**Технически университет – Варна** Специалност: Софтуерни и интернет технологии(СИТ)

**Проект по Обектно-ориентирано програмиране първа част**

**Изготвен от Борис Ивайлов Илиев**

**Факултетен номер:20621624**

**Задание на проекта:**

Да се напише програма, реализираща четене и операции с XML файлове. Програмата обработва и записва информация съхранена във файл.

Характеристиките на XML елементите да са:

* Идентификатор на елемента
* Списък от атрибути и стойности
* Списък от вложени елементи или текст
* Да поддържат уникални идентификатори

Програмата да поддържа следните функционалности:

* Да извежда на екрана прочетената информация от XML файла
* Да извежда стойност на атрибут по даден идентификатор на елемента и ключ на атрибута
* Да присвояване на стойност на атрибут
* Да извежда списък с атрибути на вложените елементи
* Да осъществява достъп до n-ти пореден наследник на елемент
* Да осъществява достъп до текста на елемент
* Да изтрива атрибут на елемент по зададен ключ
* Да добавя нов наследник на елемент, със зададен идентификатор
* Да изпълнява прости Xpath 2.0 заявки

**Цел на проекта**

Проектът представлява Java програма, която съхранява и обработва информация свързна със студентски досиета. Всеки „главен“ елемент във файл съдържа списък от различни видове данни и характеристики свързани с един студент. Потребителят комуникира с програмата чрез команден ред, който предоставя възможност за извършване на множество операции свързани с наличните данни. Съхраняването и четенето на данните се извършва с помощта на текстови файл, който работи на принципа на .xml файловете. Самата обработка на прочетените данни се извършва в програмата като промените се запазват само по желание на потребителя.

Програмата осигурява на потребителя някои основни функционалности за работа с xml файлове, като добавяне на нови елементи, атрибути и свързаната с тях информация. Основната информационна единица, с която потребителя работи е обектът “Student” (Студент), чиито основни характеристики са идентификатор(id),име, възраст и комплексната характеристика дисциплина. Разработката предлага пълната свобода да се добавят нови елементи към всеки студент, които са могат да бъдат напълно персонализирани с помощта на фукциите, с които потребителя разполага.

Тази документация ще разгледа реализацията и работата на зададения проект, с помощта на различни изображения, диаграми и схеми. В следващите модули ще разгледат предметната област, проектирането и реализацията на заданието, завършвайки с кратко заключение.

**Преглед на предметната област**

Предметната област, в която тази програма намира своето приложение е административно-академична. Йерархоята на която се подничява организацита на информацията е Студент(име,възраст),дисциплина и по-късно добавени обекти от потребителя. Предоставените функции могат да бъдат разделени в две категории – тези които манипулират информацията, и тези които само имат достъп до нея. Голяма част от използваните алгоритмни и са цикли свързани с обхождане на данни и проверки за техните съвпадения или еднаквости.

За комуникацията между потребителя и програмата се грижи командният ред, който интерпретира зададената команда и нейните аргументи и връща резултата от нея в командния прозорец.

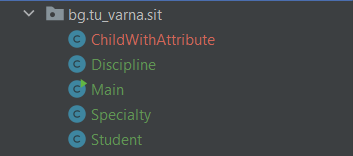
Единственото изискване към потребителя е да бъде добре запознат с командите и техните формулировки, могат да бъдат видени чрез изписване на командата „help“, както и да бъде внимателен в изреждането на аргументите свързани със всяка операция, тъй като поради невнимание могат да възникнат неочаквани грешки.

**Проектиране**

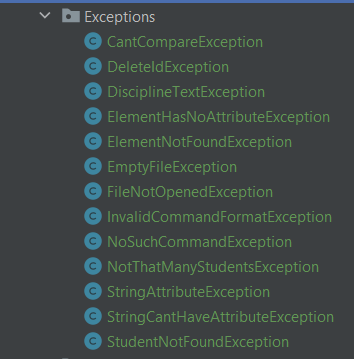
В тази част от документацията ще разгледаме общата структура на системата. Програмата се състои от 3 пакета чиито класове са изобразени долу:

* Основен пакет - bg.tu\_varna.sit съдържа основните класове и информационни единици на продукта. Те са както следват – студент, дисциплина,специалност, Main и класът ChildWithAttribute описващ всеки нов добаввен елемент към Student.
* Пакет с команди – CommandLine съдържащ всички команди, които програмата поддържа. Абстрактния клас CommandFactory разчита зададения от потребителя команден ред и извиква съответната команда. С помощта на FactoryMethod и интерфейсът Command се обединява функционалността на всички команди.
* Пакет с изключения – Exceptions съдържа всички предвидени грешки, които може да възникнат по време на работата с програмата. Ако се изпълни условие за извикване на някое изключение, то ще изведе съответното съобщение за справяне със възникналия проблем.





Основен пакет



Пакет с команди

Пакет с изключения

Основният клас „***Main“***служи за връзка с потребителя, като единствената му функция е да осъществи един безкраен цикъл, който приема команди, обработва ги като взима основната команда и нейните аргументи, и извежда съответните грешки при нужда. Това е неговото съдържание:

public static void main(String[] args) throws IOException {

Scanner sc = new Scanner(System.in);

while(true) {

try {

System.out.print(">");

String variables = sc.nextLine();

String[] command;

command = variables.split("\\s+");

String commandName = command[0];

Object[] commandArgs = new String[command.length - 1];

System.arraycopy(command, 1, commandArgs, 0, commandArgs.length);

Command commands = CommandFactory.getCommandName(commandName);

if (commands != null)

commands.execute(commandArgs);

}

catch (Exception e)

{System.out.println(e);}

}

}

**Реализация и тестване**

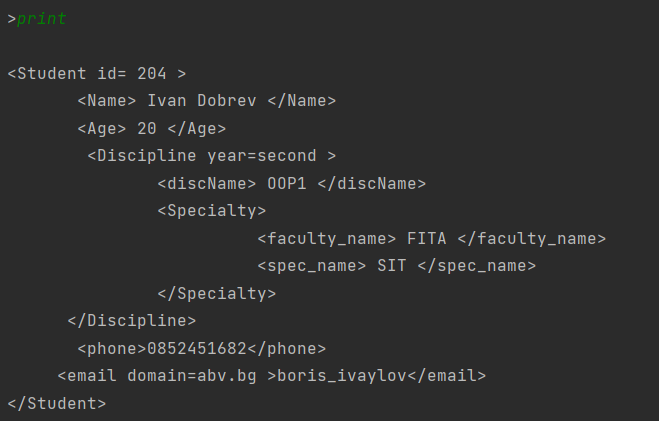
За да представя най-общо работата на програмата ще започна с отварянето и прочитането на предварително попълнен файл.

Нека стартираме програмата и заредим файл presentation с командата open.



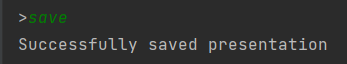
Успешно отворихме и прочетохме информацията във файла. Тъй като по време на работа програмата не работи с постоянно отворен файл, тя еднократно прочита неговото съдържание като го записва в достъпния за цялата програма Arraylist filedata. Именно чрез този начин на четене на информация потребителя може да прави многократни промени в прочетените данни, без да повлиява по какъвто и да е начин на самото съдържание във файла, от който тя е била извлечена.

Следващата команда (print) извежда цялата прочетена информация от файла.



Извеждане на информацията за един студент

Командите save и saveas запазват информацията, намираща се текущо в програмата, с тази разлика, че save запазва редактираната информация във файла от който тя е била първоначално прочетена, докато saveas я запазва в файл оказан от потребителя.

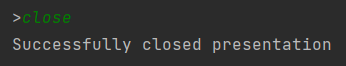


Пример save



Пример saveas

Командата close „затваря“ текущия файл т.е. изтрива цялата прочетена и редактирана преди извикването на командата информация.



Пример Close

Командата exit прекъсва безкрайния цикъл във Main и прекратява работата на програмата.

Всички останали команди освен тези , които бяха описани по-горе се използвата за по-задълбочена работа с прочетената информация.

За бъдеще развитие на проекта бихме могли да доразработим класът ChildWithAttributes, така че да може да изпълнява ролята на главен елемент във файла.Така не само ще се олекоти работата на програмата, но и ще се предостави възможност за работа с разнородни данни, коите може и да не са тясно свързани с досието на един студент.

Източници:

Youtube.com

examples.javacodegeeks.com

stackoverflow.com

[www.geeksforgeeks.org](http://www.geeksforgeeks.org)

<https://www.w3schools.com/>

lightrun.com

https://www.programiz.com/java-programming/library/hashmap/remove