УДК 378.011.3-051:004

ПІДГОТОВКА МАЙБУТНІХ УЧИТЕЛІВ ІНФОРМАТИКИ ДО РЕАЛІЗАЦІЇ ОСНОВНИХ ПРИНЦИПІВ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ У НАВЧАЛЬНОМУ ПРОЦЕСІ СУЧАСНОЇ ШКОЛИ

Васенко О.В.
Переяслав-Хмельницький державний педагогічний університет
імені Григорія Сковороди

У статті на основі аналізу діючого шкільного курсу інформатики здійснено спробу узагальнити та охарактеризувати основні принципи об'єктно-орієнтованого програмування, розглянуто особливості їх реалізації в умовах компетентісного підходу до навчання учнів майбутніми вчителями інформатики в процесі їх професійної діяльності.
Ключові слова: принцип навчання, компетентісно-орієнтований підхід, об'єктно-орієнтоване програмування, клас, об'єкт, інкапсуляція, поліморфізм, успадкування.

Постановка проблеми. Сучасний етап розвитку освітньої галузі України характеризується ревізією поглядів на навчально-виховний процес, де ключовим напрямом виступає компетентісно-орієнтований підхід. Це обумовлено, в першу чергу, проблемою підготовки індивіда до життя та діяльності в умовах зростання інформаційного потоку та застосування інформаційно-комунікаційних технологій в найрізноманітніших галузях людської діяльності.

Тому компетентісний підхід у підготовці індивіда до використання інформаційно-комунікаційних технологій як у професійній, так і повсякденній діяльності є актуальною проблемою сьогодення. Особливо, це стосується навчального процесу в школі, де закладаються основи такої підготовки. У таких умовах важливою складовою сучасної шкільної освіти України є навчальний предмет інформатики, пропедевтичний курс якої починає викладатися вже починаючи з 2 класу. Помітний відбиток на це накладає й наявність, практично у кожній сім'ї, комп'ютерної техніки та ранній досвід знайомства з нею дитини, що, природно, вимагає й перегляду самого змісту навчання і відмови від суто «користувацького підходу» на користь розвитку компетенцій, пов'язаних зі створенням власних програмних засобів.

Так, уже у 8 класі рекомендується, при вивченні теми «Основи подійно- та об'єктно-орієнтоване програмування», застосовувати повнофункціональну мову (зокрема, Object Pascal, Visual Basic, Python, Java, C#, C++ тощо) і середовище програмування (Lazarus (мова Object Pascal), Visual Studio (безкоштовна версія Community Edition, мова Visual Basic), IDLE for Python (мова Python) та інші). Учні знайомляться з ключовими поняттями та створюють найпростіші програми з графічним інтерфейсом.

Такий підхід до організації шкільної освіти ставить відповідні вимоги до системи підготовки майбутнього вчителя інформатики для виконання ним своїх професійних обов'язків. Важливою складовою такої підготовки повинна бути реалізація основних принципів об'єктно-орієнтованого програмування й відмова від практики застосування застарілих засобів та методик у сучасних мовах та середовищах програмування, що реалізуються лише у вивченні синтаксису та семантики тієї чи іншої з них. Це дає змогу підвищити фаховий рівень майбутнього вчителя інформатики і, разом з тим, його конкурентоздатність на сучасному ринку праці.

Аналіз останніх досліджень і публікацій. Проблемі підготовки вчителя інформатики до виконання ним професійних обов'язків та організації фахівцем навчально-виховного процесу на основі змісту цього предмету відображено в роботах: В.Ю. Бикова, Ю.В. Горошка, А.П. Єршова, М.І. Жалдака, В.І. Клочка, О.А. Кузнєцова, А.Г. Кушніренка, М.П. Лапчика, В.В. Лапінського, Н.В. Морзе, Ю.С. Рамського, С.А. Ракова, В.Д. Руденко, О.М. Спіріна, Є.К. Хеннера, С.М. Яшанова та багатьох інших.

Крім того, вчені В.Ю. Биков, Л.І. Білоусова, М.І. Жалдак, М.П. Лапчик, Н.В. Морзе, С.А. Раков, Ю.С. Рамський, вказують на необхідність удосконалення наявної методичної системи підготовки вчителя інформатики.

Проблематика застосування об'єктно-орієнтованого програмування в процесі навчання розглядається як у роботах вітчизняних учених: Л.В. Гришка, М.М. Цибулька та ін., так і зарубіжних, зокрема, М. Окур (Mehmet C. OKUR), М. Кьолінг (Michael Kölling) та ін.

На основі проаналізованого масиву даних можна стверджувати, що деякі важливі аспекти, з точки зору підготовки фахівця-інформатики в системі професійної освіти, не отримали належного та об'єктивного висвітлення. У першу чергу це стосується підготовки вчителя інформатики до реалізації основних принципів об'єктно-орієнтованого програмування та узгодження методик його підготовки з рівнем, необхідним для якісного виконання ним своїх професійних обов'язків.

З огляду на це **метою статті** є аналіз, узагальнення та характеристика основних принципів об'єктно-орієнтованого програмування, їх реалізація майбутніми вчителями інформатики в умовах компетентісно-орієнтованого підходу сучасної шкільної освіти.

Виклад основного матеріалу. Крім специфіки змісту сучасного шкільного курсу інформатики на підготовку конкурентоспроможного фахівця-предметника впливають нинішні тенденції й вимоги до створення програмних продуктів. Характерними особливостями останніх є зростання обсягів інформації та накопичення даних, для об-

робки яких потрібно розробляти складніші й технологічніші проекти.

У контексті сказаного варто зазначити, що в остаточному звіті IEEE-CS (Computer Society of the Institute for Electrical and Electronic Engineers) та ACM (Association for Computing Machinery), опублікованому в 2001 р., сформульовано шість моделей навчання програмуванню:

– імперативна модель;
– об'єктна модель;
– функціональна модель;
– модель з охопленням максимальної кількості матеріалу;
– алгоритмічна модель;
– апаратна модель [1].

Ураховуючи все вище виділене, можемо говорити, що найефективніше при підготовці майбутнього вчителя інформатики застосовувати об'єктну модель. В її основу покладено орієнтацію на принципи використання об'єктно-орієнтованих мов програмування та дизайну [4].

За визначенням Р. Буча: об'єктно-орієнтоване програмування – це «методологія програмування, програми, які є сукупностями взаємодіючих об'єктів, кожен з яких – це екземпляр певного класу, що входить в ієрархію класів, пов'язаних відношенням успадкування. У таких програмах класи, зазвичай, вважаються статичними, а об'єкти – мають більш динамічну природу, що підтримує механізми динамічного зв'язку та поліморфізму» [3].

Важливою особливістю об'єктно-орієнтованого програмування є пріоритетність даних над кодом. На відміну процедурного програмування, тут код не керує даними, а дані керують кодом програми.

Ключовими принципами об'єктно-орієнтованих мов є:

– абстракція;
– інкапсуляція;
– наслідування;
– поліморфізм [8].

Для ефективної реалізації цих принципів майбутніми вчителями інформатики необхідною умовою є чітке усвідомлення їх значення та умов застосування. З огляду на це детальніше розглянемо принципи та дамо їх узагальнену характеристику.

В об'єктно-орієнтованому програмуванні під абстракцією даних розуміють присвоєння певному об'єкту характеристик, що відрізнятимуть його від решти об'єктів. Тобто користувачеві надається можливість визначати нові типи даних, котрі функціонально не будуть відрізнятися від базових. Ключовою ідеєю тут є відділення способу використання складених даних від деталей, що їх реалізують, тобто від більш простих об'єктів. Такий підхід є основою об'єктно-орієнтованого програмування і дозволяє працювати з об'єктами не звертаючи уваги на особливості їх реалізації. Також абстракція може бути застосована при роботі з шаблонами, тобто абстрактними типами даних, що залежно від умов їх опрацювання набуватимуть того чи іншого типу [2].

Наступним принципом об'єктно-орієнтованого програмування є *інкапсуляція*. Це властивість мови програмування, що дозволяє користувачу ігнорувати складність реалізації програмного компонента, який використовується й взаємодіяти з ним за допомогою наявного інтерфейсу, а також об'єднати та зберегти необхідні для компонента дані як від зовнішнього впливу так і від помилкового використання [6].

Варто додати, що основою абстракції та інкапсуляції в об'єктно-орієнтованих мовах є клас. Тобто, тип даних, що визначається користувачем та об'єднує структури даних і функції їх обробки. Конкретні змінні цього типу даних називаються екземплярами класу або об'єктами. Програми, що розробляються на основі концепцій об'єктно-орієнтованого програмування, реалізують алгоритми, які описують взаємодію між об'єктами.

Клас містить константи та змінні, що мають назву поля, а також операції та функції, які виконуються над ними. Функції класу називаються методами. Тому поведінку та інтерфейс класу визначають методи, що оперують даними його екземплярів.

Основною метою створення класів – є інкапсуляція складності, тому передбачені механізми для приховування її реалізації в середині класу. Доступ до певних частин класу регулюється за допомогою ключових слів public (відкрита частина) та private (закрита частина) [7].

Методи, розміщені у відкритій частині, вказують на все, що потрібно знати зовнішнім користувачам класу, тобто формують інтерфейс і можуть вільно викликатися через відповідні об'єкти. Методи та поля закритої частини є доступними лише для методів даного класу. Тобто код, що не належить цьому класу не зможе отримати доступ до відмічених таким чином полів та методів [3].

Інкапсуляція підвищує надійність програм, запобігаючи помилковому доступу до полів об'єкта. Також таку програму простіше модифікувати, оскільки при збереженні інтерфейсу класу можна змінювати його реалізацію, що не вплине на зовнішній програмний код.

Наступний принцип – *успадкування* – дає змогу користувачу створювати ієрархію класів, тобто нащадки мають спільні з предками властивості, а також змінювати ці властивості й додавати нові. Оскільки більшою частиною даних ефективно керувати можна лише за допомогою ієрархічної класифікації, то застосування цього принципу на практиці є вкрай важливим. Крім того, у випадку відсутності такої класифікації кожен об'єкт потребував би явного визначення всіх своїх характеристик. Принцип успадкування дає можливість описувати лише ті властивості об'єкта, що роблять його унікальним в межах власного класу, а це зменшує об'єм написаної програми.

Для представлення ієрархії класів використовують деревовидну структуру, де загальні класи розташовуються ближче до кореня, а спеціалізовані формують гілки та листя. У деяких випадках предків називають надкласами або суперкласами, а нащадків підкласами або субкласами [7].

Успадкування тісно пов'язане з інкапсуляцією, тобто у випадку, коли клас інкапсулює деякі атрибути, то будь-який підклас буде мати ті ж самі атрибути, а також атрибути, які він додає як частину своєї спеціалізації. Така концепція не дозволяє без необхідності ускладнювати код. Новий підклас успадковує всі атрибути предків і не отримує непередбачуваних зав'язків з рештою коду в про-

грамі. Якщо об'єкт наслідує свої атрибути від одного базового класу, це буде просте успадкування. У випадку, коли він робить це від кількох предків, то буде множинне успадкування [3].

Принцип, що дає можливість об'єктам з однаковою специфікацією мати різну реалізацію називається – *поліморфізм*. Суть його полягає в тому, що користувач має змогу використовувати в різних класах ієрархії одне ім'я для позначення схожих за змістом дій під час виконання програми. Специфічна дія точно визначається в залежності від конкретної ситуації. Взагалі, суть концепції поліморфізму можна виразити фразою «один інтерфейс, багато методів». Поліморфізм дозволяє створювати абстрактніші програми і збільшити відсоток повторно використаного коду. Спільні властивості об'єктів об'єднують в систему, що може по різному називатися – інтерфейс, клас та мати зовнішнє і внутрішнє вираження. Зовнішня спільність проявляється як однаковий набір методів з однаковими іменами, а внутрішня спільність – одну й ту ж функціональність методів. Можливість прописати різну функціональність одному методу, функції чи операції називається перевантаженням [7].

Як результат, правильно спроектована ієрархія класів є основою для повторного використання коду, що економить час на багаторазовому його переписуванні й тестуванні. Принцип інкапсуляції збільшує надійність та захищеність програми та надає доступ за допомогою public-інтерфейсу класів, а поліморфізм дозволяє створювати зрозумілий та читабельний код [3].

Крім того, методика розробки алгоритму чи програми з використанням основних принципів об'єктно-орієнтованого програмування включає в себе ряд етапів, що виходять за межі традиційного процедурного програмування. Це дозволяє значно підвищити ефективність розв'язування задач учнями у процесі навчання алгоритмізації і основ програмування за рахунок розвитку логічного та алгоритмічного мислення.

Висновки. У результаті проведеного дослідження, з'ясовано, що в процесі підготовки майбутнього вчителя інформатики до забезпечення компетентісно-орієнтованого підходу в навчально-виховному процесі школи доцільно застосовувати об'єктну модель вивчення програмування. Важливою складовою такої підготовки є реалізація ним у своїй професійній та повсякденній діяльності основних принципів об'єктно-орієнтованого програмування (абстракція, інкапсуляція, успадкування, поліморфізм). Такий підхід дозволить розширити інструментарій фахівця-інформатика в процесі викладання розділу курсу алгоритмізації та програмування в школі, а також забезпечить конкурентоздатність та затребуваність педагога на ринку праці.

Список літератури:

1. Computing Curricula 2001 Computer Science [Електронний ресурс]: Final Report (December 15, 2001). Режим доступу.: http://www.acm.org/education/curric_vols/cc2001.pdf.
2. Okur Mehmet C. Teaching object oriented programming at the introductory level Mehmet C. Okur Journal of Yasar University, 2006. – № 1(2).
3. Буч Г. Объектно-ориентированное проектирование с примерами применения / Г. Буч. – М.: Конкорд, 1992. – 512 с.
4. Гришко Л.В. Концептуальні підходи до навчання основ програмування у вищій школі [Електронний ресурс]: Л.В. Гришко Науковий часопис НПУ імені М.П. Драгоманова. Серія № 2. Комп'ютерно-орієнтовані системи навчання: зб. наукових праць / Ред. рада. – К.: НПУ імені М.П. Драгоманова, 2004. – № 1(8). – С. 19-24. Режим доступу до журн.: http://readera.org/article/kontseptualni-pidkhodye-do-navchannja-osnov- 10114277.html.
5. Додаток до листа Міністерства освіти і науки України від 17.08.2016 р. № 1/9-437. [Електронний ресурс]. – режим доступу: http://old.mon.gov.ua/ua/about-ministry/normative/6119.
6. Кравець П.О. Об'єктно-орієнтоване програмування: [навч. посібник] / П.О. Кравець. – Львів: Видавництво Львівської політехніки, 2012. – 624 с.
7. Орлов С.А. Теория и практика языков программирования: [учебник для вузов. Стандарт 3-го поколения] / С.А. Орлов. – СПб.: Питер, 2013. – 688 с.
8. Цибулько М.М. Системи об'єктно-орієнтованого програмування: з чого розпочинати М.М. Цибулько Комп'ютер у школі та сім'ї № 4, 2011. – С. 11-14.

Васенко А.В.
Переяслав-Хмельницкий государственный педагогический университет
имени Григория Сковороды

ПОДГОТОВКА БУДУЩИХ УЧИТЕЛЕЙ ИНФОРМАТИКИ К РЕАЛИЗАЦИИ ОСНОВНЫХ ПРИНЦИПОВ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ В УЧЕБНОМ ПРОЦЕССЕ СОВРЕМЕННОЙ ШКОЛЫ

Аннотация
В статье на основе анализа действующего школьного курса информатики предпринята попытка обобщить и охарактеризовать основные принципы объектно-ориентированного программирования, рассмотрены особенности их реализации в условиях компетентностного подхода к обучению учащихся будущими учителями информатики в процессе их профессиональной деятельности.

Ключевые слова: принцип обучения, компетентностно-ориентированный подход, объектно-ориентированное программирование, класс, объект, инкапсуляция, полиморфизм, наследование.

Vasenko O.V.
Pereiaslav-Khmelnytskyi Hryhorii Skovoroda State Pedagogical University

TRAINING OF FUTURE COMPUTER SCIENCE TEACHERS FOR THE IMPLEMENTATION OF THE BASIC PRINCIPLES OF OBJECT-ORIENTED PROGRAMMING IN THE EDUCATIONAL PROCESS OF MODERN SCHOOL

Summary
In the article on the basis of the analysis of the current school course of informatics an attempt was made to generalize and characterize the basic principles of object-oriented programming, the feature of their realization in the conditions of the competent approach to the learning of students by future teachers of informatics in the process of their professional activity are considered.
Keywords: training principle, competence-oriented approach, object-oriented programming, class, object, encapsulation, polymorphism, inheritance.