. — 2008. — №2.
2. *Вирт Н.* Алгоритмы + Структуры данных = Программа. — М.: Мир, 1989.
3. *Михалкович С.С.* Учебная система программирования Pascal-ABC: опыт разработки и использования // Вторая международная научно-практическая конференция «Современные информационные технологии и ИТ-образование»: сборник трудов. — М.: 2006. — С. 394–399.
4. *Barnes D.J, Kцlling M.* Objects First with Java // A Practical Introduction Using BLUEJ. — Prentice Hall, 2008.
5. *Docks van D, Steegmans E.* A new pedagogy for Programming // Position paper for ECOOP // Sixth Workshop on Pedagogies and Tools for Learning Object Oriented Concepts. — Malaga, Spain, 2002.
6. *Kolling M.* The problem of teaching Object-Oriented Programming, Part 1: Languages // Journal of Object-Oriented Programming, 11(8), 1999. — P. 8–15.
7. *Kolling M.* The Problem of teaching Object-Oriented Programming, Part 2: Environments // Journal of Object-Oriented Programming, 11(9), 1999. — P. 6–12.
8. *Nguyen D., Wong S.* OOP in Introductory CS: Better Students through Abstraction // Position paper for OOPSLA // Fifth Workshop on Pedagogies and Tools for Assimilating Object Oriented Concepts. — Umea, Sweden, 2001.
9. *Nino J., Hosch F.*, An Introduction to Programming and Object Oriented Design Using Java. — Wiley, 2001.
10. *Okur M.* Teaching Object-Oriented Programming at the introductory level // Journal of Yasar University, 1(2). — Izmir, Turkiye, 2006. — P. 149–157.
11. *Stroustrup B.* The Design and Evolution of C++. Addison-Wesley Pub Co, 3rd edition, 2000.
12. *Ventura P., Alphonce C.* Teaching OOD and OOP through Java and UML in CS1 and CS2. // Position paper for OOPSLA // Fifth Workshop on Pedagogies and Tools for Assimilating Object Oriented Concepts. — Umea, Sweden, 2001.