УДК 373.5.091.313:53:004.4

Ворошилов В.В., Бєлошапка О.Я.

[1] *здобувач другого (магістерського) рівня вищої освіти за ОП «Середня освіта (Фізика)», фізико-математичний факультет ДВНЗ «ДДПУ»*
e-mail: v.v.voroshylov@gmail.com, ORCID 0009-0009-7535-2880
[2] *старший викладач кафедри фізики, ДВНЗ «ДДПУ»*
e-mail: beregslav2015@gmail.com, ORCID 0000-0001-7448-3832

ІНТЕГРАЦІЯ ІНФОРМАТИКИ ДО НАВЧАННЯ ФІЗИКИ

Стаття присвячена висвітленню користі зв'язку програмування та фізики для учнів. Обґрунтовано вибір початкового предмету та мови програмування для даної ідеї синергії предметів. В статі надано приклад теоретичного та практичного матеріалів для міжпредметного зв'язку – дві задачі різних тем та різного рівня складності, лістинг коду на мові програмування Java який обчислює дві надані задачі та теоретичний матеріал для розуміння і пояснення коду.

Ключові слова: *навчальний процес, релевантність, програмування, java, розв'язання задач.*

DOI: https://doi.org/...

Вступ

Постановка проблеми. Не для кого не є таємницею, що інформаційні технології все більше з'являються у повсякденному житті, тому школа зобов'язана робити все можливе, щоб надати учням релевантну освіту і крок до цього – поєднати два предмети і зробити так, щоб учні одночасно змогли навчатися і математично направленому предмету і програмуванню.

Було обрано фізику – як не легший предмет. За допомогою цього можна буде продемонструвати більше матеріалу учням стосовно програмування. Також буває таке, що вчитель не встигає приділяти увагу всім учням в класі, і коли клас вирішує задачу не простішого рівня, то учні з поганою успішністю або які просто не зрозуміли поточну тему уроку, не готові йти на рівні з іншим класом, просто виключаються з навчального процесу, починають займатися своїми справами, така особливість людського мозку. І розв'язання задачі з фізики за допомогою програмування дає шанс, що такі учні знову будуть залучені у навчальний процес, бо програмування в деякій мірі творчий процес, що приверне увагу всіх, незалежно від рівня складності задачі з фізики.

У якості мови програмування було обрано Java, тому, що:

По-перше вона комерційна, тобто на ній відбувається розробка програмних продуктів різного застосування, а не так, як наприклад Pascal який використовується тільки для навчання чи Volfram який використовується тільки вченими. Так у учня з'явиться уявлення, що він це вчить не просто, щоб вчити та отримувати оцінки або схвалення від вчителя і ніколи йому це не знадобиться, а може використати в своїх цілях. До того ж зараз у програмістів склався приємний та авторитетний імідж, що ще більше приверне увагу до даного процесу збоку учнів.

По-друге Java одна з най поширених мов, яка вже існує досить давно, що спростить пошук матеріалу як при підготовці уроку викладачем так і при виконанні завдань учнями, так як Java-спільнота дуже велика та активна.

Основна частина

Як відбуватиметься процес навчання – спочатку йде звичайне вирішення задачі, і потім, коли задача вирішена переходимо до програмування. Спочатку треба розповісти учням теорію та першу задачу робити з ними разом, розповісти все покроково. Другу задачу надати більш складну, знову ж вирішити класичним способом і потім вже учні повинні самі спробувати написати програму, яка буде вирішувати цю задачу.

Перейдемо до прикладів. Було обрано 2 задачі різного рівня та різних тем, щоб з'явилося розуміння який це має вигляд. Задачу 1 було взято із збірника завдань для тематичного контролю з фізики, 8 клас. Задачу 2 було взято з інтернет джерела – Освітній проект «На Урок» для вчителів.

Задача 1:

На плече важеля довжиною 20см підвісили вантаж масою 400г. Якої маси вантаж потрібно підвісити на плече довжиною 80см, щоб урівноважити важіль? Масою важеля можна знехтувати.

Розв'язання.

$l_1 = 20\text{см} = 0{,}2\text{м}$ $m_1 = 400\text{г} = 0{,}4\text{кг}$ $l_2 = 80\text{см} = 0{,}8\text{м}$ $m_2 - ?$	$\frac{F_1}{F_2} = \frac{l_2}{l_1} => F_2 = \frac{F_1 * l_1}{l_2}$ $F_1 = m_1 * g$ $F_2 = m_2 * g => m_2 = \frac{F_2}{g}$	$F_1 = 0{,}4 * 9{,}8 = 3{,}92$ $F_2 = \frac{3{,}92 * 0{,}2}{0{,}8} = 0{,}98$ $m_2 = \frac{0{,}98}{9{,}8} = 0{,}1\text{кг}$

Відповідь: $m_2 = 0{,}1\text{кг}$

Лістинг до задачі 1:

```
package examples;

import java.text.DecimalFormat;

public class ExampleOne {

   DecimalFormat df = new DecimalFormat("#.##");

   public double l1 = 0.2;
   public double m1 = 0.4;
   public double l2 = 0.8;
   public double g = 9.8;
```

```
    public double solutionForceOne(){
        double F1 = m1 * g;

        System.out.println("F1 = " + df.format(F1));

        return F1;
    }

    public double solutionForceTwo(){
        double F2 = solutionForceOne() * l1 / l2;

        System.out.println("F2 = " + df.format(F2));

        return F2;
    }

    public double solutionMassTwo(){
        double m2 = solutionForceTwo() / g;

        System.out.println("m2 = " + df.format(m2));

        return m2;
    }
}
```

Задача 2:
Яку потужність має атомна електростанція, ККД якої становить 26%, а щодоби витрачається 150 г ізотопу Урану-235? Вважайте, що внаслідок кожного поділу ядра виділяється енергія 200 МеВ.

Розв'язання.

$\eta = 26\% = 0{,}26$
$m = 150г = 0{,}15$ кг
$t = 1$ доба $= 8{,}64 * 10^4 =$
$= 1 * 24 * 3600$ с
$E_0 = 200$ МеВ $= 200 *$
$*10^6 * 1{,}6 * 10^{-19}$ Дж $=$
$= 3{,}2 * 10^{-11}$Дж
$M = 235 * 10^{-3}$ кг/моль
$N_A = 6{,}02 * 10^{23}$ моль$^{-1}$

$P_{кор} - ?$

$$\eta = \frac{E_{кор}}{E_{повна}}$$
$$E_{кор} = P_{кор} * t$$
$$E_{повна} = N * E_0$$
$$N = \frac{m}{M} * N_A$$
$$E_{повна} = \frac{m}{M} * N_A * E_0$$
$$\eta = \frac{P_{кор} * t}{\frac{m}{M} * N_A * E_0} = \frac{P_{кор} * t * M}{m * N_A * E_0}$$
$$P_{кор} = \frac{\eta * m * N_A * E_0}{t * M}$$

$$P_{кор} = \frac{0{,}26 * 0{,}15 * 6{,}02 * 10^{23} * 3{,}2 * 10^{-11}}{24 * 3600 * 235 * 10^{-3}}$$
$P_{кор} \approx 37$ МВт

Відповідь: $P_{кор} \approx 37$ МВт

Лістинг до задачі 2:

```
package examples;
```

```
import java.text.DecimalFormat;

public class ExampleTwo {
   DecimalFormat df = new DecimalFormat("#.##");

   public double n = 0.26;
   public double m = 0.15;
   public double t = 1 * 24 * 3600;
   public double E0 = 3.2 * Math.pow(10, -11);
   public double M = 235 * Math.pow(10, -3);
   public double N = 6.02 * Math.pow(10, 23);

   public double solutionP(){
      double P = (n * m * N * E0) / (t * M);
      P /= 1000000;
      System.out.println("P = " + df.format(P));
      return P;
       }
}
```

Тепер потрібно написати головний клас, з якого починає роботу програма.

Лістинг головного класу:

```
import examples.ExampleOne;
import examples.ExampleTwo;

public class Main {
   public static void main(String[] args) {

      ExampleOne exampleOne = new ExampleOne();

      exampleOne.solutionForceOne();
      exampleOne.solutionForceTwo();
      exampleOne.solutionMassTwo();

      ExampleTwo exampleTwo = new ExampleTwo();
      exampleTwo.solutionP();

   }
}
```

Теоретичний матеріал. Перейдемо до теорії, яку слід надати на першому занятті. Приклади коду будуть взяті з двох попередніх задач. Всі матеріали може бути знайдено у джерелах, вказаних у розділі літератури.

Слід сказати, що Java є об’єктно орієнтованою мовою.

Об'єктно-орієнтоване програмування – одна з найбільш важливих методологій розробки, яка ґрунтується на уявленні про програму як про

сукупність об'єктів, кожен з яких є екземпляром певного класу, а класи утворюють ієрархію наслідування.

Перше, на що слід звернути увагу – це пакети, або pacage.

У Java файли групуються в пакети для кращої організації коду. Пакети – це просто теки, де зберігаються файли з кодом Java. Коли ви створюєте новий проект у Java, зазвичай ви організовуєте свій код в різних пакетах залежно від його функціональності або логічної спільності.

Наприклад, якщо у вас є багато класів, які відносяться до прикладів використання вашого програмного забезпечення, ви можете створити пакет з назвою "Examples" і помістити всі ці класи туди. Таким чином, коли ви або інші розробники працюють з вашим кодом, їм буде легше знаходити потрібні класи, оскільки вони будуть організовані у відповідних пакетах. Приклад: package Examples.

Бібліотеки в Java – це набори готового коду, які можна використовувати для виконання різних завдань без необхідності писати код з нуля. Вони містять в собі класи, інтерфейси та інші компоненти, які допомагають у вирішенні типових задач. Бібліотеки розширюють функціональність мови Java, надаючи доступ до різних інструментів та функцій, які можуть бути використані в програмах.

Наприклад, java.text.DecimalFormat, яка знадобиться для вирішення задач – це частина стандартної бібліотеки Java, яка надає засоби для форматування чисел у вигляді рядків. Цей клас дозволяє встановлювати формат чисел, наприклад, вказуючи кількість знаків після коми, використовуючи різні символи роздільника груп тисяч і таке інше.

У Java існують чотири рівні доступу, які контролюють доступ до класів, методів та інших частин коду. Ось їх опис:

public (публічний): Це найбільш широкий рівень доступу. Класи, методи та інші елементи з цим рівнем доступу можуть бути доступні з будь-якого місця в програмі, навіть з інших пакетів.

protected (захищений): Елементи з цим рівнем доступу доступні в межах того ж пакета, а також у підкласах навіть тоді, коли вони знаходяться в іншому пакеті. Це означає, що вони доступні для успадкування.

default (за замовчуванням): Якщо не вказано інший рівень доступу (public, protected або private), то за замовчуванням встановлюється рівень доступу "default". Елементи з цим рівнем доступу доступні тільки в межах того ж пакета.

private (приватний): Це найбільш обмежений рівень доступу. Елементи з цим рівнем доступу доступні тільки у межах того самого класу, де вони були оголошені. Це означає, що вони недоступні для інших класів, навіть для класів у тому ж пакеті чи підкласів.

Рівень доступу встановлюється перед оголошенням класу, методу або змінної.

Класи – це основна будівельна одиниця в об'єктно-орієнтованому програмуванні (ООП) в Java і багатьох інших мовах програмування. Вони використовуються для опису об'єктів, які мають спільну структуру та поведінку.

Об'ява класу включає в себе наступні елементи:

Ключове слово class: Вказує компілятору, що ми оголошуємо новий клас.

Ім'я класу: Це ім'я, за яким ми будемо посилатися на клас у нашій програмі. Ім'я класу повинно починатися з великої літери.

Тіло класу (в фігурних дужках {}): Це місце, де ми визначаємо атрибути (змінні) та методи класу.

Отже, загальний формат об'яви класу виглядає так:

```
public class ClassName {
}
```

Конструктори – це спеціальні методи в класі, які використовуються для ініціалізації об'єктів. Коли ви створюєте новий об'єкт класу за допомогою оператора new, викликається конструктор цього класу для ініціалізації нового об'єкта.

Приклад: DecimalFormat df = new DecimalFormat("#.##");

Де new DecimalFormat("#.##") це конструктор, який створює новий об'єкт класу DecimalFormat. У цьому конструкторі передається рядок "#.##", який вказує, як саме числа будуть відформатовані. У цьому випадку "#.##" означає, що будуть збережені два знаки після коми.

df – це змінна класу DecimalFormat.

Змінні в програмуванні – це контейнери для зберігання даних. Вони дозволяють вам зберігати, читати і змінювати значення в ході виконання програми. Ось деякі ключові аспекти змінних в Java:

Тип даних: Кожна змінна має тип даних, який вказує на вид і обсяг даних, які вона може зберігати. Наприклад, цілі числа зберігаються в змінних типу int, числа з плаваючою точкою - в double, рядки - в String тощо.

Ім'я: Змінні мають ім'я, яке використовується для посилання на них в коді. Ім'я змінної повинно починатися з літери або символу підкреслення (_), може містити літери, цифри і символ підкреслення, і не може співпадати з ключовими словами.

Ініціалізація: Змінні можуть бути ініціалізовані – це означає присвоєння їм початкового значення. Якщо змінну не ініціалізувати явно, вона отримає значення за замовчуванням для свого типу даних (нуль для числових типів, false для boolean, null для посилань тощо).

Область видимості: Змінні можуть мати область видимості, що визначає, де вони можуть бути доступні в програмі. Зазвичай, змінні, оголошені в межах блоку коду (наприклад, в тілі методу), будуть доступні лише в цьому блоку.

DOI: https://doi.org/...

Змінні та константи: У Java також існують константи – це змінні, значення яких не можна змінити після ініціалізації. Ключове слово final використовується для оголошення констант.

Ідентифікатори: Ідентифікатори в Java (ім'я змінних, методів, класів тощо) регістрозалежні, тобто різні змінні з різним регістром літер вважаються різними. Наприклад, myVariable, MyVariable і myvariable – це різні змінні.

Змінні в Java дуже важливі для роботи з даними та забезпечення гнучкості програм. Вони дозволяють зберігати та маніпулювати різними типами даних під час виконання програми.

Методи в Java – це блоки коду, які виконують певні дії. Вони оголошуються в межах класу і можуть бути викликані для виконання певних завдань. Ось деякі ключові аспекти методів:

Синтаксис оголошення методу: Оголошення методу містить його модифікатор доступу, тип повернення, ім'я методу, список параметрів і тіло методу. Наприклад:

```
public int add(int a, int b) {
    return a + b;
}
```

У цьому прикладі:

public – модифікатор доступу, який вказує на те, що метод може бути викликаний з будь-якого місця програми.

int – тип повернення, який вказує на те, що метод повертає ціле число.

add – ім'я методу.

(int a, int b) – список параметрів методу. У цьому випадку метод add приймає два цілочисельних параметра a та b.

{} – тіло методу, де виконуються певні дії.

Тип повернення: Метод може повертати значення певного типу або не повертати значення зовсім. Якщо метод не повертає значення, його тип повернення повинен бути вказаним як void.

Параметри методу: Параметри методу – це значення, які передаються методу при його виклику. Вони вказуються в дужках після імені методу.

Тіло методу: Тіло методу - це блок коду, який виконується при виклику методу. В ньому визначаються дії, які потрібно виконати.

Виклик методу: Метод можна викликати з іншого місця програми, вказавши його ім'я та передавши потрібні аргументи (якщо вони потрібні).

Методи є важливим елементом в програмуванні, оскільки вони дозволяють структурувати код, уникати повторень та забезпечувати повторне використання функціональності.

Тепер поговоримо про основний метод в Java, з якого починає виконуватись код:

```
public static void main(String[] args) {
}
```

Це основний метод у багатьох програмах на Java. Коли ви запускаєте програму Java, виконання починається з методу main. Ось що треба знати про цей метод:

Синтаксис: Він завжди оголошується як public static void main(String[] args){}.

public: Це модифікатор доступу, який робить метод доступним з будь-якої частини програми.

static: Це ключове слово, яке вказує на те, що метод є статичним, тобто він належить класу, а не конкретному об'єкту класу.

void: Це тип повернення методу, який означає, що метод не повертає жодного значення.

main: Це ім'я методу. У Java, метод, який викликається при запуску програми, завжди повинен мати ім'я main.

Основне призначення методу main полягає в тому, щоб виконувати основні дії програми, такі як ініціалізація, введення даних, обробка, виведення результатів тощо. Все, що ви хочете, щоб ваша програма зробила при запуску, зазвичай виконується в методі main.

Виклик методу. Виклик методу класу полягає в тому, що ми вико ристовуємо об'єкт класу для виклику його методу. Ось як це відбувається в коді:

Створення об'єкта класу:

Перш ніж викликати метод, ми створюємо об'єкт цього класу за допомогою оператора new. Наприклад, ExampleOne exampleOne = new ExampleOne(); створює новий об'єкт класу ExampleOne із ім'ям exampleOne.

Виклик методу за допомогою об'єкта:

Після створення об'єкта ми можемо викликати його методи, використовуючи ім'я об'єкта, крапку (.) і ім'я методу. Наприклад, exampleOne.solutionForceOne(); викликає метод solutionForceOne() для об'єкта exampleOne.

Виконання коду методу:

Коли ми викликаємо метод, виконується код, який був визначений у тілі цього методу. У даному прикладі методи solutionForceOne(), solutionForceTwo(), solutionMassTwo() з класу ExampleOne та метод solutionP() з класу ExampleTwo виконують певні обчислення та виводять езультати.

Таким чином, виклик методу класу полягає в тому, що ми використовуємо об'єкт класу для доступу до його функціональності, описаної у вигляді методів, і виконуємо певні дії, які були визначені у тілі цих методів.

Перейдемо до труднощів технічного характеру та їх вирішення, а саме: технічне забезпечення, відсутність інтернету, програмне забезпечення.

Стосовно технічного обладнання зараз школи по мірі можливостей допомагають з цим, посилають запити на ноутбуки для учнів. Тому це не буде проблемою.

Навчальні матеріали, документація з програмування можуть бути попередньо завантажені і використовуватися за відсутності інтернету.
Програмне забезпечення для рівня потрібного на даному етапі – безкоштовне.

Висновки

Даний підхід до навчання має великий потенціал, допоможе учням засвоювати краще одразу два предмети, через зацікавленість хоча б одним з них. На даному прикладі – кому цікава фізика, почне опановувати програмування, хто зацікавиться програмуванням або вже зацікавлений, як приємний бонус підніме свій рівень фізики. Далі можна буде обирати іншу мову програмування, застосовувати даний підхід до зв'язку з іншими предметами. Зараз учні як ніколи потребують якісну та головне релевантну освіту.

Література

1. Блох Дж. Java. Ефективне програмування. 3-тє вид. Діалектика, 2023. 464 с.
2. Бхаргава А. Грокаємо алгоритми. Ілюстрований посібник для програмістів і допитливих. ArtHuss, 2020. 280 с.
3. Кожухівський А. Атомна енергія | Тест з фізики – «На Урок». Освітній проект «На Урок» для вчителів. URL: https://naurok.com.ua/test/atomna-energiya-2899742.html (дата звернення: 22.05.2024).
4. Левшенюк В. Я., Левшенюк Я. Ф., Трофімчук А. Б. Завдання для тематичного контролю. Фізика. 8 клас. Рівне : Прінт-Експрес, 2009. 64 с.
5. Мартін Р. Чиста архітектура: мистецтво розробки програмного забезпечення. Фабула, 2019. 416 с.

Vyacheslav V. Voroshilov, Oleksandr Y. Beloshapka
Donbas State Pedagogical University, Sloviansk, Ukraine;
Sloviansk secondary school of I-III centuries No 10, Sloviansk, Donetsk region, Ukraine.

Integration of computer science into physics education

The article dedicated to highlighting the benefits of integrating programming and physics for students. The choice of the initial subject and programming language for this interdisciplinary synergy is substantiated. The article provides examples of both theoretical and practical materials for interdisciplinary connections – two problems from different topics and of varying complexity, a Java code listing that solves the given problems, and theoretical material for understanding and explaining the code.

Keywords: *educational process, relevance, programming, Java, problem-solving.*