Java 포트폴리오

- ATM : Singleton 패턴 구현
- MyArrayList : java.util.ArrayList 직접 구현
- 학생 관리 : Spring의 구조를 java로 간접 구현,
  getter & setter 메서드 적용

contents

Ⅰ. ATM

01 설명

- UserManager는 Singleton으로 작성하여 User의 계좌 정보가 한 곳에만 저장 및 호출 되도록 하였습니다.
- 회원가입을 하거나 계좌를 삭제하는 등의 메서드를 UserManager에서 작성하였고 이를 ATM 클래스에서는 호출만 하도록 구현하였습니다.
- FileManager도 Singleton으로 작성하여 User의 계좌 정보는 한 번 저장되도록 합니다.

Ⅰ. ATM

02 코드

Java | UserManager

```java
private UserManager() {}
private static UserManager instance = new UserManager();
public static UserManager getInstance() {
    return instance;
}
```

- 로그인 여부를 알 수 있는 log의 경우에는 ATM에서 static으로 선언해서 ATM.log로 정보를 가져올 수 있다.

- Singleton으로 작성

- 사용자 로그인, 가입, 탈퇴 기능
- 계좌 추가, 삭제, 이체, 출금 기능

Ⅰ. ATM

02 코드

Java | FileManager

```
private FileManager () {}
private static FileManager instance = new FileManager();
public static FileManager getInstance() {
    return instance;
}
```

- Singleton으로 작성
- Save, load 기능

Ⅰ. ATM

02 코드

save_data

```
User user1 = UserManager.getInstance().user[i];

if (user1.accCount > 0) {
    data += "/";
    for (int j=0;j<user1.accCount;j++) {
        Account acc1 = user1.acc[j];

if (i != UserManager.getInstance().userCount - 1) {
    data += "\n";
}
```

- UserManager에서 user의 정보를 가져온 뒤 저장합니다.
- /로 구분하여 저장하고 계좌의 개수가 User마다 다르기 때문에 accCount 변수를 활용합니다.
- User 한 명에 대한 정보가 다 저장되면 띄어쓰기로 구분을 해줍니다.

Ⅰ. ATM

02 코드

load_data

```
UserManager.getInstance().user = new User[count];
UserManager.getInstance().userCount = count;
```

- 저장한 User의 정보를 load합니다.
- 이름, 패스워드, 계좌 개수에 대한 코드와 실제 저장된 txt파일 예시입니다.

```
UserManager.getInstance().user[i] = new User();
UserManager.getInstance().user[i].name = temp[0];
UserManager.getInstance().user[i].pw = temp[1];
UserManager.getInstance().user[i].accCount = Integer.parseInt(temp[2]);
```

```
2
qwer/qwer/1/1234/8000
java/java/2/2345/10000/3456/5000
```

II. MyArrayList

01 설명

- java.util.ArrayList를 직접 구현하였습니다.
- User클래스와 MyVector 클래스를 만들고 main에서 MyVector<User>에 대한 객체인 vec과 User의 객체인 temp를 선언합니다.
- 이후 MyVector에서 add 메서드를 작성하여 ArrayList에서의 add기능을 구현하도록 합니다.

II. MyArrayList

02 코드

Java | main

```
public static void main(String[] args) {
    MyVector<User> vec = new MyVector<User>();
    User temp = new User();
    temp.name = "철수";
    vec.add(temp);

    temp = new User();
    temp.name = "민수";
    vec.add(temp);

    for (int i=0;i<vec.size();i++) {
        vec.get(i).print_name();
    }
}
```

User / MyVector

```
class User {
    String name;
    void print_name() {
        System.out.println(name);
    }
}

class MyVector<T> {
```

II. MyArrayList

02 코드

Java | MyVector

```
void add(T data) {
    if (count == 0) {
        arr = new Object[1];
    }
    else {
        Object[] temp = arr;
        arr = new Object[count + 1];
        for (int i=0;i<count;i++) {
            arr[i] = temp[i];
        }

        temp = null;
    }

    arr[count] = data;
    count++;
}
```

- private Object arr[] = null;
- private int count = 0;

- 아직 배열에 count가 0이면 바로 값을 넣어주고, 그렇지 않으면 temp를 이용하여 값을 넣어줍니다.

III. 학생 관리

01 설명

- Spring의 개념을 java로 간접 구현하여 객체지향의 개념과 DB의 CRUD의 개념을 이해하기 위한 기초가 되는 예제입니다.

- 최종적으로는 StudentDAO에서 학생 정보의 삽입, 수정, 삭제가 이루어지도록 하는데 그 과정에서 Controller를 두어 관리를 하도록 구현하였습니다.

- Insert를 예시로 설명을 하도록 하겠습니다.

- 1) Controller에서 StudentController의 객체를 생성합니다.
- 2) StudentController에서 getter & setter 메서드로 작성된 insert를 가져옵니다.
- 3) StudentInsert에서의 insert 메서드를 호출합니다.
- 4) StudentDAO에서 HashMap으로 정보를 저장합니다.

III. 학생 관리

02 코드

Java | Controller

```
StudentController controller = new StudentController();
StudentInsert stInsert = controller.getInsert();
for (int i=0;i<ids.length;i++) {
    Student st = new Student(nums[i], ids[i], names[i]);
    stInsert.insert(st);
}
```

- StudentController의 객체 생성
- get/set메서드로 작성된 insert를 가져옵니다.

```
public StudentController() {
    stDAO = new StudentDAO();
    insert = new StudentInsert(stDAO);
    delete = new StudentDelete(stDAO);
```

```
public StudentInsert getInsert() {
    return insert;
}
public void setInsert(StudentInsert insert) {
    this.insert = insert;
}
```

02 코드

Java | StudentInsert

```java
public void insert(Student st) {
    String id = st.getId();
    if (checkId(id)) {
        stDAO.insert(st);
    }
    else {
        System.out.println("중복 아이디 입니다.");
    }
}

public boolean checkId(String id) {
    Student st = stDAO.select(id);
    return st == null ? true : false;
}
```

StudentDAO

```java
private HashMap<String, Student> stDB
public void insert(Student st) {
    stDB.put(st.getId(), st);
}
```

- 이후에는 StudentInsert에서 insert 메서드가 호출됩니다.
- 중복 Id 체크를 한 뒤 StudentDAO 에서 HashMap으로 학생 정보를 저장하도록 구현하였습니다.

감사합니다.

지원자 최지완 드림