김태경

객체지향 시스템 설계

2022년 5월 23일

**Assignment2**

1. **실행 방법**

WSL과 jdk 17환경에서, ‘assignment2’패키지가 있는 경로로 이동 후

javac ./assignment2/HospitalManagementSystem.java

명령어 통해 클래스를 생성한 후,

java assignment2. HospitalManagementSystem

명령어 통해 코드 실행할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. **구조 설계 및 methods**
   1. **클래스**(Patient, InPatient, OutPatient, HospitalPersonnel, Nurse, Doctor,

HospitalManagementSystem)

**Patient:**

환자의 ID와 이름, 나이, 담당 의사, 처방을 저장하는 클래스이다.

환자의 정보가 담긴 변수들은 private 접근제어자를 통해 다른 클래스에서 접근하지 못하도록 하여 보안성을 높였다.

생성자는 2개로, ID부터 나이까지 입력을 받거나, ID부터 담당 의사까지 입력을 받아 초기화 한다.

**InPatient**:

Patient에서 상속을 받아 작동된다.

환자의 입원일과 퇴원일을 저장한다.

**OutPatient**:

Patient에서 상속을 받아 작동된다.

환자의 진료일을 저장한다.

**HospitalPersonnel**:

추상 클래스로 병원 근로자의 ID와 비밀번호, 이름, 병원 시스템을 저장한다.

생성자는 모든 변수들을 받아 초기화 한다.

**Nurse, and Doctor**:

HospitalPersonnel에서 상속을 받아 작동된다.

**HospitalManagementSystem:**

Main 메소드가 동작하는 클래스이다.

HospitalPersonnel과 Patient를 저장하는 배열들이 있으며, 그 인덱스와 함께 static으로 선언이 되어 있으며, 생성자를 통해 초기화 한다.

* 1. **함수**

**Patient:**

**getPid, getDoctor-** 환자의 아이디와 의사를 리턴한다.

**setDoctor, setPrescription-** 의사와 처방 값을 넣는다.

**toString-** 환자의 기본 정보들을 문자열로 변환하여 리턴한다.

**outputPatientInfo-** toString에서 리턴된 문자열과 담당의사와 처방에 값이 있을 경우 추가하여 같이 출력한다.

**InPatient**:

**setInOutDate**- 입원일과 퇴원일 값을 넣는다.

**toString-** Patient의 toString 함수를 override 하여 기존 toString에서 리턴된 값에 입원일과 퇴원일을 추가하여 리턴한다.

**OutPatient**:

**setAppointment**- 진료일 값을 넣는다.

**toString-** Patient의 toString 함수를 override 하여 기존 toString에서 리턴된 값에 진료일을 추가하여 리턴한다.

**InPatient**:

**setInOutDate**- 입원일과 퇴원일 값을 넣는다.

**toString-** Patient의 toString 함수를 override 하여 기존 toString에서 리턴된 값에 입원일과 퇴원일을 추가하여 리턴한다.

**HospitalPersonnel**:

**getName**- 이름을 리턴한다.

**managePatient-** 추상 메소드로 다른 상속된 클래스에서 override를 통해 기능이 달라진다.

**Nurse**:

**managePatient**- Nurse 전용 메뉴화면을 출력한다.

**updatePatient-** 환자의 ID와 날짜 2개를 파라미터로 받는다. 전달된 아이디로 환자를 찾은 후 날짜가 1개일 경우 진료일을 OutPatient에, 날짜가 2개일 경우 입원일과 퇴원일을 InPatient에 업데이트 한다.

**Doctor**:

**managePatient**- Doctor 전용 메뉴화면을 출력한다.

**updatePatient-** 환자의 ID와 처방을 파라미터로 받는다. 전달된 아이디로 환자를 찾은 후 환자의 처방을 업데이트 한다.

**HospitalManagementSystem:**

**addDoctor-** Doctor를 파라미터로 받은 후 hospitalPersoneelList에 저장하고, 인덱스를 한 칸 증가시킨다.

**addNurse-** Nurse를 파라미터로 받은 후 hospitalPersoneelList에 저장하고, 인덱스를 한 칸 증가시킨다.

**addPatient-** Patient를 파라미터로 받은 후 patientList에 저장하고, 인덱스를 한 칸 증가시킨다.

**outputPatient-** 두개의 함수로 overload 되며, 환자의 ID를 파라미터로 받을 경우 해당 아이디의 환자를 찾아 outputPatientInfo를 실행한다. 파라미터가 없을 경우에는 저장된 모든 환자의 outputPatientInfo를 실행한다.

**login-** 이름과 비밀번호를 파라미터로 받아 hospitalPersonnelList에서 일치하는 객체가 있는지 확인한다. 일치하는 객체가 있을 경우에는 해당 객체를 리턴하고, 없을 경우에는 null값을 리턴한다.

**checkPid-** 환자의 ID를 파라미터로 받아 해당 아이디의 환자를 리턴한다. 환자가 없을 경우에는 null값을 리턴한다.

**getDoctor-** 의사의 이름을 파라미터로 받아 해당 이름의 의사 객체를 리턴한다. 의사를 찾지 못할 경우에는 null값을 리턴한다.

* 1. **Main**

우선 HospitalPersonnel과 Patient 배열을 생성하여 HospitalManagementSystem객체를 생성자와 함께 초기화 한다.

과제 설명의 내용과 같이 Doctor(Emma와 Olivia)와 Nurse(Ava와 Mia)를 생성한 후 addDoctor와 addNurse를 통해 HospitalManagementSystem에 추가한다.

Patient(Bob)과 InPatient(Tom), OutPatient(Alice)를 Patient로 생성하여, addPatient를 통해 HospitalManagementSystem에 추가한다.

이름과 비밀번호를 입력 받은 후 login으로 전달하여 로그인 여부를 확인 및 로그인 된 객체를 리턴받는다.

Nurse로 로그인을 할 경우, managePatient를 통해 Nurse 전용 메인화면이 출력된다.

0을 입력하면 프로그램이 종료된다.

1을 입력하면 새로운 Patient를 추가할 수 있다. 환자의 ID를 입력하면 해당 ID가 이미 존재하는지 확인을 한다. 존재하지 않을 경우 이름과 나이 등 관련 정보들을 계속 입력할 수 있다. 나이 또는 의사 이름 까지만 입력을 할 경우 Patient로 저장이 되며, 날짜를 1개까지만 입력을 할 경우 OutPatient로 저장이 된다. 날짜를 2개까지 입력을 할 경우에는 InPatient로 저장이 된다.

2를 입력하면 기존 환자의 정보를 업데이트 할 수 있다. 우선 업데이트를 할 환자의 ID를 입력 후 해당 ID의 환자가 있을 경우 환자의 타입이 출력되며 의사 지정이 가능하다. 그리고 환자의 타입에 따라 입원일 퇴원일을 입력하거나 진료일만을 입력 후 업데이트를 할 수 있다.

3을 입력하면 환자의 정보들을 출력할 수 있다. 환자의 ID를 입력할 경우 해당 아이디의 환자만의 정보를 확인할 수 있으며, 0을 입력할 경우 모든 환자의 정보를 한 번에 확인할 수 있다.

Doctor로 로그인을 할 경우, managePatient를 통해 Doctor 전용 메인화면이 출력된다.

0을 입력하면 Nurse에서와 같이 프로그램이 종료된다.

1을 입력하면 환자의 처방을 업데이트 할 수 있다. 우선 환자의 ID를 입력 후 해당 환자의 의사가 로그인한 의사라면 처방을 입력할 수 있다.

2를 입력하면 Nurse에서와 같이 환자의 정보를 출력할 수 있으며 환자의 ID를 입력하여 ID별로 확인하거나, 0을 입력하여 모든 환자의 정보를 한번에 확인할 수 있다.