سوال دوم

سیستم رخدادنگار یا Logging System

همه ما با نحوه چاپ خروجی در کنسول آشنا هستیم و از این قابلیت برای ایجاد تعامل بین برنامه و کاربر در محیط کنسول استفاده می‌کنیم. همچنین برای آنکه از صحت اجرای برنامه خود آگاه شویم با چاپ مقداری مشخصی در لحظات و رخدادهای مشخص شده به نحوه عملکرد برنامه خود پی می‌بریم.

در دنیای برنامه‌نویسی ثبت رخداد یا به بیان دیگر ثبت لاگ[1] امری مرسوم است. این کار به دلایل مختلفی انجام می‌شود. مانند مشاهده نحوه عملکرد سیستم، گزارش‌گیری از کارهایی که سیستم در یک بازه زمانی مشخص انجام داده است و ... .

در این سوال از شما می‌خواهیم که برنامه‌ای را بنویسید که با استفاده از آن بتوان لاگ ثبت کرد (مانند تابع printLn که به وسیله آن متنی را چاپ می‌کنیم). به این منظور به مثال زیر توجه کنید.

فرض کنید برنامه‌ای داریم که به کمک آن می‌توانیم هویت افراد ثبت شده در سامانه را بررسی کنیم. متون زیر لاگ‌هایی هستند که این سامانه در کنسول ثبت کرده است.

```
2021-10-4 21:23:29:400 [INFO] org.ta.ce.AuthenticationService
    Authentication request is: name: Parham Ahmady, Password:123456
2021-10-4 21:23:29:432 [INFO] org.ta.ce.AuthenticationService
    Authentication started
2021-10-4 21:23:29:807 [INFO] org.ta.ce.AuthenticationService
    Authentication failed, user Parham Ahmady is not registered
```

همانطور که مشاهده می‌کنید، سه رخداد توسط برنامه احراز هویت به کمک برنامه رخدادنگار ثبت (چاپ) شده است.

هر لاگ از چهار بخش تشکیل شده است:

۱. تاریخ و ساعت
۲. سطح لاگ
۳. کلاسی که رخداد در آن اتفاق افتاده است
۴. پیام لاگ

[1] Log

بنابراین در اولین لاگ ثبت شده در مثال بالا یکی از توابعی که در کلاس AuthenticationService قرار داشته‌است فراخوانی شده و پیام Authentication request is: name: Parham Ahmady, Password:123456 را در سطح INFO ثبت کرده است.

سطوح لاگ به سه بخش INFO، WARN و ERROR تقسیم می‌شوند که نشان دهنده وضعیت سامانه در آن لحظه هستند. دقت کنید که در لاگ سوم مشکلی برای سامانه به وجود نیامده است و سامانه پس از دریافت ورودی نادرست نیز به کار خود ادامه می‌دهد. به همین دلیل سطح لاگ INFO است.

شما باید کلاسی به نام Logger را پیاده سازی کنید که دارای ویژگی‌های زیر باشد:

۱. تنها راه ساخت آبجکت از این کلاس باید به وسیله یکی از توابع آن صورت بگیرد که در ورودی این تابع نام کلاسی که از Logger استفاده می‌کند ورودی داده می‌شود.
۲. این کلاس باید توابعی داشته باشد که با استفاده از آن‌ها فرد استفاده کنند، بتواند در هر یک از سطوح ذکر شده لاگ ثبت کند. (ورودی اول این تابع رشته[2]، و ورودی دوم آن آرایه‌ای از رشته‌ها است) به عنوان مثال:

```
logger.info("Authentication request is: name:{}, Password:{}", {" Parham Ahmady","
123456"});
```

خروجی تابع فوق باید مانند مثال ذکر شده باشد.

برای تست برنامه خود، یک کلاس با نام LoggingTest ایجاد کنید و یک آبجکت از جنس logger ساخته و در هر یک از سطوح ذکر شده لاگ ثبت کنید.

** رعایت ساختار شیء گرایی و درج Java Doc الزامی است **

[2] String

سوال سوم

بازی حیوانات جنگل

در این سوال قصد پیاده‌سازی یک بازی ساده را در کنسول داریم.

در این بازی تعداد زیادی کارت وجود دارد که هر کارت، نماد یک حیوان است. در ابتدای بازی، به صورت رندوم به هریک از بازیکنان ۳۰ کارت داده می‌شود. در این ۳۰ کارت، هر حیوان می‌تواند صفر تا پنج بار تکرار شده باشد (نباید به هیچ بازیکنی شش کارت یا بیشتر از شش کارت، از یک حیوان داد)

سپس هر بازیکن، از بین ۳۰ کارتی که به او داده شده، ۲۰ کارت را کنار می‌گذارد و ۱۰ کارت را انتخاب می‌کند و در ادامه با این ده کارت بازی خواهد کرد.

پیش از توضیحاتِ ادامه‌ی بازی، مشخصات کارت‌ها را در جدول زیر می‌بینیم:

نام حیوان	ضربه‌ی معمولی	ضربه‌ی قوی	انرژی	جان
شیر	۱۵۰	۵۰۰	۱۰۰۰	۹۰۰
خرس	۱۳۰	۶۰۰	۹۰۰	۸۵۰
ببر	۱۲۰	۶۵۰	۸۵۰	۸۵۰
کرکس	۱۰۰	-	۶۰۰	۳۵۰
روباه	۹۰	-	۶۰۰	۴۰۰
فیل	۵۰	۷۰	۵۰۰	۱۲۰۰
گرگ	-	۷۰۰	۷۰۰	۴۵۰
گراز	۸۰	-	۵۰۰	۱۱۰۰
اسب آبی	۱۱۰	-	۳۶۰	۱۰۰۰
گاو	۹۰	۱۰۰	۴۰۰	۷۵۰
خرگوش	۸۰	-	۳۵۰	۲۰۰
لاک پشت	۲۰۰	-	۲۳۰	۳۵۰

بعد از این که هر بازیکن ۱۰ کارت خود را انتخاب کرد، در حالی که هرکس می‌تواند ۱۰ کارت حریف خود را نیز ببیند، بازی شروع می‌شود.

هر بازیکن یک نوبت حرکت می‌کند و سپس نوبت نفر بعدی می‌شود.

هر بازیکن در نوبت خود می‌تواند یک یا چند کارتِ خود را انتخاب کرده و با آن‌ها، یکی از کارت‌های حریف را مورد هدف قرار دهد و به آن حمله کند. بازیکن می‌تواند انتخاب کند که هر کدام از کارت یا کارت‌هایی که انتخاب کرده، در حمله از ضربه‌ی معمولی یا ضربه‌ی قوی خود استفاده کنند. در نهایت به مقدار مجموع قدرتِ ضربه‌ی هر یک از حیوانات حمله‌کننده، تقسیم بر تعداد حیوانات حمله کننده، از انرژی هر یک از حیوانات حمله‌کننده کم می‌شود و یا به عبارتی مقدار انرژی کم شده از هر حیوان حمله کننده یکسان خواهد بود. پس از اعمال حمله، به اندازه‌ی مجموع قدرت ضربه‌ها، از جان حیوانی که به آن حمله شده کم خواهد شد. اگر پس از اعمال حمله جان کارتی که به آن حمله شده برابر صفر یا کمتر از آن شد در واقع آن حیوان کشته شده و باید از لیست کارت‌های بازیکن حذف شود.

دقت کنید که قبل از اعمال حمله باید بررسی شود که هر کدام از حیوانات حمله کننده میزان انرژی لازم برای حمله را دارند. در غیر این صورت حمله انجام نخواهد شد و بازیکن باید مجددا حرکت جدیدی را انتخاب کند.

همچنین، هر یک از بازیکنان، سه بار در طول بازی این اجازه را دارد که به جای حمله به یکی از کارت‌های حریف، انرژی یکی از کارت‌های زنده‌ی خود را (یعنی کارتی که جانش بیشتر از صفر است و از بازی حذف نشده) ترمیم کند. با این کار، انرژیِ این کارت برابر با مقدار اولیه‌ی خود خواهد شد.

بازی به همین ترتیب ادامه پیدا می‌کند و هرگاه که یکی از بازیکنان موفق شد تمام کارت‌های حریف را از بازی حذف کند، برنده خواهد شد.

همچنین در ابتدای بازی، باید از کاربر پرسیده شود که می‌خواهد دو نفره بازی کند یا حریفش کامپیوتر باشد. می‌توانید تمام عملکردهای کامپیوتر را به صورت تصادفی پیاده‌سازی کنید. از جمله: انتخاب ۱۰ کارت از ۳۰ کارت، انتخاب کارت یا کارت‌های حمله‌کننده، انتخاب کارت هدف، انتخاب نوع ضربه و انتخاب بین تجدید انرژی یا حمله.

برای درک بهتر روش بازی، یک نمونه را با هم بررسی می‌کنیم:

ابتدا به بازیکن اول به صورت تصادفی ۵ کارت شیر، ۵ کارت خرس، ۵ کارت ببر، ۳ کارت کرکس، ۳ کارت روباه، ۳ کارت فیل، ۳ کارت گرگ و ۳ کارت خرگوش داده می‌شود. این بازیکن ۵ کارت شیر، ۳ کارت خرس، ۱ کارت روباه و ۱ کارت گرگ را انتخاب می‌کند.

به همین ترتیب، به بازیکن حریف هم ۳۰ کارت تصادفی پیشنهاد داده می‌شود که از بین آن‌ها، ۵ کارت گاو، ۱ فیل و ۴ کارت خرگوش را انتخاب می‌کند.

حال نوبت بازیکن اول است تا با کارت یا کارت‌های خود، به کارت‌های حریف حمله کند. این بازیکن، کارت شیر را انتخاب می‌کند و با آن به خرگوش حریف ضربه‌ی معمولی می‌زند. این ضربه ۱۵۰ واحد قدرت دارد، در نتیجه انرژی شیر از ۱۰۰۰ به ۸۵۰ کاهش می‌یابد و جان خرگوش مورد هدف حریف، از ۳۵۰ به ۲۰۰ کاهش می‌یابد.

حال بازیکن دوم، با ۴ کارت گاو خود، به روباه بازیکن اول حمله می‌کند و با هر یک، یک ضربه‌ی قوی به روباه حریف می‌زند. در نتیجه، انرژی این ۴ گاو، از ۴۰۰ به ۳۰۰ کاهش خواهد یافت و روباه بازیکن اول هم، از بازی حذف می‌شود.

حالا دوباره نوبت بازیکن اول است. این بازیکن ۵ کارت شیر، ۳ کارت خرس و ۱ کارت گرگ در دست دارد، که تصمیم می‌گیرد انرژی شیر خود را ترمیم کند، در نتیجه انرژی شیر او که ۸۵۰ شده بود، دوباره ۱۰۰۰ می‌شود. پس از این، بازیکن اول تنها ۲ بار دیگر این فرصت را دارد که در نوبتش، به جای حمله، انرژی یکی از حیواناتش را ترمیم کند.

پس از این که انرژی شیر کامل شد، دوباره نوبت بازیکن دوم است و او هم می‌تواند از یکی از ۳ فرصتش استفاده کند، و انرژی یکی از گاوهایش را از ۳۰۰ به ۴۰۰ افزایش دهد.

دوباره نوبت بازیکن اول است. او می‌تواند با یک کارت گرگ ضربه‌ی قوی (۷۰۰) و یک شیر ضربه‌ی قوی (۵۰۰)، به کارت فیل حریف حمله کند. از فیل ۱۲۰۰ جان کم خواهد شد، که باعث حذف این کارت می‌شود. همچنین از هر یک از کارت‌های شیر و گرگ بازیکن اول که با آن دو حمله کرده بود، ۶۰۰ واحد انرژی کم خواهد شد.

بازی به همین ترتیب ادامه خواهد داشت، تا هنگامی که یکی از بازیکنان تمام حیواناتش را از دست بدهد.

نکات پیاده‌سازی:

- برای پیاده سازی این بازی، نمی‌توانید از ارث بری استفاده کنید.
- حتما یک کلاس Player برای بازیکن‌ها تعریف کنید.
- نحوه‌ی پیاده سازی رابط کاربری بر عهده‌ی شماست، اما رابط کاربری باید ساده و قابل فهم باشد. در ضمن مشخص است که رابط کاربری باید قابلیت شروع بازی جدید، انتخابِ حریف (کامپیوتر یا انسان)، اعلام نتیجه‌ی نهایی بازی و بازگشت به منوی اصلی پس از پایان بازی را داشته باشد.
- حتما یک کلاس برای مدیریت بازی داشته باشید. پیاده‌سازی بخش‌های بازی داخل کلاس Main مجاز نیست.

** رعایت ساختار شیء گرایی و درج Java Doc الزامی است **

سوال چهارم

سیستم مدیریت بیمارستان

در این سوال قصد داریم سیستم مدیریت یک بیمارستان را طراحی کنیم.

در ابتدای برنامه فهرستی به صورت زیر برای کاربر چاپ می‌شود:

1. ADMIN – LOGIN
2. PATIENT – LOGIN
3. DOCTOR – LOGIN
4. PATIENT – SIGNUP
5. EXIT

پنل Admin:

وقتی گزینه Admin Login انتخاب می‌شود، ابتدا رمز عبور مدیر از کاربر خواسته می‌شود که این رمز ۱۲۳ بوده و در صورت غلط بودن رمز وارد شده توسط کاربر، پیامی مناسب چاپ شده و در صورت درست بودن رمز، فهرستی به شکل زیر نمایش داده می‌شود:

1. DoctorsList
2. PatientsList
3. AddDoctor
4. LogOut

در قسمت DoctorsList لیستی از پزشکان به همراه نام، نام خانوادگی، تخصص و دستمزد آن‌ها نمایش داده می‌شود.

در قسمت PatientsList لیستی از بیماران به همراه نام، نام خانوادگی، آدرس و سن بیمار چاپ می‌شود.

در قسمت AddDoctor مشخصات دکتر از قبیل نام، نام خانوادگی، سن، تخصص و دستمزد از کاربر گرفته‌شده و در ArrayList تعریف شده در کلاس بیمارستان اضافه می‌شود.

با گزینه LogOut نیز به منو اصلی برنامه برمیگردیم.

پنل Patient:

وقتی گزینه Patient Login انتخاب می‌شود، فهرستی از بیماران نمایش داده شده و پس از انتخاب بیمار مورد نظر، فهرستی به صورت زیر نمایش داده می‌شود:

1. ViewProfile
2. BookAppointments
3. ViewReport
4. LogOut

در قسمت View Profile تمامی مشخصات بیمار نمایش داده می‌شود.

در قسمت Book Appointment ابتدا کاربر تخصص پزشک را مشخص می‌کند و سپس لیست متخصص‌های مورد نظر به همراه دستمزد آن‌ها چاپ شده و کاربر با توجه به دستمزد دریافتی، پزشک دلخواه خود را انتخاب می‌کند و در ادامه برنامه تاریخی را برای نوبت تعیین می‌کند و در آخر مشخصات نوبت شامل نام و نام خانوادگی پزشک و بیمار به همراه تخصص پزشک و تاریخ چاپ می‌شود.

در قسمت View Report لیستی از گزارش‌های دریافتی بیمار نمایش داده می‌شود و با انتخاب گزارش مورد نظر توسط کاربر، اطلاعات گزارش شامل نام و نام خانوادگی پزشک به همراه داروهای تجویز شده و ملاحظات چاپ می‌شود.

با گزینه LogOut نیز به منو اصلی برنامه برمیگردیم.

پنل **Doctor**:

وقتی گزینه Doctor Login انتخاب می‌شود، فهرستی از پزشکان نمایش داده شده و پس از انتخاب پزشک مورد نظر فهرستی به صورت زیر نمایش داده می‌شود:

1. ViewProfile
2. ViewAppointments
3. LogOut

در قسمت View Profile، تمامی مشخصات پزشک نمایش داده می‌شود.

در قسمت View Appointments، تمامی نوبت‌های معین‌شده نمایش داده می‌شود و با انتخاب بیمار مورد نظر، گزارش مربوطه را می‌نویسد و به گزارش‌های بیمار اضافه می‌کند.

با گزینه LogOut نیز به منو اصلی برنامه برمیگردیم.

پنل ثبت‌نام بیمار:

در این قسمت تمامی مشخصات دریافت می‌شود و در لیست بیمارهای بیمارستان ثبت می‌شود.

کلاس Main:

در این کلاس برنامه اجرا و مدیریت می‌شود.

کلاس Hospital:

فیلدهای این کلاس شامل:

- ArrayList<Doctor> doctors
- ArrayList<Patient> patients

کلاس Admin:

متدهای این کلاس شامل:

- addDoctor
- viewDoctors
- viewPatients
- viewAppointment

کلاس Patient:

فیلدهای این کلاس شامل:

- firstName
- lastName
- age
- address
- reports

کلاس Doctor:

فیلدهای این کلاس شامل:

- firstName
- lastName
- age
- doctorType
- entryCharge
- appointments

فیلد doctorType شامل انواع زیر می‌شود:

1. Eyes Specialist
2. Ear Specialist
3. Heart Specialist
4. Bones Specialist
5. Lungs Specialist

کلاس Report:

فیلدهای این کلاس شامل:

- doctorFirstName
- doctorLastName
- patientFirstName
- patientLastName
- medicinePrescribed
- doctorsComment

این کلاس شامل متد showReport نیز است که اطلاعات نسخه را چاپ می‌کند.

کلاس Appointment:

این کلاس شامل فیلدهای زیر است:

- doctorFirstName
- doctorLastName
- patientFirstName
- patientLastName
- doctorType
- date

فیلد date باید از کلاس LocalDateTime، و تاریخ آن به فرمت yyyy-MM-dd HH:mm و تاریخ فعلی باشد.

** استفاده از متدهای دلخواه و اضافی مجاز است **

** رعایت ساختار شیء گرایی و درج Java Doc الزامی است **