Списание „Математика, компютърни науки и образование“ Том 7, кн. 1, 2024
Journal of Mathematics, Computer Science and Education Vol. 7, No. 1, 2024

DOI: 10.54664/KGJJ7642

КЛАСИФИЦИРАНЕ НА ТРУДНОСТИТЕ ПРИ ОБУЧЕНИЕ НА СТУДЕНТИ ПО УЧЕБНАТА ДИСЦИПЛИНА „ПРОГРАМИРАНЕ НА JAVA“

Мария Павлова Христова

CLASSIFYING THE DIFFICULTIES IN TEACHING STUDENTS IN THE SUBJECT “JAVA PROGRAMMING”

Mariya Pavlova Hristova

Abstract: Students experience multiple difficulties in learning the Java programming language. This report attempts to classify these difficulties based on the author's experience from the laboratory classes in the Java Programming course with the Computer Science and Software Engineering majors. The classification will help to suggest workable methods and means to overcome or at least minimize these difficulties. Observations show that the main difficulties are related to the transition from the already studied language (C++ or C#) to the new syntactic constructions and abstractions of Java, the different implementation of the key concepts and mechanisms, and the application of the object-oriented approach when solving specific tasks.

Keywords: IT, computer programming, projects, informatics, education, exercises, students, teaching Java.

ВЪВЕДЕНИЕ

В епохата на бързо развиващата се технология, познаването на програмирането е от съществено значение за много аспекти на нашия живот. Един от ключовите елементи на този процес е усвояването на парадигмата на обектно-ориентираното програмиране (ООП). Типичен представител на обектно ориентираните езици е Java, който освен обектно-ориентирано програмиране, предоставя възможности от функционалното и процедурното програмиране, което го прави мощен и универсален език за създаване на софтуер. Java е изключително важен в индустрията на софтуера, като се използва за разработка на разнообразни приложения, уебсайтове и мобилни приложения. Но въпреки своята универсалност, ученето на Java може да бъде предизвикателство, особено за хората, които за пръв път се срещат с програмирането или имат ограничен опит в областта.

Java е „съвременен език за програмиране от високо ниво с отворен код и същевременно е подходящ за начинаещи”[1]. Той е обектно ориентиран, което означава, че поддържа основните принципи на ООП, които позволяват разработването на по-структурирани, лесно поддържащи се и разширяеми програми – абстракция, капсулиране, наследяване, полиморфизъм. Научната литература и интернет пространството изобилстват от източници, които са посветени на преподаването по Java и са предложени различни подходи за обучение.

В този контекст, обучението по програмиране на Java изисква постоянно усъвършенстване и много практика. Подходящите учебни съдържания, примери и проекти могат да помогнат на студентите да надникнат в света на Java и да изградят основни умения за разработка на софтуер.

Тези предизвикателства, въпреки трудностите си, предлагат възможности за растеж и развитие на уменията в програмирането, което е от съществено значение в съвременния свят на технологиите.

В университетите програмирането и учебните курсове по Java играят ключова роля в обучението на студентите от „компютърни“ специалности. Тези курсове предоставят фундаментални познания и практически умения в програмирането с езика Java.

В този доклад се акцентира върху основните предизвикателства пред студентите, които включват разбиране и прилагане на синтаксиса на Java, разбиране на концепциите на ООП, работа с паметта и управлението на ресурсите, дебъгване и отстраняване на грешки и възприемане на различните понятия. Основната му цел е да се предложат идеи за улесняване на процеса на обучение и усвояване на знанията от студентите. Направен е анализ на избрана литература и са проведени анкети със студенти, които са преминали през курса.

Изложение

Обучението не е просто предаване на знание, то е активен процес на включване и участие на студентите в реални практически сценарии и задачи, свързани с програмирането [8]. Вместо да бъдат просто информирани за синтаксиса на Java и основните концепции, студентите имат възможността да учат чрез практически упражнения и проекти, които им позволяват да прилагат уменията си в реални ситуации. Този подход към обучението е важен, защото позволява на студентите да развият не само техническите си умения, но и способността си да мислят аналитично, да решават проблеми и да работят в екип, което са ключови умения за успешната им кариера в областта на софтуерното инженерство.

Обучението по Java се фокусира не само върху основните концепции на програмирането, но и върху различни аспекти на разработката на софтуер. Това включва обектно ориентирано програмиране, структуриране на данни, алгоритми, управление на изключения, работа със стрингове, масиви и колекции, разработка на графични потребителски интерфейси и много други [9]. Това са и едни от основните неща, които затрудняват студентите по време на тяхното обучение.

За да се открият и систематизират трудностите, които срещат студентите, е проведено изследване чрез анкета, наблюдение и анализ на резултатите от контролни работи на студентите от специалностите „Компютърни науки“ и „Педагогика на обучението по история и информационни технологии“ към ВТУ „Св. св. Кирил и Методий“.

1. Трудности, които срещат студентите при изучаване на Java

Причините за проблемите, свързани с усвояването на предмета „Програмиране на Java“, могат да бъдат разнообразни и включват липса на способности за абстрактно мислене и логика, когнитивни фактори като индивидуални стилове на учене и ниво на мотивация, трудности при усвояването на програмирането, което пък изисква използване на множество умения и процеси, включително анализ на проблеми, разработка на алгоритми, отстраняване на грешки и тестове на програмен код.

Едно от първите неща, с които се сблъскват студентите при изучаването на Java, е разбирането на основните концепции и синтаксис на езика. За тяхно улеснение, повечето университети предлагат първо курсове по обектно ориентирано програмиране (ООП), които имат за цел да ги въведат във важните концепции като класове, обекти, методи, наследяване, полиморфизъм и капсулация и учат как да се използват различни структури от данни и алгоритми, които са основни за разработката на софтуерни приложения.

„Основна критика от педагогическа гледна точка е наличието на прекалено много нови термини, които се струпват накуп пред обучавания.“ [2] За хора, които не са учили друг програмен език преди това като C++ или C# се намесват понятия като модификатори за достъп (public); класове (class); статични методи (static); входни параметри на метод, в които клас за текстови низ (String) и масив ([]); изходни параметри (void); и извикване на метод от основната библиотека (System.out.println).

В [6] е направен анализ на вида и честотата на проблемите с качеството на кода, които възникват в два милиона Java програми на начинаещи програмисти. Те също така проучват дали учениците могат да решават тези проблеми и дали решаването на проблеми с качеството на кода се подобрява, когато учениците имат инсталирани инструменти за анализ на код. Друга констатация е, че използването на инструменти има малък ефект върху възникването на проблеми [7].

Според направено проучване от [4] преподавателите имат различни мнения относно честотата на грешките, които правят начинаещите програмисти. Освен това, убежденията на преподавателите не съвпадат с реалните грешки, които се появяват в програмите на студентите, които изучават езика за програмиране Java. Това несъответствие подчертава нуждата от допълнителни изследвания за проблемите в програмирането, които ще помогнат на преподавателите да разберат къде точно студентите най-често грешат и имат нужда от по- засилено внимание.

Учебната дисциплина „Програмиране на Java“ се провежда в продължение на един семестър и включва общо 30 часа лекции и 30 часа упражнения, които се разпределят в рамките на 15 седмици. Обикновено упражненията са синхронизирани с лекциите и се провеждат в същата седмица. Студентите са разделени на няколко групи, което осигурява по-добра възможност за разбиране на учебното съдържание. Този подход също така дава възможност на преподавателите да оценят по-ефективно нуждите и напредъка на всяка отделна група студенти. Наблюденията показват, че студентите се затрудняват предимно със синтаксиса на езика, с възприемането на различните понятия, с отстраняването на грешки, с работата с изключения и управление на паметта.

За да се разбере мнението на студентите относно трудностите, които срещат, се проведе анонимно анкетно проучване, в което взеха участие 37 участника. Проучването има за цел да определи кои са точно проблемите, върху които трябва да се поработи, и да се наблегне на начините за решаването им. Като обобщение от анкетното проучване може да се каже, че повечето студенти са доволни от предложеното учебното съдържание, но срещат много затруднения при разбирането и решаването на конкретна задача.

Фигура 1. *Резултати от анкетното проучване за най-често срещаните затруднения по учебната дисциплина „Програмиране на Java“*

На Фигура 1. са представени резултатите от анкетното проучване, фокусиращо се върху най-често срещаните трудности по учебната дисциплина „Програмиране на Java“. По-голяма част от анкетираните са на мнение, че лекциите и упражненията не са достатъчни и може би това е една от основните пречки за усвояването на учебното съдържание. Въпреки че на останалите въпроси са посочени различни отговори, обобщено може да се достигне до извода, че работата с класове и обекти, различния синтаксис, спрямо изучаван преди това език като C++, и разбирането и прилагането на принципите на обектно ориентираното програмиране, писането на Unit тестове

са сред основните предизвикателства, с които се сблъскват студентите при обучението по програмиране на Java.

1.1. Затруднение със синтаксиса на езика

Студентите често се сблъскват със сложни конструкции и правила за писане на код, които могат да бъдат доста объркващи, особено ако преди това е изучаван друг език за програмиране с по-различен синтаксис. Това е и една от основните причини да се нуждаят от допълнително обучение и практика. Езикът Java, и не само, изисква строга синтактична структура, като правилното използване на скоби, точки, точки и запетаи и тяхното коректно използване е от съществено значение за работата на програмата. Грешките в синтаксиса могат да доведат до неработещи програми. Затова е важно да се отдели специално внимание на синтаксиса по време на обучението и да се упражнява редовно, за да се осигури разбиране и усвояване на правилните практики.

Студенти, на които се налага да използват онлайн среда за разработка, могат да срещнат много повече трудности. Инсталирането на професионални среди за разработка като Apache NetBeans или IntelliJ IDEA може да предостави значителни предимства като подсказки за синтаксиса и автоматично допълване на кода.

В Java, методът за извеждане на съобщения на конзолата се нарича println, а много от студентите, които използват онлайн компилатор, пишат printin. Това също е една от честите грешки при писане на програми на Java.

При използването на условни конструкции като if, студентите често допускат грешка, като объркват оператора за присвояване (=) с оператора за сравнение (==). Това води до синтактична грешка, защото в условния израз вместо да проверяват дали две стойности са равни, те неволно опитват да присвоят стойност на променлива. Тази грешка е често срещана сред начинаещите програмисти, тъй като операторите за присвояване и сравнение изглеждат подобни, но имат съвсем различни функции.

Това са само малка част от примерите за допускане на синтактични грешки, които се виждат по време на упражненията по „Програмиране на Java“.

1.2. Проблеми с възприемането на различните понятия

Проблеми с възприемането на различните понятия в езика за програмиране Java могат да възникнат поради сложността на някои концепции или поради неясната в тяхното обяснение. Някои трудни за разбиране за начинаещите студенти понятия са класове, обекти, наследяване, полиморфизъм, абстракция и капсулация. Освен това разбирането на начина, по който Java управлява паметта и работи с референции към обекти, може да бъде предизвикателно за студентите, които са свикнали с езици като C или C++, където се използват явно указатели.

Например разграничаването на понятията „клас“ и „обект“ е обичайна трудност при започването на учене на Java. Студентите не съумяват да правят разлика между едното и другото. Класът определя структурата и поведението на обектите от даден тип, докато обектът е конкретен екземпляр на този клас, създаден по време на изпълнение на програмата.

1.3. Проблем с отстраняването на грешки

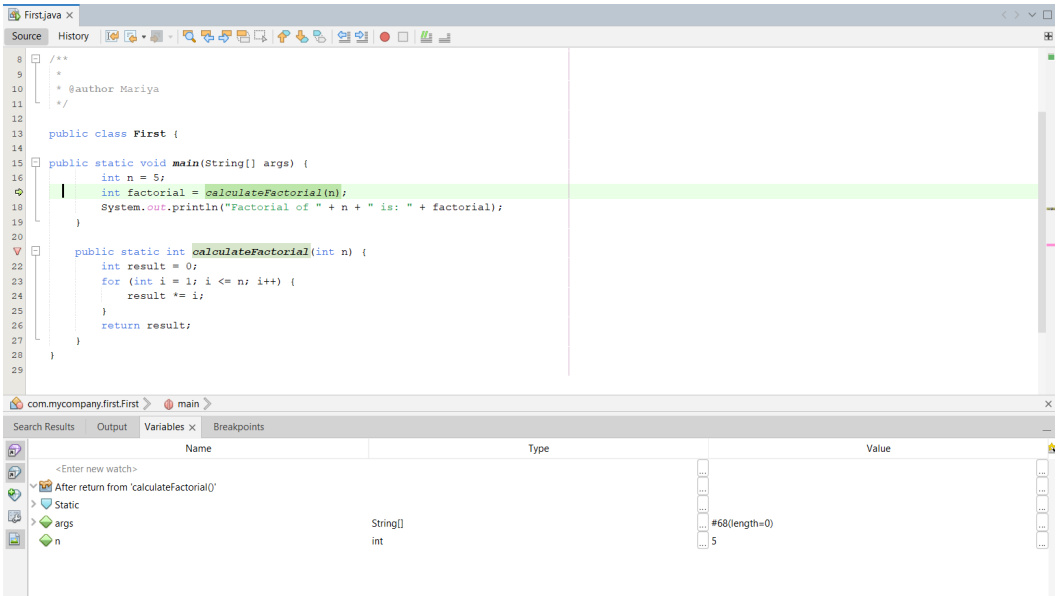
Отстраняването на грешки в Java е важна част от процеса на програмиране и обикновено се извършва чрез внимателно преглеждане на кода и редактиране на неправилно написаните части. Съществуват различни подходи и техники, които студентите трябва да усвоят. Една от тях е използването на дебъгер. Дебъгерът е инструмент, който позволява да се следи изпълнението на кода стъпка по стъпка и да се анализират стойностите на променливите във всяка точка от програмата. Това помага за по-бързото им откриване и отстраняване. Други подходящи подходи са използване на изключения, писане на тестове и т.н.

```
public class Main {
    public static void main(String[] args) {
        int n = 5;
        int factorial = calculateFactorial(n);
      System.out.println("Factorial of " + n + " is: " + factorial);
    }
    public static int calculateFactorial(int n) {
        int result = 0;
        for (int i = 1; i <= n; i++) {
            result *= i;
        }
        return result;
    }
}
```

В примерния код, който е предоставен, се изчислява факториелът на числото 5, но резултатът е некоректен. Вместо очакваното „**Factorial of 5 is: 120“**, получаваме „**Factorial of 5 is: 0“**. Това е ясен индикатор за наличие на логическа грешка в програмата.

Проблемът в кода е в неправилното изчисление на факториела. В метода **calculateFactorial(int n)** е инициализирана променливата **result** със стойност 0, а след това се използва операция за умножение (**result *= i;**). Това означава, че резултатът винаги ще бъде 0, тъй като всяко число се умножава по 0.

За да се открие и отстрани тази логическа грешка, се използва дебъгер. Трябва да се добави точка за спиране (breakpoint) на началото на метода **calculateFactorial(int n)**. След това програмата се стартира с възможност за дебъгване и изпълнението ѝ се проследява стъпка по стъпка. При всяка стъпка може да се преглеждат стойностите на променливите и да се анализира как се променят по време на изпълнението на цикъла. Това ще помогне да се идентифицира проблема и да се коригира, като се промени начина на изчисление на факториела.



Фигура 2. *Използване на дебъгер*

1.4. Затруднение при работа с изключения

Използването на изключения (exceptions) в Java е важен механизъм за обработка на грешки по време на изпълнение на програмата. Те предоставят начин за сигнализиране за

програмни грешки, които могат да възникнат по време на изпълнението на програмата и които могат да прекъснат нормалното ѝ изпълнение. Обработката им става чрез конструкции като try-catch блокове, които позволяват да се улавят (catch) изключенията и да се извършват необходимите действия за обработка на грешките.

Едно от основните предизвикателства, пред които се изправят студентите, е да разберат къде точно да поставят конструкции за обработка на изключения (try-catch блокове) в тяхната програма. Това изисква разбиране на потенциалните места, където може да възникне грешка, както и да се определи подходящата стратегия за обработка на тези грешки. Например, студентите трябва да разберат, че блокът try се използва за обхващане на код, който може да хвърли изключение, докато блокът catch се използва за обработка на хвърленото изключение. Те трябва да знаят как да напишат try-catch блокове по начин, който да предотврати срив на програмата и да осигури подходяща реакция на грешките.

Друг аспект, който може да бъде предизвикателен, е разбирането на различните видове изключения в Java и кога е подходящо да се използват. Студентите трябва да разберат разликата между проверяеми (checked) и непроверяеми (unchecked) изключения и да знаят как да ги обработват ефективно в техните програми.

Технологиите в образованието дават възможност за персонализирано обучение, което отговаря на индивидуалните нужди на всеки студент, което от своя страна предоставя значителни предимства за студентите и за преподавателите. Студентите могат да учат със собствено темпо и да подбират учебното съдържание според своите нужди.За преподавателите персонализираният подход позволява по-добро използване на технологиите и им дава възможност да следят за напредъка на студентите, да им предоставят индивидуална обратна връзка и да им помагат за преодоляване на трудностите по време на обучението.

2. Фактори за успеха в обучението по „Програмиране на Java“

В проучване [3], свързано с обучението по компютърно програмиране се е установило, че студентите не успяват да завършат курсовете поради липсата на вътрешна мотивация, липсата на бъдещи очаквания, безпокойството, влиянието на връстниците и лошите умения и поведение. В допълнение към тези изследвания за неуспехи в курсовете по програмиране, които се фокусират върху неспособността или слабостите на ученика да разбере концепциите, се набляга на това, че методите на преподаване също са фактори, които допринасят за високия процент на провал на тези курсове [13].

Преподавателите по програмните езици трябва да могат да развиват и прилагат разнообразни умения, подходи и качества. Успешните преподаватели се стремят към постоянно учене и развитие както на своите собствени знания и умения, така и на своите методи за преподаване. Те са отворени към нови идеи и иновации в областта на образованието. Комуникацията е ключов аспект от преподавателската работа. Успешните преподаватели са способни да комуникират ясно и ефективно със своите студенти, да слушат внимателно техните въпроси и запитвания и да предоставят ясни отговори и обратна връзка. От особена важност е да могат да мотивират своите студенти да постигат високи стандарти на академичен успех и да развиват своите умения и потенциал.

В курса по „Програмиране на Java“ първоначално се започва с основните концепции, синтаксиса на езика, типовете данни, операторите, условните конструкции и циклите. Преподавателите включват предимно примери от реалния свят, които помагат на студентите да видят как се прилагат уменията, които усвояват, в реални ситуации и приложения.

След успешното преминаване на курса студентите трябва да могат да прилагат основните концепции и умения на Java в реални ситуации и приложения. Те трябва да са в състояние да решават задачи, свързани със създаването на функционални Java приложения, като използват подходящи структури от данни и алгоритми, обектно ориентиран подход, управление на изключения и обработка на грешки, работа с файлове и бази данни, създаване на потребителски интерфейси и тестване на приложения. Друго, с което трябва да могат да се справят, е да разбират и да анализи-

рат конкретни изисквания към проекти и да ги превръщат във функционални програми, които да отговарят на нуждите на потребителите.

Фигура 3. *Класификация на затрудненията*

3. Развиване на програмистки умения чрез различни дидактически подходи в учебната дисциплина „Програмиране на Java“

Изучаването на програмирането е предизвикателен и сложен процес за много студенти [10], тъй като изисква както теоретично разбиране, така и практическо приложение на синтаксиса и семантиката на конкретни езици, както и алгоритмично мислене и умения за програмиране [11].

Някои международни организации (като Computing at School Group; Computer Science Teacher Association; или Association for Computing Machinery) са определили насоки за учебни програми за насърчаване на компютърната грамотност на учениците, които трябва да научат основното съдържание на програмирането, като алгоритми, последователности, променливи, условия, цикли, синхронизация, паралелизъм, процедури и отстраняване на грешки, за да могат да разработят решения на конкретни проблеми [12].

Развитието на програмистки умения изисква ефективна и иновативна образователна стратегия. Прилагането на различни дидактически подходи може да подпомогне този процес, като насърчава активното участие на студентите, развива техните аналитични умения и им предоставя възможности за практическо прилагане на придобитите знания.

Има много подходящи начини да се представи учебното съдържание на студентите в началните курсове, свързани с програмиране на Java. Най-често използваните са структуриране на уроците по конкретни теми и задачи, което позволява на студентите да се фокусират върху конкретни умения и концепции. Това може да включва изучаване на основни концепции като синтаксис на езика, типове данни, оператори, условни конструкции и цикли чрез демонстрации и упражнения. Съществуват и други, по-подходящи начини, за привличане на вниманието на студентите – програмиране по двойки, интерактивни демонстрации, използване на визуални помощни средства, както и състезателен подход. Тези методи на обучение не само ангажират студентите активно в учебния процес, но и ги подготвят за реални сценарии на приложение на знанията, като ги окуражават да работят в екип, да изследват проблеми и да прилагат концепциите, които са учили, в решаването на реални задачи.

3.1. Програмиране по двойки

Програмирането по двойки е изключително ефективен метод при обучението на студенти. Това е техника при която двама души работят на един компютър, като единия от тях пише кода, а другия следи написаното, анализира го и предлага подобрения. Този, който пише програмния

код е фокусиран върху синтаксиса, докато наблюдаващия има поглед върху общата структура и логиката на програмата. „Програмирането по двойки може да подобри познанията на учениците по програмиране и да им помогне да постигнат по-добро представяне в обучението. Той също така помага на обучаемите да създават висококачествени кодове и намалява времето за изпълнение на задачите по програмиране, отколкото използването на самостоятелно програмиране. При подходящи условия е от полза за насърчаване на увереността на учащите и мислене от по- висок ред, като изчислително мислене. Освен това учениците, обучавани с програмиране по двойки, съобщават за по-високо удовлетворение и удоволствие от тези, които работят самостоятелно“ [5].

При този метод на работа не само се подобряват познанията на студентите по програмиране, но и им се помага да стигнат до по-добро представяне в обучението. Те се учат да комуникират ефективно помежду си и да разпределят задачите си спрямо силните страни и интересите на всеки на всеки един. При поява на проблем те могат да предложат различни решения и да работят заедно за тяхното отстраняване. Подходът на съвместно програмиране поощрява студентите да бъдат отворени към различни гледни точки и да ценят важността на екипната работа. Освен това, то може да увеличи мотивацията на студентите и да им осигури по-приятен и изпълнен със смисъл учебен опит. Така че, тези умения и опит, придобити чрез програмиране по двойки, имат дългосрочни ползи за развитието на студентите както в академичен, така и в професионален план.

3.2. Интерактивни демонстрации

Интерактивните демонстрации са метод на преподаване, при който студентите активно участват в процеса на решаване на проблеми и наблюдават как те се реализират в реално време. В този контекст преподавателят използва различни инструменти и технологии, като симулации, визуализации и динамични примери с код, за да илюстрира учебния материал. Тези демонстрации позволяват на студентите да видят конкретни приложения на теоретични концепции, като ги свързват с реални задачи и сценарии.

Подходът включва няколко етапа. Първоначално, преподавателят представя проблема и приканва студентите да обсъдят възможни решения в групи, без намеса. Това е важна част от метода, тъй като насърчава критичното мислене и сътрудничеството сред студентите. След дискусията преподавателят демонстрира свое решение, като пише кода на живо, обяснява всяка стъпка и приканва студентите да се включват с въпроси или предложения. Тази демонстрация е интерактивна, тъй като студентите участват активно, задават въпроси и предлагат различни начини за решаване на задачата.

Една подходяща задача за такава интерактивна демонстрация би могла да бъде създаване на програма за управление на студентски данни, където студентите обсъждат как трябва да изглежда структурата на програмата, как да се обработват данните и какви оптимизации могат да се направят. Примерът с въвеждането на данни за студенти може да бъде модифициран и разширен с допълнителни функции като сортиране на данни, филтриране по критерии или изчисляване на по-сложни статистики.

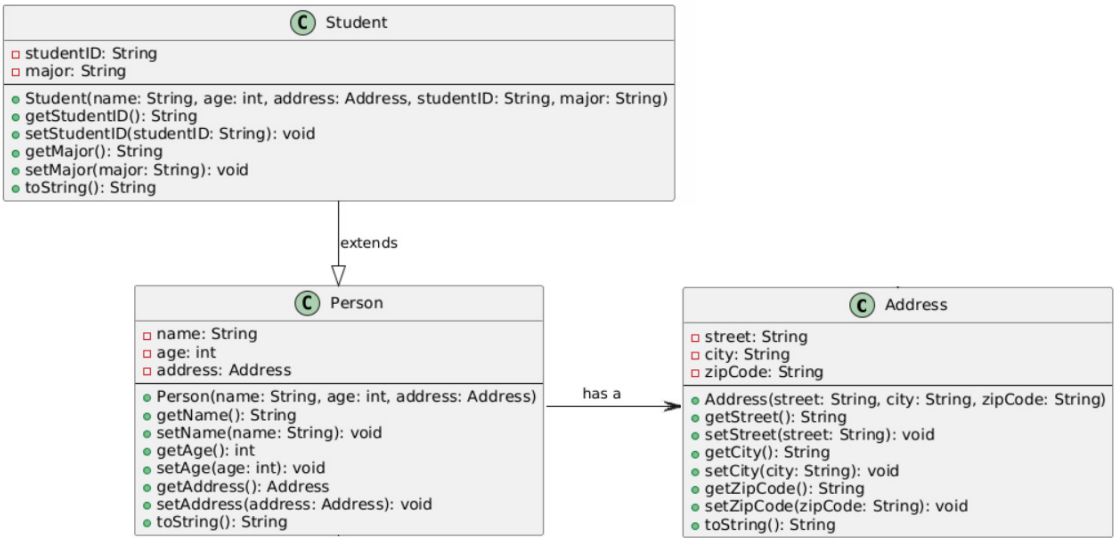
След демонстрацията, студентите и преподавателят обсъждат предимствата и недостатъците на предложеното решение в сравнение с тези, които студентите са генерирали по време на обсъждането. Това насърчава критичен анализ и задълбочено разбиране на различни подходи за решаване на един и същ проблем.

3.3. Използване на визуални помощни средства

Използването на визуални помощни средства е важен метод за улесняване на обучението и разбирането на концепциите от страна на студентите. Тези средства включват графики, диаграми, таблица, анимации и други визуални елементи, които помагат да се представят сложни идеи по по-ясен и достъпен начин.

Пример за използване на визуални помощни средства е представянето на кода на програмата чрез клас-диаграми. Клас-диаграмата е тип UML (Unified Modeling Language) диаграма, която визуализира класовете в система, техните атрибути, методи и връзки помежду им. Тя помага на

студентите да разберат структурата на обектите, с които работят, и как те взаимодействат помежду си.



Фигура *5.* *Клас диаграма на програма*

На Фигура 5 е представена клас диаграма, която илюстрира взаимоотношенията между три основни класа: Address, Person и Student. Класовете са свързани чрез асоцииране и наследяване, показвайки как обекти от типа Person и Student взаимодействат с обекти от типа Address.

Класът Person представлява основна информация за човек и съдържа полета за име, възраст и адрес, който е обект от тип Address. От своя страна, класът Student наследява всички свойства и поведение на Person, като добавя допълнителни полета, специфични за студент, като уникален идентификационен номер на студента (studentID) и специалност (major).

Чрез асоцииране се визуализира връзката “има” между класовете Person и Address, докато наследяването между Student и Person показва йерархична връзка, в която всеки студент е и човек с допълнителни характеристики.

3.4. Състезателен подход в обучението по програмиране на Java

Състезанията предизвикват студентите да се ангажират активно в учебния процес и да развиват своите умения. Постигането на по-добри резултати ги мотивира да учат повече и да се стремят към постигане на високи стандарти. В процеса на решаване на задачи или разработване на проекти студентите се ангажират с реални проблеми и се учат да прилагат своите знания на практика. Този вид активно участие и ангажираност е от съществено значение за техния учебен напредък и развитие.

Когато се постави конкретна задача и определено време за работа, студентите се стимулират да изразходват всички свои умения и знания, за да я решат в срок и по възможно най-добрия начин. Този вид насоченост им помагат да усвоят нови концепции по-бързо и да ги приложат на практика. В същото време, конкуренцията между тях им осигурява допълнителна мотивация да се стремят към постигане на отлични резултати.

По време на упражненията по „Програмиране на Java“ на студентите са поставяни конкретни задачи, които трябва да решат за определен период от време. Този метод, известен като „срочно учене“ или „спринтове“, им помогна да се фокусират върху конкретни цели и да работят ефективно за тяхното постигане. Ограниченият срок за решаване на задачите ги кара да се концентрират върху съществени аспекти на материала и да използват времето си по-ефективно. Това им помага да усвояват новата информация по-бързо и по-ефективно. Освен това при постигане на положителни резултати, студентите се чувстват по-удовлетворени и горди от своя труд. Това ги мотивира да продължават да работят усилено и да се стремят към още по-високи постижения.

5. Резултати от проучването

10% от студентите се затрудняват при работата с класове и обекти, което е показателно за трудността на обектно-ориентираното програмиране (ООП). Това е основна концепция в езици като Java, която включва работа с абстракция, капсулация, наследяване и полиморфизъм. От своя страна, 7% от студентите имат затруднения със самото разбиране на тези принципи. Те са ключови за създаването на добре структуриран код, но често изискват по-задълбочено разбиране и практика, за да бъдат усвоени напълно.

Не малки са и процентите при студентите, които срещат трудности в различни области на Java програмирането: 6% от тях изпитват затруднения с анонимните класове, 7% с изключенията и обработката на грешки, 8% с управлението на паметта и 8% с ламбда изразите. 12% от студентите смятат, че NetBeans е подходящата среда за разработка, докато останалите може да предпочитат други среди като IntelliJ IDEA или Eclipse.

След промяната на методиката с включването на гореспоменатите техники, процентите на затруднения на студентите в областта на обектно-ориентираното програмиране значително намаляха. Студентите показаха по-високи нива на ангажираност и интерес, а усвояването на ключови концепции, като класове, обекти и наследяване, стана по-лесно и достъпно. Визуализацията на концепциите и практическите упражнения, проведени в групи, също допринесоха за по-доброто разбиране и прилагане на теорията в практиката.

ЗАКЛЮЧЕНИЕ

Класифицирането на трудностите при обучение на студенти в програмиране на Java е от съществено значение за разбирането на техните нужди и предизвикателства. Някои студенти могат да се сблъскат с трудности при разбирането на основни и напреднали концепции в програмирането на Java, като например обектно ориентирано програмиране, работа със структури от данни и алгоритми, управление на изключения и други. При други синтаксисът на Java може да бъде объркващ и труден за разбиране, което може да доведе до грешки и затруднения при дебъгването на техния код.

От изключителна важност е преподавателите да използват доказано добри методи за обучение и да предоставят допълнителни упражнения и материали, за да помогнат на студентите да преодолеят затрудненията си. Важно е също така да се насърчава активното им участие и да се осигурява индивидуална подкрепа, когато е необходимо. Освен това, създаването на подходяща образователна среда, където студентите се чувстват комфортно да изразяват своите въпроси и затруднения, е ключово за успешното им обучение.

Тези са само някои от възможните проблеми, които студентите могат да срещнат при ученето на програмиране на Java. Решението на тези проблеми изисква подходящи дидактически материали, добре структурирани уроци и достатъчно практика.

ЛИТЕРАТУРА

[1] **Наков С.,** колектив. Основи на програмирането с Java, София, 2017, ISBN: 978- 619-00-0636-7 // Nakov S., collective, Basics of programming with Java, Sofia, 2017, ISBN: 978-619-00-0636-7

[2] **Петров, Ф. (2023),** Дидактически бележки относно използването на безименни класове, нестатични стартиращи методи и записи при обучението в уводни курсове по обектно-ориентирано програмиране с Java 21; Математика, компютърни науки и образование, 2023 / Том 6 / Брой 1, DOI: https://doi.org/10.54664/DQTC9983, Страници: 57–64. // Petrov, F. (2023), Teaching notes on using anonymous classes, non-static initializer methods, and records in teaching introductory courses in object-oriented programming with Java 21; Mathematics, Computer Science and Education, 2023 / Volume 6 / Issue 1, DOI: https://doi.org/10.54664/DQTC9983, Pages: 57-64

[3] **Akpotuzor, S., Akinola, S., Agbonlahor, I.**, A review on effective approach to teaching computer programming to undergraduates in developing countries, Scientific African Volume 16,

July 2022, https://doi.org/10.1016/j.sciaf.2022.e01240

[4] **Brown, N. C. C. and Altadmri, A.** (2017). Novice java programming mistakes: Large- scale data vs. educator beliefs. ACM Transactions on Computing Education, 17(2).

[5] **Fan Xu, Ana-Paula Correira,** Adopting distributed pair programming as an effective team learning activity: a systematic review, J Comput High Educ. 2023 Feb 15: 1–30, doi: 10.1007/s12528-023-09356-3

[6] **Keuning, H., Heeren, B., and Jeuring, J.** (2017), Code quality issues in student programs. In Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education, pages 110–115/

[7] **Koblaher, C., Vierhauser, M., Groher, I.,** (2023), Common Code Quality Issues of Novice Java Programmers: A Comprehensive Analysis of Student Assignments, In Proceedings of the 15th International Conference on Computer Supported Education – Volume 2: CSEDU, ISBN 978-989-758-641-5, SciTePress, pages 349-356. DOI: 10.5220/0011715400003470

[8] **Lave, J., Wenger, E. (**2012), Situated Learning: Legitimate Peripheral Participation, ISBN: 9780511815355

[9] **Lewis, J., Loftus, W., Cocking, C.,** Java Software Solutions: Foundations of Program Design, 2023, ISBN-13: 9780137920846

[10] **Martins, S. W., Mendes, A. J., Figueiredo, A. D.** Diversifying Activities to Improve Student Performance in Programming Courses. Commun. Cogn. 2013, 46, 39–58.

[11] **Piedade, J.; Dorotea, D.; Sampaio, F.F.; Pedro, A.,** A Cross-analysis of Block-based and Visual Programming Apps with Computer Science Student-Teachers. Educ. Sci. 2019, 9, 181.

[12] **Piedade, J., Dorotea, N., Pedro, A., Matos., J,** On Teaching Programming Fundamentals and Computational Thinking with Educational Robotics: A Didactic Experience with Pre-Service Teachers, Educ. Sci. 2020, 10(9), 214; https://doi.org/10.3390/educsci10090214

[13] **Sarpong, K., Arthur, J., Owusu, P.**, Causes of failure of students in computer programming courses: the teacher – learner perspective, Int. J. Comput. Appl., 7712 (2013), pp. 0975-8887

ИНФОРМАЦИЯ ЗА АВТОРА

Мария Христова, асистент, докторант, специалност
„Информатика и компютърни науки“, Факултет „Математика и информатика“,
Великотърновски университет „Св. св. Кирил и Методий”, e-mail: m.p.ivanova@ts.uni-vt.bg

ABOUT THE AUTHOR

Mariya Hristova, Assistant Professor, PhD student in Methodology of Informatics
and Information Technology Education, Faculty of Mathematics and Informatics,
”St. Cyril and St. Methodius” University of Veliko Tarnovo, e-mail: m.p.ivanova@ts.uni-vt.bg