МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Лабораторна робота №3

з курсу

«Інженерія даних та знань»

*на тему:*

**«Розробка інтерфейсу користувача для системи, основаної на**

**знаннях про предметну область»**

Виконав:

ст. гр. ITIT-12

Галах А.Б.

Прийняла:

Рибчак З.Л.

**Львів 2017**

**Мета роботи:**

Придбання і закріплення навиків в роботі по моделюванню предметних областей, інтеграції даних, в модульному програмуванні.

**Індивідуальне завдання:**

**Варіант 3**

Для заданої прикладної області розробити опис об'єктів цієї

області. Розробити процедури, що реалізують базові операції над цими

об'єктами, зокрема:

* текстове введення-виведення (консольний і файловий);
* присвоювання;
* видалення;
* редагування
* задання константних значень;
* порівняння (не менше 2-х типів);
* пошук за шаблоном.

Підготувати файл початкових даних, що містять не менше 10

значень конкретних об'єктів. В якості формату даних/файлів

використовуйте CSV або XML.

Використовуючи процедури і описи типу даних, розробити

програму, що забезпечує введення та зчитування початкових даних з

першого файлу даних в пам'ять і зберігання їх в масиві, сортування

масиву в алфавітному і за числовим параметром.

Для кожної предметної області перераховані параметри об'єкту. Серед

параметрів обов'язково є ключове алфавітне поле (наприклад,

прізвище), яке ідентифікує об'єкт, у кожного об'єкту є також одне або

декілька числових полів, по яким вірогідні звернення до об'єкту. Набір

характеристик може бути розширений і ускладнений по розсуду

виконавця.

|  |  |  |
| --- | --- | --- |
| **№** |  | **Завдання** |
| 3 | Спортивні  змагання | прізвище спортсмена, ім'я, команда, вид  спорту, заліковий результат, штрафні очки |

**Порядок виконання:**

Було розроблено опис об’єктів заданої предментної області. Для збереження нікапсуляції всі поля об’єктів мають private ідентифікатор доступу. Було реалізовано наступні класи, що відображають предметну область:

1. Sportsman – відображає базову інформацію про спортсмена
2. Team – відображає інформацію про команду
3. SprtType – відображає тип спорту (було реалізовано в виді enum, оскільки присутня фіксована кількість варіантів значень)

Було реалізовано зчитування початкових даних з файлу формату XML. Для сортування даних класи імплементують інтерфейс Comparable імплементація якого дозволяє сортувати дані класи за допомого статичного методу sort класу Collections.

Код програми:

Клас Sportsman.java

**public class** Sportsman **implements** Comparable<Sportsman>{  
  
 **private int mFinePoints**;  
 **private int mResult**;  
 **private** String **mName**;  
 **private** String **mSurname**;  
 **private** Team **mTeam**;  
 **private** SportType **mSportType**;  
  
 **public** Sportsman() {  
 }  
  
 **public** Sportsman(**int** finePoints, **int** result, String name, String surname, SportType sportType) {  
 **this**.**mFinePoints** = finePoints;  
 **this**.**mResult** = result;  
 **this**.**mName** = name;  
 **this**.**mSurname** = surname;  
 **this**.**mSportType** = sportType;  
 }  
  
 **public int** getFinePoints() {  
 **return mFinePoints**;  
 }  
  
 **public void** setFinePoints(**int** finePoints) {  
 **this**.**mFinePoints** = finePoints;  
 }  
  
 **public int** getResult() {  
 **return mResult**;  
 }  
  
 **public void** setResult(**int** result) {  
 **this**.**mResult** = result;  
 }  
  
 **public** String getName() {  
 **return mName**;  
 }  
  
 **public void** setName(String name) {  
 **this**.**mName** = name;  
 }  
  
 **public** String getSurname() {  
 **return mSurname**;  
 }  
  
 **public void** setSurname(String surname) {  
 **this**.**mSurname** = surname;  
 }  
  
 **public** Team getTeam() {  
 **return mTeam**;  
 }  
  
 **public void** setTeam(Team team) {  
 **this**.**mTeam** = team;  
 }  
  
 **public** SportType getSportType() {  
 **return mSportType**;  
 }  
  
 **public void** setSportType(SportType sportType) {  
 **this**.**mSportType** = sportType;  
 }  
  
 @Override  
 **public int** compareTo(Sportsman o) {  
 **return mName**.compareTo(o.getName());  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Sportsman{"** +  
 **"mFinePoints="** + **mFinePoints** +  
 **", mResult="** + **mResult** +  
 **", mName='"** + **mName** + **'\''** +  
 **", mSurname='"** + **mSurname** + **'\''** +  
 **", mSportType="** + **mSportType** +  
 **'}'**;  
 }  
}

Enum SportType.java

**public enum** SportType {  
 ***Football***(1),  
 ***Basketball***(2),  
 ***Volleyball***(3),  
 ***Tenis***(4),  
 ***Unknown***(-1);  
  
 **private int mId**;  
  
 SportType(**int** id) {  
 **mId** = id;  
 }  
  
 **public int** getId() {  
 **return mId**;  
 }  
  
 **public static** SportType fromId(**int** id) {  
 **for** (SportType type : *values*()) {  
 **if** (type.getId() == id) {  
 **return** type;  
 }  
 }  
 **return *Unknown***;  
 }  
}

Клас Team.java

**public class** Team **implements** Comparable<Team>{  
  
 **private int mId**;  
 **private** String **mName**;  
 **private** SportType **mType**;  
 **private** List<Sportsman> **mMembers** = **new** ArrayList<>();  
  
 **public** Team() {  
 }  
  
 **public** Team(**int** id, String name, SportType type) {  
 **mId** = id;  
 **mName** = name;  
 **mType** = type;  
 }  
  
 **public int** getId() {  
 **return mId**;  
 }  
  
 **public void** setId(**int** id) {  
 **mId** = id;  
 }  
  
 **public** String getName() {  
 **return mName**;  
 }  
  
 **public void** setName(String name) {  
 **mName** = name;  
 }  
  
 **public** SportType getType() {  
 **return mType**;  
 }  
  
 **public void** setType(SportType type) {  
 **mType** = type;  
 }  
  
 **public boolean** isMember(Sportsman sportsman) {  
 **return mMembers**.indexOf(sportsman) != -1;  
 }  
  
 **public** List<Sportsman> getMembers() {  
 **return mMembers**;  
 }  
  
 **public void** setMembers(@NotNull List<Sportsman> members) {  
 **mMembers** = members;  
 }  
  
 **public void** addMember(Sportsman sportsman) {  
 **mMembers**.add(sportsman);  
 }  
  
 **public void** removeMember(Sportsman sportsman) {  
 **mMembers**.remove(sportsman);  
 }  
  
 @Override  
 **public int** compareTo(Team o) {  
 **return mName**.compareTo(o.getName());  
 }  
  
 @Override  
 **public** String toString() {  
 StringBuilder builder = **new** StringBuilder();  
 **mMembers**.forEach(sportsman -> builder.append(sportsman.toString()).append(**"\n"**));  
 **return "Team{"** +  
 **"mId: "** + **mId** +  
 **", mName: '"** + **mName** + **'\''** +  
 **", mType: "** + **mType** +  
 **", mMembers:\n"** + builder.toString() +  
 **'}'**;  
 }  
}

Клас Main.java

**public class** Main {  
  
 **private static** String *XML\_FILE* = **"G:\\навчання\\5 курс\\Інженерія даних та знань\\lab3\\lab3\\xmlFile.xml"**;  
  
 **public static void** main(String[] args) **throws** Exception {  
 List<Team> teams = *readFromXml*();  
 Collections.*sort*(teams);  
 teams.forEach(team -> Collections.*sort*(team.getMembers()));  
 teams.forEach(team -> System.***out***.println(team.toString()));  
 }  
  
 **private static** List<Team> readFromXml() **throws** Exception {  
 List<Team> elementList = **new** ArrayList<>();  
  
 File fXmlFile = **new** File(*XML\_FILE*);  
 DocumentBuilderFactory dbFactory = DocumentBuilderFactory.*newInstance*();  
 DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();  
 Document doc = dBuilder.parse(fXmlFile);  
 doc.getDocumentElement().normalize();  
  
 Element node = doc.getDocumentElement();  
 NodeList nList = node.getElementsByTagName(**"Team"**);  
 **for** (**int** i = 0; i < nList.getLength(); i++) {  
 Team team = *fromNode*(nList.item(i));  
 **if** (team != **null**) {  
 elementList.add(team);  
 }  
 }  
  
 **return** elementList;  
 }  
  
 **private static** Team fromNode(Node node) {  
 **if** (node.getNodeType() != Node.***ELEMENT\_NODE***) {  
 **return null**;  
 }  
 Element element = (Element) node;  
 Team team = **new** Team();  
 team.setId(Integer.*parseInt*(element.getAttribute(**"id"**)));  
 team.setName(element.getAttribute(**"name"**));  
 team.setType(SportType.*fromId*(Integer.*parseInt*(((Element) node).getAttribute(**"sportType"**))));  
  
 NodeList membersNodes = element.getElementsByTagName(**"SportsMan"**);  
 **for** (**int** i = 0; i < membersNodes.getLength(); i++) {  
 Sportsman sportsman = *getSportsmanFromNode*(membersNodes.item(i));  
 **if** (sportsman != **null**) {  
 team.addMember(sportsman);  
 sportsman.setTeam(team);  
 }  
 }  
 **return** team;  
 }  
  
 **private static** Sportsman getSportsmanFromNode(Node node) {  
 **if** (node.getNodeType() != Node.***ELEMENT\_NODE***) {  
 **return null**;  
 }  
 Element element = (Element) node;  
 Sportsman sportsman = **new** Sportsman();  
 sportsman.setName(element.getAttribute(**"name"**));  
 sportsman.setSurname(element.getAttribute(**"surname"**));  
 sportsman.setSportType(SportType.*fromId*(Integer.*parseInt*(element.getAttribute(**"sportType"**))));  
 sportsman.setResult(Integer.*parseInt*(element.getAttribute(**"result"**)));  
 sportsman.setFinePoints(Integer.*parseInt*(element.getAttribute(**"fine"**)));  
 **return** sportsman;  
 }  
  
}

**Результат виконання програми**

Результати виконання програми представлений на Рис.1. Як видно з рисунку дані були зчитані з початкового файлу та відсортовані в алфавітному порядку з ключовими параметрами, зокерма назви команди і ім’я спортсмена.

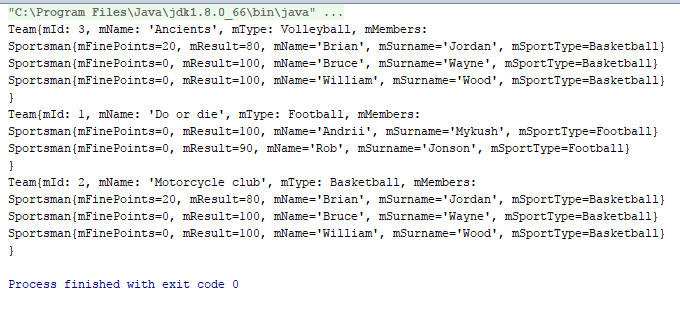


Рис.1 Результат виконання програми

**Висновок:** В результаті виконання даної лабораторної роботи було розроблено додаток для зчитування вхідних даних з файлів формату XML предметної області та сортування їх за ключовими параметрами.