Київський національний університет імені Т. Шевченка
Факультет комп'ютерних наук та кібернетики

Паралельні обчислення
Лабораторна робота №1
Звіт

Виконав: студент групи ІПС-31
Кравчук Павло

Київ-2020

Умова завдання:

Розробити програму, що забезпечує ввід і редагування інформації про об’єкти згідно з заданою предметною областю. Інформація про об’єкти має зберігатись в окремому файлі формату XML.

Варіант 8

Предметна область: Футбол

Об'єкти: Команди, Гравці

Файл Player.java:

```
public class Player {
    public int code;
    public String name;
    public boolean isCaptain;
    public int salary;

    String getXml() {
        return "\t\t<player id=\"" + code + "\" name=\"" + name + "\" isCaptain=\""
                + ((isCaptain) ? "1" : "0") + "\" salary=\"" + salary + "\"/>\n";
    }

    public String toString() {
        return "\t" + name + "\n";
    }
}
```

Клас гравця, містить поля для унікального ідентифікатору, імені, статусу капітана команди та зарплатні. Методи гравця: метод повернення строки з відомостями гравця у вигляді xml та звичайний метод приведення до строки, який повертає лише ім’я.

Файл Team.java:

```
public class Team {         //don't use constructor since we're filling elements one
by one
    public int code;
    public String name;
    public List<Player> players = new ArrayList<>();

    String getXml() {
        StringBuilder result = new StringBuilder("\t<team id=\"" + code + "\"
name=\"" + name + "\">\n");

        for (Player player : players) {
            result.append(player.getXml());
        }

        result.append("\t</team>\n");
```

```
        return result.toString();
    }

    public String toString() {
        StringBuilder result = new StringBuilder(name + ":\n");

        for (Player player : players) {
            result.append(player.toString());
        }

        return result.toString();
    }
}
```

Клас команди, містить поля для унікального ідентифікатору, назви та гравців в команді. Методи команди: метод повернення строки з відомостями команді та гравців у вигляді xml та звичайний метод приведення до строки, який повертає лише назву.

Файл FootballParser.java:

```
import java.util.ArrayList;
import java.util.List;

public class FootballParser extends DefaultHandler {
    List<Team> teams = new ArrayList<>();
    List<Player> allPlayers = new ArrayList<>();
    Team currTeam;
    Player currPlayer;
    int playerMaxId = 1;
    int teamMaxId = 1;

    List<Team> getTeams() {
        return teams;
    }

    List<Player> getAllPlayers() {
        return allPlayers;
    }

    int getPlayerMaxId() {
        return playerMaxId;
    }

    int getTeamMaxId() {
        return teamMaxId;
    }

    @Override
    public void startElement(String namespaceURI, String localName, String qName,
Attributes attrs) {
        switch (qName) {
            case "team":
                currTeam = new Team();
                break;
            case "player":
                currPlayer = new Player();
                break;
        }

        int attributeLength = attrs.getLength();
```

```
        for (int i = 0; i < attributeLength; i++) {
            String attrName = attrs.getQName(i);
            String attrVal = attrs.getValue(i);

            switch (qName) {
                case "team":
                    switch (attrName) {
                        case "id":
                            currTeam.code = Integer.parseInt(attrVal);
                            if (currTeam.code > teamMaxId)
                                teamMaxId = currTeam.code;
                            break;
                        case "name":
                            currTeam.name = attrVal;
                            break;
                    }
                    break;

                case "player":
                    switch (attrName) {
                        case "id":
                            currPlayer.code = Integer.parseInt(attrVal);
                            if (currPlayer.code > playerMaxId)
                                playerMaxId = currPlayer.code;
                            break;
                        case "name":
                            currPlayer.name = attrVal;
                            break;
                        case "isCaptain":
                            currPlayer.isCaptain = attrVal.equals("1");
                            break;
                        case "salary":
                            currPlayer.salary = Integer.parseInt(attrVal);
                            break;
                    }
                    break;
            }
        }
    }

    public void endElement(String namespaceURI, String localName, String qName) {
        switch (qName) {
            case "team":
                teams.add(currTeam);
                break;
            case "player":
                currTeam.players.add(currPlayer);
                allPlayers.add(currPlayer);
                break;
        }
    }
}
```

Клас що виконує задачу зчитування даних про команди та гравців із xml файлу. Функція startElement створює та записує в поля гравців та команди які зараз оброблюються а також заповнює поля новостворених гравця чи команди. Функція endElement додає до списку гравців чи команд відповідно гравця чи команду. Також наявні функції get для результуючих даних.

Файл Football.java:

```
import org.xml.sax.SAXException;

import javax.xml.XMLConstants;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import javax.xml.transform.stream.StreamSource;
import javax.xml.validation.Schema;
import javax.xml.validation.SchemaFactory;
import javax.xml.validation.Validator;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.List;

public class Football {
    private List<Team> teams;
    private List<Player> players;
    private int playerMaxId;
    private int teamMaxId;

    public void saveToFile(String filename) {
        StringBuilder result = new StringBuilder("<?xml version=\"1.0\"
encoding=\"WINDOWS-1251\"?>\n\n<football>\n");

        for (Team team : teams) {
            result.append(team.getXml());
        }

        result.append("</football>");

        try (FileWriter fr = new FileWriter(filename)) {
            fr.write(result.toString());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void loadFromFile(String filename) throws ParserConfigurationException,
SAXException, IOException {
        SAXParserFactory factory = SAXParserFactory.newInstance();
        SAXParser parser = factory.newSAXParser();
        FootballParser fp = new FootballParser();

        try {
            SchemaFactory scFactory =
                    SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
            Schema schema = scFactory.newSchema(new File("football.xsd"));
            Validator validator = schema.newValidator();
            validator.validate(new StreamSource(new File(filename)));
        } catch (IOException | SAXException e) {
            e.printStackTrace();
            return;
        }

        System.out.println("Validation success");

        parser.parse(new File(filename), fp);

        teams = fp.getTeams();
        players = fp.getAllPlayers();
        playerMaxId = fp.getPlayerMaxId();
        teamMaxId = fp.getTeamMaxId();
    }
```

```
    public void addTeam(int code, String name) throws Exception {
        for (Team team : teams) {
            if (team.code == code)
                throw new Exception();
        }

        Team team = new Team();
        team.code = code;
        team.name = name;
        teams.add(team);
        if (code > teamMaxId)
            teamMaxId = code;
    }

    public Team getTeam(int code) throws Exception {
        for (Team team : teams) {
            if (team.code == code)
                return team;
        }

        throw new Exception();
    }

    public Team getTeamInd(int index) throws Exception {
        if (index >= teams.size())
            throw new Exception();

        return teams.get(index);
    }

    public int countTeams() {
        return teams.size();
    }

    public void deleteTeam(int code) throws Exception {
        for (Team team : teams) {
            if (team.code == code) {
                for (Player player : team.players) {
                    players.remove(player);
                }
                teams.remove(team);
                return;
            }
        }

        throw new Exception();
    }

    int getTeamMaxId() {
        return teamMaxId;
    }

    public void addPlayer(int code, String name, boolean isCaptain, int salary, int
teamCode) throws Exception {
        for (Player player : players) {
            if (player.code == code)
                throw new Exception();
        }

        Team team = getTeam(teamCode);      //will raise exception itself

        Player player = new Player();
        player.code = code;
        player.name = name;
        player.isCaptain = isCaptain;
        player.salary = salary;
```

```
        team.players.add(player);
        players.add(player);
        if (code > playerMaxId)
            playerMaxId = code;
    }


    public Player getPlayer(int code) throws Exception {
        for (Player player : players) {
            if (player.code == code)
                return player;
        }

        throw new Exception();
    }

    public Player getPlayerInd(int index) throws Exception {
        if (index >= players.size())
            throw new Exception();

        return players.get(index);
    }

    public int countPlayers() {
        return players.size();
    }

    public void deletePlayer(int code) throws Exception {
        for (Player player : players) {
            if (player.code == code) {

                for (Team team : teams) {
                    if (team.players.indexOf(player) > -1) {
                        team.players.remove(player);
                        break;
                    }
                }
                players.remove(player);

                return;
            }
        }

        throw new Exception();
    }

    int getPlayerMaxId() {
        return playerMaxId;
    }
}
```

Клас що зберігає в собі дані про гравців та команди а також керує ними. Наявні функції зберігання, видалення, отримання за id, отримання за індексом, отримання кількості гравців та команд. Також наявні функції завантаження даних з файлу та їх збереження в файл.

Приклад xml:

```
<?xml version="1.0" encoding="WINDOWS-1251"?>

<football xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="football.xsd">
```

```
    <team id="1" name="Шахтер">
        <player id="1" name="Евгений" isCaptain="0" salary="400000"/>
        <player id="2" name="Андрей" isCaptain="1" salary="900000"/>
        <player id="3" name="Николай" isCaptain="0" salary="525000"/>
    </team>
    <team id="2" name="Динамо">
        <player id="4" name="Виктор" isCaptain="1" salary="450000"/>
        <player id="5" name="Артем" isCaptain="0" salary="250000"/>
    </team>
</football>
```

XSD схема:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="football">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="team" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="player" maxOccurs="unbounded">
                                <xs:complexType>
                                    <xs:attribute name="id" type="xs:integer" use
="required"/>
                                    <xs:attribute name="name" type="xs:string" us
e="required"/>
                                    <xs:attribute name="isCaptain" type="xs:boole
an" use="required"/>
                                    <xs:attribute name="salary" type="xs:integer"
 use="required"/>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>

                        <xs:attribute name="id" type="xs:integer" use="required"/
>
                        <xs:attribute name="name" type="xs:string" use="required"
/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```