//대본 - 별희

다음은 StList 클래스입니다. 학생의 정보를 저장하려고 만든 클래스입니다.

학생 정보를 어떤 배열 또는 자료구조에 담아야 할지가 첫번째 고민이었습니다.

먼저 로그인을 할 때에 학번과 이름으로 하게 되므로 그 둘을 연결지을 수 있는 자료였으면 좋겠다는게 1번. 학번은 겹칠 수 없다가 2번. 동명이인은 가능하다 3번 까지가 최소한의 조건이었습니다.

HashMap 자료구조의 경우 key와 value가 연결되어 있기 때문에 학번과 이름을 연결지을 수 있고, key 값은 중복이 허용되지 않지만 value 값은 허용이 된다는 점에서 HashMap을 선택하게 되었습니다.

그리고 static 초기화로 이미 증명서발급기를 사용하고 있다는 설정의 기존 학생들의 정보가 HashMap 안에 저장되어 있게 하려고 했습니다.

>>클릭<<

—-----------------------------------------------------------------------------------------------------------------------

StList 클래스의 기능으로는 학생정보 즉, HashMap 직렬화, 조회, 추가, 수정, 삭제가 있습니다. 사실상 관리자 클래스에서 구현되어야 할 것 같 메소드들이 이 클래스에 있는데요. >>클릭<<

—------------------------------------------------------------------------------------------------------------------------

처음엔 StList 클래스에서 학생 정보를 담을 HashMap 자료구조를 생성하고

>> 클릭<<

관리자 클래스인 AdminMenu 클래스에서 StList클래스 인스턴스를 생성해서 map에 접근해 학생을 추가한다든가 이런 식으로 관리하려고 했습니다. 학생 정보에 접근하려면 매번 새롭게 클래스 인스턴스를 생성해야 했으니 당연하게도 변경된 값들이 공유되지 않았습니다.

>>클릭<<

—------------------------------------------------------------------------------------------------------------------------

결국 직렬화 메소드를 만들었고 info.ser 라는 파일에 학생정보를 담은 자료구조 map을 저장했습니다.

>>클릭<<

—------------------------------------------------------------------------------------------------------------------------

이 메소드는 학생 정보를 조회합니다. HashMap 자료구조가 가지고 있는 iterator() 메소드를 이용해 key와 value 값을 출력했습니다.

>>클릭<<

—------------------------------------------------------------------------------------------------------------------------

근데 넘 길면 걍 메소드 몇개는 스킵할수도있어여—-----

AdminMenu클래는 관리자의 기능을 담고 있는 클래스입니다.

// 멤버 변수

멤버 변수에는 관리자 고유 비밀번호가 있는데 이건 private으로 선언과 초기화를 해두었고 아래는 역직렬화로 읽어들인 값을 담는 변수들입니다.

// adLogin()

이 메소드는 관리자 로그인을 인증하는 기능입니다.

사용자에게 비밀번호를 입력받고 멤버 변수로 지정해놓은 값과 비교해 3번이상 틀리면 실행이 끝나 메인의 while문이 다시 돌아가도록 되어있습니다.

입력값을 받아 아래의 메뉴선택 메소드를 호출합니다.

// adSelect()

Switch문으로 메뉴를 선택하게 했습니다.

// readCashSerial(), readCardSerial(), readAccountSerial, readPaperSerial()

이 다음 메소드들은 현금, 카드 수입, 계좌이체 수입, 종이 재고 객체를 저장한 파일에 담긴 객체를 읽어서 멤버 변수에 담습니다.

//paperSerial()

이 메소드는 종이 재고 객체를 파일에 저장하는 기능을 합니다.

// adIncome()

이 메소드는 수입을 조회하는 기능을 합니다.

역직렬화 메소드를 호출해서 멤버변수에 값을 담고 각각 출력해줍니다. 여기서 현금은 수입만 출력할 것이므로 readCashTrays 배열의 0번방만 출력합니다. (1번방은 거스름돈)

adBack()을 호출해 로그아웃 또는 뒤로가기를 선택합니다.

// adManage()

이 메소드는 거스름돈을 조회하고 관리하는 기능을 합니다. 조회를 선택하면 현금파일을 조회하고 거스름돈이 담긴 2번방의 배열들만 출력합니다.

조회가 아니면 아래의 adCashFull()를 호출합니다.

// adCashFull()

Switch문으로 권종을 선택해서 정해놓은 수 만큼만 채울 수 있게, 초기화를 해주었습니다.

여기선 조회가 아니라 값이 바뀌었으니 모든 곳에서 공유할 수 있게 저장을 해줍니다. 거스름돈을 직렬화하는 메소드가 이미 BasketandPay에 있으므로 인스턴스 변수로 호출해 저장합니다.

// adPaper()

이 메소드는 종이 재고를 조회하고 관리합니다. 조회는 역직렬화해서 읽어오고 관리는 아래의 adPpFull() 메소드를 호출합니다.

// adPpFull()

채울 장 수를 입력받고 멤버변수에 더해준 후 그 값을 다시 직렬화해서 공유되게끔 합니다.

+ 추가된 기능 추가…(if문)

// adStManage()

이 메소드는 학생 정보를 관리하는 기능인데 지금 보시면 전부 StList 클래스의 인스턴스 변수인 st를 통해서 메소드를 호출해 오는 걸 확인할 수 있습니다.

(문제점) 원래는 관리자 클래스 내에서 기능을 구현하려고 했는데 그 시점에서는 직렬화를 하지 않았어서 조회를 예로 들면 StList 클래스 인스턴스를 새롭게 퍼올려서 그 인스턴스에 있는 HashMap을 읽어 왔습니다. 당연히 새롭게 생성된 HashMap에 담긴 정보를 읽어오게되니 수정하거나 추가, 삭제한 게 전혀 반영이 되지 않았습니다. 이를 어떻게든 해결해보고자 관리 자체를 StList 클래스에서 구현하고 관리자에서는 그 메소드를 호출만 해오면 어떨까 했던게 이어져서 지금의 구조가 되었습니다.

(해결) 직렬화, 역직렬화를 하게 된 지금은 st.으로 메소드를 호출하지 않고 관리자 클래스 내에서 바로 구현이 가능하지만 그대로 두었습니다.

그리고 조회, 수정, 추가, 삭제가 끝나면 아래의 adStManageBackMenu() 를 호출해서

// adStManageBackMenu()

학생 정보 관리로 돌아갈지 관리자 메뉴로 돌아갈지 선택지를 줍니다.

// adExit()

이 메소드는 메인에서 while문으로 계속 돌아가고 있는 것을 이용해 단순 실행이 끝나면 다시 증명서발급기가 실행되게끔 한 것입니다.

// adClose()

이 메소드는 System 클래스에 있는 exit()를 호출해 현재 실행하고 있는 프로세스를 강제 종료합니다.

// adBack()

이 메소드는 위의 관리, 조회가 끝난 후 로그아웃 또는 뒤로가는 선택지를 줍니다.

// stockCheck()

마지막으로 이 메소드는 종이 재고가 일정 수 이하이면 안내메세지를 출력하거나 기기를 종료

이 메소드는 첫 화면에서 실행되어 증명서 발급기를 켜면 관리자가 바로 종이 재고를 파악해 관리할 수 있도록 합니다.