УДК 378.14

Конюхов С. Л.

ПРОЕКТУВАННЯ ЗМІСТУ НАВЧАННЯ ОБ'ЄКТНО-ОРІЄНТОВАНОМУ ПРОГРАМУВАННЮ МАЙБУТНІХ ІНЖЕНЕРІВ-ПРОГРАМІСТІВ

Зміст підготовки майбутніх інженерів-програмістів у закладах вищої освіти має сприяти формуванню у студентів професійних компетентностей. Підґрунтям для формування такого змісту є парадигми програмування, які забезпечують його усталеність в умовах постійного оновлення технологій розробки. У межах дослідження розглядається проблема удосконалення змісту навчання студентів об'єктно-орієнтованій парадигмі. Для її вирішення пропонується формування наскрізної змістовної лінії вивчення об'єктно-орієнтованої парадигми у складі дисципліни «Програмування», а також низки інших дисциплін циклу професійної підготовки. Для проектування змісту використовується метод якісного контент-аналізу джерел, присвячених об'єктно-орієнтованій парадигмі. За результатами аналізу визначена послідовність вивчення об'єктно-орієнтованих засобів мов програмування, а також методів об'єктно-орієнтованого аналізу і проектування. Результатом дотримання цієї наскрізної лінії є формування у студентів компетентностей у сфері об'єктно-орієнтованої розробки.

Ключові слова: *професійна підготовка, майбутній інженер-програміст, заклад вищої освіти, об'єктно-орієнтоване програмування, об'єктно-орієнтований аналіз, об'єктно-орієнтоване проектування, мова програмування, зміст навчання.*

Важливим етапом розробки освітніх програм професійної підготовки фахівців у закладах вищої освіти (далі – ЗВО) є визначення її змісту. Складність цього завдання стосовно підготовки майбутніх інженерів-програмістів пов'язана з постійним оновленням технологій, які використовуються в індустрії програмного забезпечення, оскільки очікується, що випускник ЗВО повинен володіти новими технологіями або принаймні швидко вивчати їх. Багато новацій спираються на фундаментальні концепції, наприклад, на положення об'єктно-орієнтованої парадигми (далі – ООП). Це дозволяє стверджувати, що здатність випускника до розуміння і застосування базових концепцій є необхідною для подальшого самостійного опанування нових інструментів програмної розробки.

У процесі проектування змісту навчання майбутніх інженерів-програмістів об'єктно-орієнтованої розробки викладач має вирішити, які саме питання підлягають розгляду, на прикладі якої мови програмування слід здійснювати вивчення, які додаткові засоби доцільно використовувати. Вирішуючи це завдання, доцільно орієнтуватися на рекомендації «Керівництва з викладання програм для бакалавріату у галузі комп'ютерних наук» [6]. Зокрема, під час вивчення курсу «Об'єктно-орієнтоване програмування» у студентів необхідно сформувати знання основних концепцій і понять ООП (об'єктна декомпозиція, класи (поля, методи, конструктори), ієрархія класів, успадкування, інкапсуляція, поліморфізм), а також здатності до їх практичного застосування [6, с. 157]. Під час вивчення курсу «Розробка/проектування програмного забезпечення» мають розглядатися такі питання, як: приховування інформації, повторне використання стандартних структур, об'єктно-орієнтований аналіз і проектування; також повинна формуватися здатність до розуміння і застосування положень об'єктно-орієнтованого підходу у процесі розробки програмного забезпечення [6, 180]. Окрім того, початковий курс програмування також може бути побудований із використанням ООП [6, с. 28].

Враховуючи зазначене, робимо висновок про доцільність застосування ООП у якості наскрізної змістовної лінії, яка пов'язує між собою окремі розділи курсу програмування, а також інші дисципліни циклу професійної підготовки бакалаврів спеціальності «Комп'ютерні науки», утворюючи один із рівнів каркасу освітньої програми майбутніх інженерів-програмістів. Відсутність такої наскрізної лінії призводить до того, що студенти вивчають концепції ООП в окремому курсі, потім зрідка використовують їх у наступних курсах і наприкінці навчання не мають здатності до свідомого застосування потужних засобів цієї парадигми.

Теоретико-методологічне підґрунтя досліджень із проблем професійної підготовки майбутніх інженерів-програмістів у ЗВО утворюють роботи, присвячені загальним аспектам вищої професійної освіти (А. Алексюк, І. Зязюн, А. Конох, Н. Ничкало, С. Сисоєва й ін.), компетентнісному підходу у вищій школі (М. Єлькін, Ю. Рашкевич й ін.), проблемам методології інформатики й інформатизації навчального процесу у ЗВО (В. Биков, А. Гуржій, М. Жалдак, М. Львов, Ю. Рамський, О. Співаковський та ін.).

У дослідженнях із проблем професійної підготовки майбутніх інженерів-програмістів та ІТ-фахівців у ЗВО розкриті її теоретико-методологічні засади (Л. Гришко, В. Круглик, В. Осадчий, З. Сейдаметова, С. Семеріков та ін.), впровадження компетентнісного підходу (М. Вінник, Л. Зубик, В. Седов, А. Стрюк та ін.), окремі аспекти вивчення об'єктно-орієнтованого програмування (О. Баранюк, І. Барков, Т. Вакалюк, В. Єремєєв, С. Жуковський, Л. Калініна, Г. Рудакова та ін.). Водночас у наявних дослідженнях об'єктно-орієнтований підхід здебільшого розглядається у межах окремих дисциплін, тому актуальним є вивчення можливостей зазначеної парадигми як з'єднуючої ланки окремих курсів програмування і розробки програмного забезпечення.

Мета статті – проектування наскрізного змісту навчання об'єктно-орієнтованого програмування майбутніх інженерів-програмістів у ЗВО.

Один із варіантів побудови освітньої програми підготовки майбутніх інженерів-програмістів передбачає вивчення дисципліни «Програмування» протягом усього періоду навчання, а також низки інших професійно-орієнтованих дисциплін («Програмування та підтримка веб-застосувань», «Технології проектування комп’ютерних систем» й ін.). У зв’язку з цим постає завдання здійснити обґрунтований розподіл навчального матеріалу для забезпечення поступового формування професійних компетентностей у сфері об’єктно-орієнтованої розробки: від розуміння базових понять до здатності пояснити фундаментальні концепції ООП і далі до здатності вільно використовувати їх у практичній діяльності.

З метою визначення найбільш значущих питань, які підлягатимуть розгляду під час вивчення ООП, нами був проведений якісний контент-аналіз змісту літературних джерел. У якості сукупності джерел було обрано книги, підручники і методичні посібники вітчизняних і закордонних авторів, а у якості елементів аналізу – основні концепції (абстракція, клас, інкапсуляція, поліморфізм, успадкування) і методи (програмування, аналіз, проектування) об’єктно-орієнтованого підходу. У процесі аналізу було вивчено зміст 50 джерел, що надало можливість спроектувати послідовність розкриття концепцій і методів ООП у процесі професійної підготовки майбутніх інженерів-програмістів у ЗВО.

Розпочинати проектування наскрізної змістовної лінії вивчення ООП у ЗВО доцільно, спираючись на роботу Г. Буча «Об’єктно-орієнтований аналіз і проектування». Аналізуючи витоки, зміст і застосування ООП, автор подає визначення трьох основних методів ООП (програмування, аналіз і проектування) і пояснює зв’язок між ними [1]. Отже, професійна підготовка майбутніх інженерів-програмістів у ЗВО за бакалаврськими програмами має включати вивчення усіх названих методів у їх взаємозв’язку.

У першій частині роботи [1] викладені концептуальні основи ООП, тому її можна рекомендувати для формування змісту вступного курсу об’єктно-орієнтованого програмування. У другій і третій частинах розглядається процес і прагматика об’єктно-орієнтованого аналізу і проектування, а також вплив об’єктної моделі на керування реальними процесами розробки. Цей матеріал доцільно використовувати під час формування змісту відповідних курсів, наприклад «Об’єктно-орієнтований аналіз і проектування», «Технології проектування комп’ютерних систем», «Управління програмними проектами» тощо.

Подальше опанування мов програмування має спиратися на компетентності, сформовані у процесі вивчення концепцій ООП. Забезпечити це можливо, якщо у змісті відповідних курсів робити наголос на використанні методів об’єктно-орієнтованого підходу. Далі розглянемо джерела, які доцільно застосовувати для формування такого змісту.

В індустрії розробки програмного забезпечення нині широко використовується об’єктно-орієнтована, кросплатформна мова програмування Java. У зв’язку з цим однією з тенденцій вищої ІТ-освіти є використання Java в якості першої мови програмування або основної мови для вивчення власне об’єктно-орієнтованого програмування. На нашу думку механізми ООП найбільш повно реалізовані у мові C++ [2], тому саме її доцільно використовувати для демонстрації концепцій і методів об’єктно-орієнтованого підходу. Вивчення мови Java є обов’язковим для професійної підготовки майбутніх інженерів-програмістів у ЗВО.

Як зазначає Б. Еккель, Java, як і C++, є гібридною мовою, яка підтримує різні стилі програмування, але вона є більш «чистою» об’єктно-орієнтованою мовою, тобто програмування на Java передбачає повне «занурення» до об’єктно-орієнтованих концепцій [5, с. 85]. Отже, для формування гарного стилю програмування на Java студенти повинні розуміти принципи ООП.

У процесі визначення змісту навчання майбутніх інженерів-програмістів об’єктно-орієнтованого програмування мовою Java можна спиратися на фундаментальну роботу Б. Еккеля «Thinking in Java» [5]. Автор дотримується такого порядку викладення основних концепцій ООП: вступ до ООП, ініціалізація і завершення (конструктори, перевантаження методів, фіналізація, збирання сміття), керування доступом (інтерфейс, реалізація, доступ до класів), повторне використання класів (композиція, успадкування, делегування, перетворення типів), поліморфізм, інтерфейси (абстрактні класи і методи, інтерфейси, відокремлення інтерфейсу від реалізації, розширення інтерфейсу), аналіз і проектування.

Інший підхід до вивчення Java, більш наближений до формування у студентів практичних умінь і навичок, пропонує Р. Печіновський. Його робота «OOP – Learn Object Oriented Thinking and Programming» [7] є гарним методичним посібником для викладачів, які планують зробити акцент на механізмах ООП. Автор подає послідовність уроків, спрямованих на формування у студентів здатності до розробки програм мовою Java. У першій частині книги він розглядає усі основні концепції і механізми ООП і пропонує для виконання невеликі за обсягом навчальні вправи. Це зроблено з метою створення у студентів загальної картини ООП. Друга частина книги містить матеріал, який розширює і доповнює зміст першої частини і спрямований на формування практичних умінь програмування. У третій частині подається поглиблений курс ООП на Java, для засвоєння якого пропонується розробка завершеного проекту.

Дидактичний підхід Р. Печіновського «Спочатку архітектура» передбачає, що студенти з самого початку ознайомлюються з основними архітектурними принципами ООП і відразу переходять до їх практичної реалізації [7, с. 3]. Це дозволяє сформувати загальне уявлення про ООП і початкові навички програмування на Java, які вдосконалюються надалі. Навчальний матеріал подається у формі діалогу між автором і читачем: Р. Печіновський формулює питання, які виникають у студента, і відповідає на них. Викладачі можуть використовувати наведені питання у своїй практиці або, спираючись на них, формулювати власні питання і заохочувати студентів до активної співпраці.

Загалом, теоретичний матеріал і практичні завдання, наведені у перших двох частинах книги [7], можна використовувати в якості основи для побудови вступного курсу Java. Третя частина може бути запропонована для розробки поглибленого спецкурсу або використана в основному курсі, якщо дозволить рівень базової підготовки студентів.

Професійна підготовка майбутніх інженерів-програмістів у ЗВО також часто включає вивчення ще однієї промислової об'єктно-орієнтованої мови програмування – C#. Опанування навичок розробки програм цією мовою також має спиратися на знання концепцій ООП, розширюючи і поглиблюючи їх. Визначити зміст цієї частини курсу програмування можна, спираючись на роботу Д. Кларка [4], яка містить три змістові розділи. У першому викладено і проілюстровано за допомогою UML-діаграм концепції ООП. Другий розділ присвячений практичним аспектам реалізації ООП під час програмування мовою C#. У третьому розділі наведені відомості про розробку застосувань Net мовою C#.

Викладаючи методологію ООП, Д. Кларк дотримується такої послідовності: загальний огляд ООП (історія ООП, основні концепції, історія C#); визначення структури класів під час проектування програмних рішень; моделювання взаємодії об'єктів у процесі проектування програмних рішень; практичний приклад проектування структури класів [4]. Робота Д. Кларка є джерелом інформації для побудови курсу програмування мовою C#. Докладний опис концепцій ООП буде корисним, якщо мову C# обрано основною для вивчення об'єктно-орієнтованого підходу. Якщо ж вивчення курсу C# ґрунтується на знаннях ООП, то основну увагу можна приділити практичній реалізації об'єктно-орієнтованої розробки засобами C#.

Далі розглянемо вивчення об'єктно-орієнтованих механізмів мови програмування JavaScript. Майбутні інженери-програмісти зазвичай вивчають JavaScript у курсах, присвячених web-розробці. Засоби ООП розглядають побіжно, здебільшого у процесі ознайомлення з об'єктною моделлю документа (DOM). Однак, враховуючи сучасний стан цієї мови й перспективи її розвитку, вважаємо необхідним посилити акцент на об'єктно-орієнтовані механізми JavaScript.

Об'єктно-орієнтовані можливості JavaScript відрізняються від можливостей класичних мов (C++, Java). Наприклад, у JavaScript усе є об'єктом, і використовується dot-нотація для доступу до властивостей і мето-

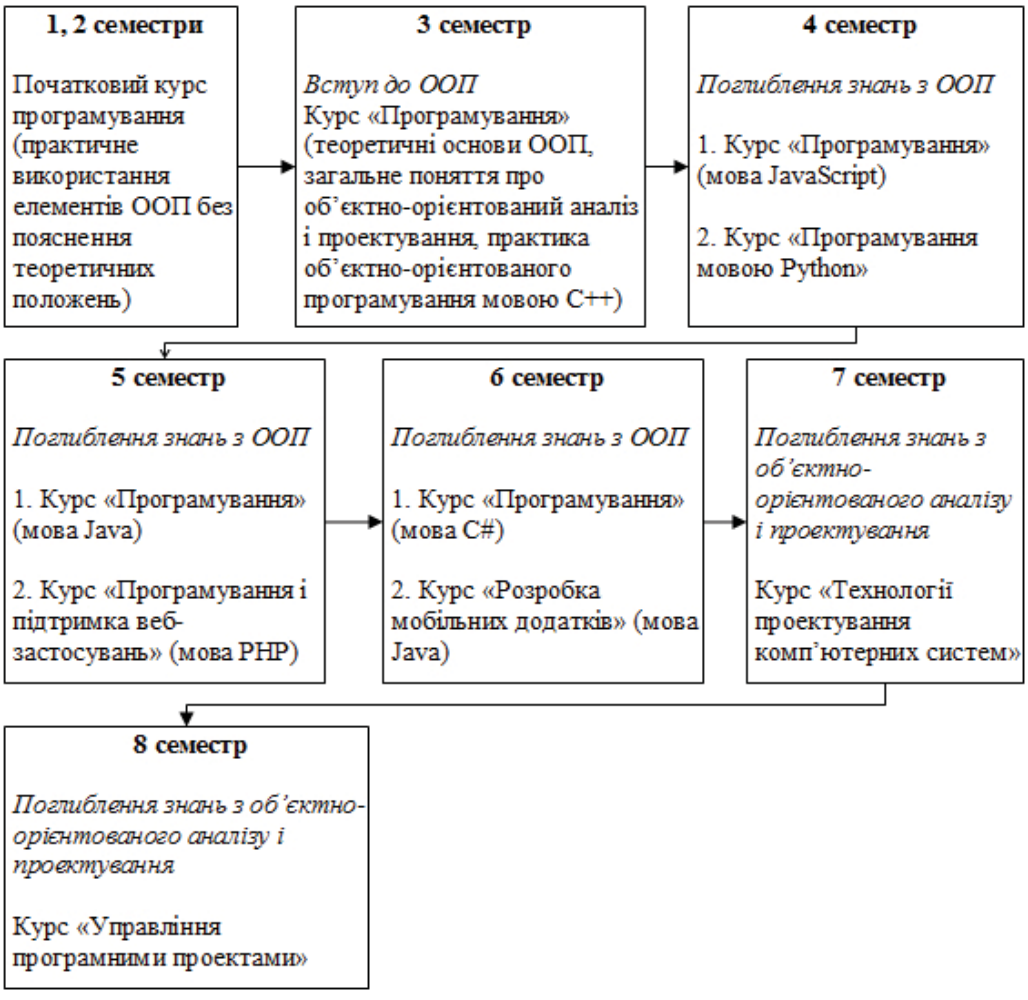

Рис. 1. Послідовність вивчення курсів із програмування

дів об’єктів, але не потрібно описувати класи [8, с. XVIII]. У зв’язку з цим вивчення ООП у JavaScript потребує від студентів здатності змінювати кут погляду на теоретичні поняття, аналізувати і порівнювати їх реалізацію у різних мовах програмування.

З метою визначення змісту навчального матеріалу для курсу програмування мовою JavaScript доцільно звернутися до робіт [3] і [8]. У роботі Дж. Резіга, Р. Фергюсона і Дж. Пакстона [3] об’єктно-орієнтовані властивості JavaScript докладно розглядаються у главах «Створення повторно використовуваного коду», «Об’єктна модель документів», «Події». Книга супроводжується прикладами коду, і запропонованого матеріалу загалом достатньо для висвітлення засобів ООП у JavaScript.

Н. Закас у роботі [8] розглядає реалізацію ООП у JavaScript. У шести розділах (Примітивні і посилальні типи; Функції, Розуміння об’єктів; Конструктори і прототипи; Успадкування; Шаблони об’єктів) автор докладно описує об’єктно-орієнтовані механізми JavaScript, супроводжуючи їх прикладами коду. Загалом, незважаючи на порівняно невеликий обсяг, робота Н. Закаса є гарною допомогою викладачам і студентам під час вивчення JavaScript.

Отже, на нашу думку, вивчати мову JavaScript слід після того, як студенти засвоїли принаймні базові поняття ООП, щоб зосередитися на її специфіці, поглибити розуміння концепцій ООП, а також сформувати розуміння відмінностей їх реалізації.

Освітні програми професійної підготовки майбутніх інженерів-програмістів у ЗВО можуть передбачати вивчення й інших мов програмування, наприклад: PHP, Python тощо. Під час викладання цих курсів також слід зробити акцент на об’єктно-орієнтованих засобах цих мов.

На основі проведеного аналізу можна запропонувати послідовність вивчення майбутніми інженерами-програмістами курсів, яка забезпечує дотримання наскрізної змістовної лінії ООП і формування у студентів компетентностей у галузі об’єктно-орієнтованої розробки (рис. 1.).

Згідно з наведеною схемою, у третьому семестрі студенти повинні вивчити основні концепції ООП (клас, об’єкт, інкапсуляція, успадкування, поліморфізм), оволодіти здатностями до їх практичного застосування мовою C++, а також отримати початкові відомості про об’єктно-орієнтований аналіз і проектування. Під час вивчення наступних курсів у 4–6 семестрах слід робити акцент і забезпечувати практику об’єктно-орієнтованого програмування іншими мовами. Систематизувати знання й уміння у сфері об’єктно-орієнтованого аналізу і проектування слід під час вивчення таких дисциплін, як «Технології проектування комп’ютерних систем», «Управління програмними проектами» у 7–8 семестрах.

Висновки. Одним зі шляхів удосконалення підготовки майбутніх інженерів-програмістів у ЗВО є проектування її змісту, який відповідає сучасному стану галузі, сприяє формуванню у студентів професійних компетентностей, утворює основу для подальшого професійного зростання. З цією метою у межах нашого дослідження пропонується розробка наскрізного змісту навчання майбутніх інженерів-програмістів об’єктно-орієнтованої розробки програмного забезпечення. Для її досягнення був проведений якісний контент-аналіз джерел, присвячених ООП, й обрані роботи, на основі яких можна забезпечити послідовне викладення навчального матеріалу. За результатами проведеного аналізу запропонована послідовність вивчення майбутніми інженерами-програмістами ООП у курсах програмування. Наскрізна змістовна лінія передбачає, що, починаючи з третього семестру, відбувається формування у студентів професійних компетентностей у сфері об’єктно-орієнтованої розробки за рахунок систематичного використання засобів ООП, реалізованих у різних мовах програмування. Подальші розвідки доцільно спрямувати на розробку змісту, форм, методів і засобів вивчення окремих елементів наскрізної лінії об’єктно-орієнтованого програмування.

Використана література:

1. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений / [Г. Буч, Р. А. Максимчук, М. У. Энгл, Б. Дж. Янг, Д. Коналлен, К. А. Хьюстон]. – Москва : ООО «И.Д. Вильямс», 2008. – 720 с.
2. Конюхов С. Л. До питання вибору мови програмування для вивчення дисципліни «Об’єктно-орієнтоване програмування» в університетах / С. Л. Конюхов // Інформаційні технології в освіті та науці : зб. наук. праць. – Мелітополь : Вид-во МДПУ ім. Б. Хмельницького, 2017. – № 1 (9). – С. 128–132.
3. Резиг Дж. JavaScript для профессионалов / Дж. Резиг, Р. Фергюсон, Дж. Пакстон. – Москва : ООО «И.Д. Вильямс», 2016. – 240 с.
4. Clark D. Beginning C# Object-Oriented Programming / D. Clark. – New York : Apress, 2011. – 362 p.
5. Eckel B. Thinking in Java / B. Eckel // Upper Saddle River: Prentice Hall, 2003. – 1151 p.
6. Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society. Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. – New York : ACM, 2013. – 518 p. – [Електронний ресурс]. – DOI : 10.1145/2534860.
7. Pecinovský R. OOP – Learn Object Oriented Thinking and Programming / R. Pecinovský. – Řepín–Živonín : Tomáš Bruckner, 2013. – 527 p.
8. Zakas N. C. The principles of object-oriented JavaScript. San Francisco : No Starch Press, Inc, 2014. – 122 p.

References:

1. Buch G., Maksimchuk R. A., Ehngl M. U., Yang B. Dzh., Konallen D., Hyuston K. A. (2008). Object-oriented analysis and design with applications. Moscow: OOO «I.D. Vilyams». 720 p. [in Russian]
2. Koniukhov S. L. (2017). On the choice of a programming language for studying the discipline "Object-oriented programming" in universities. In Informatsiini tekhnolohii v osviti ta nautsi: zb. nauk. prats, 1 (9), pp. 128–132. Melitopol: Vyd-vo MDPU im. B. Khmelnytskoho. [in Ukrainian]
3. Resig J., Ferguson R., Paxton J. (2016). Pro JavaScript Techniques. Moscow : OOO «I.D. Vilyams». 240 p. [in Russian].
4. Clark D. (2011). Beginning C# Object-Oriented Programming. New York: Apress. 362 p. [in English]
5. Eckel B. (2003). Thinking in Java. Upper Saddle River: Prentice Hall. 1151 p. [in English]
6. Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society. Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. – New York: ACM, 2013. – 518 p. – DOI: 10.1145/2534860. [in English]
7. Pecinovský R. (2013). OOP – Learn Object Oriented Thinking and Programming. Řepín–Živonín: Tomáš Bruckner. 527 p. [in English]
8. Zakas N. C. (2014). The principles of object-oriented JavaScript. San Francisco: No Starch Press, Inc. 122 p. [in English]

Конюхов С. Л. Проектирование содержания обучения объектно-ориентированному программированию будущих инженеров-программистов

Содержание подготовки будущих инженеров-программистов в учреждениях высшего образования должно способствовать формированию у них профессиональных компетентностей. Основой для формирования такого содержания являются парадигмы программирования, которые обеспечивают его устойчивость в условиях постоянного обновления технологий разработки. В данной работе рассматривается проблема совершенствования содержания обучения студентов объектно-ориентированной парадигме. Предлагается формирование сквозной содержательной линии изучения объектно-ориентированной парадигмы. Для проектирования содержания используется метод качественного контент-анализа источников, посвященных объектно-ориентированной парадигме. Определена последовательность изучения объектно-ориентированных средств языков программирования, а также методов объектно-ориентированного анализа и проектирования. Результатом соблюдения этой сквозной линии является формирование у студентов компетентностей в сфере объектно-ориентированной разработки.

Ключевые слова: *профессиональная подготовка, будущий инженер-программист, высшее учебное заведение, объектно-ориентированное программирование, объектно-ориентированный анализ, объектно-ориентированное проектирование, язык программирования, содержание обучения.*

Koniukhov S. L. Designing the content of future software engineers training in object-oriented programming

An educational content of training of future software engineers at universities should aid the formation of their professional competencies. The programming paradigms are the basis for the creation of such content since ensuring its sustainability in circumstances of software development technologies updating. In this paper, we examine the problem of improving the educational content in the field of learning of object-oriented paradigm. We offer to form a cross-cutting content line for learning object-oriented paradigm as a part of the "Programming" course, as well as a number of other curricular courses. To design the educational content we use the method of qualitative content analysis of sources devoted to object-oriented paradigm. On this basis, we define a sequence of studying object-oriented programming, analysis, and design. The result of compliance with this cross-cutting line is the formation of students' competence in the field of object-oriented development.

Key words: *professional training, future software engineer, higher education institution, object-oriented programming, object-oriented analysis, object-oriented design, programming language, educational content.*

УДК 378.14.015.62:504

Коренева І. М.

КОМПЕТЕНТНОСТІ ВЧИТЕЛЯ БІОЛОГІЇ: ПОГЛЯД КРІЗЬ ОСВІТУ ДЛЯ СТАЛОГО РОЗВИТКУ

У статті здійснено аналіз проекту стандарту вищої освіти зі спеціальності 014.05 «Середня освіта (Біологія)» з погляду його можливостей щодо формування у майбутніх вчителів біології компетентностей з освіти для сталого розвитку. Встановлено, що комплекс загальних компетентностей майбутнього вчителя біології можна вважати основою для формування загальної культури особистості на цінностях сталого розвитку й основою для формування професійної компетентності, зокрема здатності реалізовувати функції освіти для сталого розвитку в процесі своєї професійної діяльності. Серед спеціальних (предметних) компетентностей стандартом передбачено таку: «Здатність у процесі навчання та виховання учнів розуміти й реалізовувати стратегію сталого розвитку людства». Саме ця компетентність відображає контент сталого розвитку та спрямовує майбутню підготовку вчителів біології на засади освіти для сталого розвитку, забезпечує просування освіти для сталого розвитку у професійній підготовці вчителя біології. В цілому аналізований проект стандарту орієнтований на освіту для сталого розвитку. Проте він потребує доповнення в частині програмних результатів навчання, які здатні оцінити рівень підготовки майбутніх вчителів біології до реалізації функцій освіти для сталого розвитку та наскрізної лінії «Екологічна безпека та сталий розвиток».

Ключові слова: *освіта для сталого розвитку, стандарт вищої освіти, підготовка вчителів біології, компетентності вчителя біології.*