Introdução à Linguagem Java

Lista 09

Instruções

Vale a mesma observação da aula anterior, o uso de ferramentas de busca como o Google, a documentação do Java[1], sites sobre programação como o StackOverflow, ou mesmo o Wikipedia será essencial, pois a ideia é vocês pesquisarem como resolver um problema. Por isso, antes de tentarem resolver algum exercício, procurem um pouco e vejam como resolvê-lo de maneira correta

Exercício 1 (3,0)

Crie um programa que compara o desempenho entre imprimir um texto qualquer num arquivo e imprimir o mesmo texto na saída do terminal. Qual é mais rápido? Explique a diferença de desempenho.

Dica: use um laço com várias iterações para observar melhor o resultado.

Exercício 2 (5,0)

Considere os seguintes códigos[2]:

Primeiro, a classe Student:

```
public class Student {
   private String name;
   private int rollNo;

   Student(String name, int rollNo){
      this.name = name;
      this.rollNo = rollNo;
   }

   public String getName() {
      return name;
   }
```

[1] https://docs.oracle.com/javase/8/docs/api/

[2] http://www.tutorialspoint.com/design_pattern/data_access_object_pattern.htm

```
    public void setName(String name) {
        this.name = name;
    }

    public int getRollNo() {
        return rollNo;
    }

    public void setRollNo(int rollNo) {
        this.rollNo = rollNo;
    }
}
```

Em seguida, uma interface DAO (Data Access Object).

```
import java.util.List;

public interface StudentDao {
    public List<Student> getAllStudents();
    public Student getStudent(int rollNo);
    public void updateStudent(Student student);
    public void deleteStudent(Student student);
}
```

Considere então uma implementação concreta desta interface, usando uma lista como banco de dados.

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

public class StudentDaoImpl implements StudentDao {
    //list is working as a database
    List<Student> students;

    public StudentDaoImpl(){
        students = new ArrayList<Student>();
        Student student1 = new Student("Robert",0);
        Student student2 = new Student("John",1);
        students.add(student1);
        students.add(student2);
    }
    @Override
    public void deleteStudent(Student student) {
        students.remove(student.getRollNo());
        System.out.println("Student: Roll No " + student.getRollNo() +
                                    ", deleted from database");
    }
```

```
    //retrive list of students from the database
    @Override
    public List<Student> getAllStudents() {
       return students;
    }

    @Override
    public Student getStudent(int rollNo) {
       return students.get(rollNo);
    }

    @Override
    public void updateStudent(Student student) {
       students.get(student.getRollNo()).setName(student.getName());
       System.out.println("Student: Roll No " + student.getRollNo() +
                                          ", updated in the database");
    }
}
```

Finalmente, um exemplo de `main` usando DAO.

```
public class DaoPatternDemo {
    public static void main(String[] args) {
       StudentDao studentDao = new StudentDaoImpl();

       //print all students
       for (Student student : studentDao.getAllStudents()) {
          System.out.println("Student: [RollNo : " + student.getRollNo() +
                                ", Name : " + student.getName() + " ]");
       }


       //update student
       Student student =studentDao.getAllStudents().get(0);
       student.setName("Michael");
       studentDao.updateStudent(student);

       //get the student
       studentDao.getStudent(0);
       System.out.println("Student: [RollNo : " + student.getRollNo() +
                                ", Name : " + student.getName() + " ]");
    }
}
```

Agora, implemente uma nova classe que implementa a interface `StudentDao` para ler e salvar os dados dos estudantes a partir de um arquivo de texto plano.

Teste a classe criada (e somente ela) usando o JUnit[3].

Dica 1: Para facilitar, você pode escrever no arquivo usando o formato `.csv` (comma-separated values)[4]. Isso facilitará a análise posterior dos dados. Você pode usar classes pré-existentes que manipulam arquivos CSV em Java. Cuidado, existem DOIS formatos CSV (por quê?), e se seu manipulador de planilhas estiver em português, isso pode gerar um problema de visualização de dados.

Dica 2: Desde que você não perca dados, você pode sobrescrever o arquivo original ao invés de tentar modificá-lo a cada alteração.

Exercício 3 (2,0)

Crie um programa que mostre a diferença entre `PrintWriter` e `OutputStreams`. Por que essa diferença ocorre?

Dica: uma possível maneira de mostrar essa diferença seria usando um gerador de bytes aleatórios como o `Random.nextBytes()` e usar a sua saída nos métodos de escrita das classes. Obviamente, para uma comparação adequada é necessário usar os mesmos bytes gerados como entrada para as duas classes.

[3]Não é necessário testar se o arquivo é criado de fato, mas é necessário testar se você está lendo do arquivo corretamente. Se for necessário, pode incluir um arquivo de exemplo junto com o código do programa de vocês.

[4]`https://en.wikipedia.org/wiki/Main_Page`