Структури от Данни и Обектно-Ориентирано Програмиране
спец. Компютърни Науки
Практикум №14
02.06.2010г.

ТЕМА: Структура от данни MAP

Задача 1: Да се реализират на езика Java следните класове за представяне на информацията, свързана с изборните дисциплини в бакалавърските програми на Факултета:
Клас **Professor** за представяне на преподавател със следните полета:
- име;
- e-mail;
- титла;
- списък на преподаваните изборни дисциплини.

Клас **Student** за представяне на студент със следните полета:
- име;
- e-mail;
- списък на изборните дисциплини, които слуша в момента;
- списък на изборните дисциплини, по които е държал изпит, заедно с получената оценка за всяка от тях.

Клас **EligibleCourse** за представяне на изборна дисциплина със следните полета:
- идентификатор и име(символни низове);
- кредити;
- списък на изборните дисциплини, които трябва да са издържани предварително;
- максимален брой слушатели;
- преподавател;
- списък на записаните студенти;
- резултати от текущ контрол за участниците.

и метод ***EnrollmentStatus enroll (Student s),*** който се използва за записване в текущата изборна дисциплина на студента, указан с парметъра s, като връща статуса на резултата – дали записването е успешно и ако не, то по каква причина (изчерпване на капацитета, повторно записване на същата дисциплина, неизпълнено условие за предварително издържани изпити).
Клас **Faculty** за представяне на данните за Факултета със следните полета:
- списък от всички преподаватели във Факултета;
- списък на студентите във Факултета;
- каталог на изборните дисциплини;

и следните методи:

- метод ***initilalizeCatalog(String fileName),*** който служи за първоначално попълване на каталога на изборните дисциплини. Информацията за целта се извлича от текстовия файл с име, указано с параметъра на метода. Той съдържа за всяка дисциплина следните данни: идентификатор, име и кредити, разположени на един ред във файла;
- метод ***failedInCourses(Student s),*** който връща списък на всички изборни дисциплини, посещавани от указания с параметъра студент, такива че текущият контрол за съответната дисциплина е незадоволителен (<3).

Забележка: Освен явно указаните методи, горните класове да бъдат снабдени с необходимите конструктори и методи, чието използване се налага.

Задача 2: Ненаредена конфигурация с повторение /**мултимножество**/ ще наричаме множество от наредени двойки от вида: $\{(e_1,n_1),(e_2,n_2),...,(e_k,n_k)\}$, където $e_1,e_2,...,e_k$ са две по две различни – ще ги наричаме елементи на мултимножеството, а n_i са числа, които показват броя на повторенията /кратността/ на $e_i, i=1,2,...,k$ в мултимножеството.

Да се напише клас **MultySetMap** на езика Java, който реализира интерфейса **MultySet** и представя мултимножество:

interface *MultySet<E>* extends *Cloneable* {
public boolean isEmpty() {…} - проверява дали мултимножеството е празно;
public boolean equals(MultySet<E> other) {…} - проверява равенство на мултимножества;
public int contains(E e) {…} - връща кратността на елемента **e,** ако принадлежи на мултимножеството, иначе връща нула;
public MultySetMap<E> clone() – създава копие на обекта
public int size() {…} - връща броя на елементите на мултимножеството, всеки преброен толкова пъти, колкото е кратността му;
public void clear() {…} - унищожава всички елементи на мултимножеството;
public void add(E e, int card) {…} - добавя към мултимножеството елемент **e** с кратност **card**;

public int removeElement (E e) {…} - отстранява от мултимножеството елемента **e** и връща неговата кратност;
public int removeInstance (E e) {…} - отстранява от мултимножеството един екземпляр на елемента **e** и връща неговата кратност след промяната;
public boolean subSetOf(MultySet<E> other) {…} - проверява дали обектът **this** е подмножество на **other**;
public MultySet<E> union(MultySet<E> other) {…} - намира обединението на обектите **this** и **other;**
public MultySet<E> intersection(MultySet<E> other) {…} - намира сечението на обектите **this** и **other;**
public MultySet<E> difference(MultySet<E> other) {…} - намира разликата на обектите **this** и **other;**
public Iterator<E> iterator() {…} - връща итератор за мултимножеството
public E[] toArray() – връща масив, съдържащ елементите на множеството;
public String toString() – припокрива метода toString() на класа Object

Класът да съдържа и следните конструктори:
public MultySetMap() {…} - създава празно мултимножество;
public MultySetMap(E [] array) {…} - създава мултимножество от елементите на масива array

Забележка: За реализиране на класа **MultySetMap** да се използва класа ***java.util.HashMap.***