به نام خدا

تمرین پنجم درس برنامه‌نویسی پیشرفته

نیم‌سال اول ۱۳۹۹-۱۴۰۰

۰. فایل مربوط به توضیحات نحوه ارسال تمرین‌ها را که در مودل قرار دارد، مطالعه کنید.

۱. تمامی فایل‌های کد را به همراه فایل متنی که در قالب pdf است (مورد سوم را بخوانید) به صورت یک فایل آرشیو zip (zip != rar) که به قالب زیر نام‌گذاری شده است، بارگذاری نمایید.

StudentNumber_FirstName_LastName.zip

9031066_Ehsan_Edalat.zip

۲. در سوال‌هایی که ورودی و خروجی مطلوب آن‌ها مشخص شده است، برنامه‌ی شما به صورت ماشینی تصحیح می‌شود. بنابراین رعایت نحوه ورودی‌گرفتن و نمایش خروجی اهمیت بسیاری دارد. دقیقا همان‌طور که از شما خواسته شده است ورودی‌ها را خوانده و خروجی‌ها را تولید کنید.

۳. پاسخ سوالات تشریحی را به صورت تایپ‌شده و در قالب یک فایل pdf (برای کل تمرین) تحویل دهید.

۴. در صورت مشاهده هرگونه تقلبی، طبق موارد گفته شده در قوانین درس برخورد خواهد شد.

۵. در صورت وجود هرگونه ابهام می‌توانید از طریق ربات تلگرامی AP_Admin_bot با تدریس‌یاران در ارتباط باشید.

مهلت تحویل: تا یکشنبه ۱۱ آبان ۱۳۹۹ ساعت ۲۳:۵۵ شب

فهرست سوالات

صفحه

سوال اول

جاهای خالی را با کلمات مناسب پر کنید.

1) A method that is declared _____ cannot be overridden in a subclass.
2) Can hold objects of more than one type: it is the definition of _____.
3) Objects of _____ super classes cannot be instantiated.
4) All classes in Java inherit directly or indirectly from the _____ class.
5) methods declared in an interface are by default _____.
6) At interfaces all fields are public, _____ and _____.
7) In Java we can only use multiple inheritance for _____.

سوال دوم

درستی یا نادرستی عبارات زیر را مشخص کنید:

1) When you instantiate a sub class, super class constructor will be also executed.
2) An abstract class has no use until unless it is extended by some other class.
3) A constructor of abstract class is called when an instance of an inherited class is created.
4) Abstract classes may contain non-final variables, whereas variables in interface are final, public and static.
5) Protected members are accessible within a package and inherited classes outside the package.

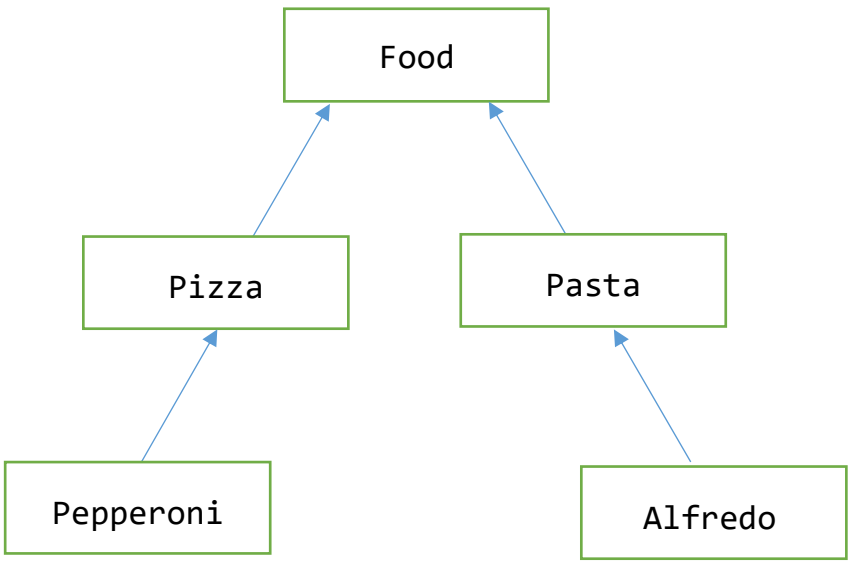
سوال سوم

مفاهیم زیر را تعریف کنید.

1) SubType and SuperType.
2) Diamond Problem.

سوال چهارم

با توجه به دیاگرام زیر به سوالات زیر پاسخ دهید.



الف) مجاز بودن یا نبودن هر یک از دستورات زیر را با ذکر دلیل مشخص کنید.

- `Food f1= new Pasta();`
- `Food f2=new Alfredo();`
- `Pizza pi1=new Food();`
- `Pasta pa1=new Alfredo();`
- `Pizza pi2=new Pepperoni();`

ب) با فرض دستورات زیر:

- `Food f1=new Food();`
- `Food f2=new Food();`
- `Pepperoni pe1=new Pepperoni();`
- `Alfredo a1=new Alfredo();`
- `Pizza pi1=new Pizza();`

مجاز بودن یا نبودن هر یک از دستورات زیر را با ذکر دلیل بیان کنید.

- `f1=pe1;`
- `pi1=pe1;`
- `a1=f1;`
- `f2=p1;`
- `f1=pe1;`

سوال پنجم

(پیاده‌سازی)

در این سوال می‌خواهیم یک سیستم بیمارستانی را پیاده‌سازی کنیم.

بخش‌های موجود در بیمارستان شامل ICU، CCU و بخش معمولی[1] است. (این بخش‌ها را چگونه در نظر بگیریم؟ enum؟ کلاس‌های متفاوت؟ یک رشته؟)

بیمه‌های مورد قبول:

```
SocialSecurity,
ArmedForcesServices,
GovernmentEmployeesServices
```

[1] Ordinary

این برنامه برای مدیریت بیمارستان خاصی نیست، پس باید بتوان اسم، بخش‌ها و سایر ویژگی‌های بیمارستان را در زمان اجرا تعیین کرد (بهتر نیست یک کلاس بیمارستان در نظر بگیرید؟)

هر بیمارستان، شامل پزشکان، کارکنان (برای بخش امتیازی) و بیماران است. طبیعتاً شما باید بتوانید به هر کدام از این گروه‌ها عضو اضافه و کم کنید.

شما باید بیماران در یک بخش را نمایش دهید. برای این منظور تابعی داشته باشید که با گرفتن نام بخش، نام بیماران را نشان دهد (می‌توانید برای هر بخش لیست بیماران داشته باشید، می‌توانید در کلاس بیمار، بخش را نگه دارید و... . ولی برای معماری خود دلیل بیاورید).

هر پزشک دارای نام، نام‌خانوادگی، شماره نظام پزشکی (این فیلد یکتاست و برای شناسایی پرشک کافیست پس برای این کلاس تابع equals را بازنویسی کنید و این فیلد را چک کنید.) و تخصص (می‌توانید با دلیل از enum یا رشته استفاده کنید) است.

باید بتوان بیماران تحت درمان یک پزشک را فهمید. برای اینکار تابعی بنویسید (این افراد را درچه کلاسی و با چه ساختاری ذخیره می‌کنید؟ چرا؟).

هر بیمار دارای نام و نام‌خانوادگی است. باید پزشک معالج بیمار را نیز بدانیم (در این کلاس فیلد دکتر خواهیم داشت؟). هر بیمار می‌تواند بیمه شده باشد و یا نشده باشد. نوع بیمه بیمار برای محاسبه هزینه مهم نیست ولی اکر بیمه باشد هزینه‌ها نصف می‌شود (حب اینکه نوع بیمه فعلاً مهم نیست یعنی کلا نباید نام بیمه را ذخیره کرد؟).

کلاس پرسنل دارای نام و فیلد کاری و حقوق می‌باشد.

اضافه کردن تابع هایی که فکر میکنید نیاز است ولی ذکر نشده است مانعی ندارد. شما باید تمامی مواردی که طراحی کردید را با ذکر دلیل بیان کنید. نوشتن جاواداک و کامنت الزامی است. طرز تفکر شما و رعایت شی‌گرایی مهم است. طبیعتاً نوشتن همه‌ی کد در یک کلاس نمره‌ای نخواهد داشت.

سوال ششم

در این تمرین، وظیفه شما ایجاد بازی شطرنج است. این بازی که از اواخر قرن پانزدهم به شکل فعلی خود شروع شد، در حال حاضر یکی از محبوب‌ترین و همه‌گیرترین بازی‌های جهان است. اما ما از شما نسخه‌ی ساده‌شده بازی را می‌خواهیم.

این کار شما باید کلاس‌ها را طوری طراحی کنید که هر بازیکن ۱ شاه، ۵ سرباز، ۲ اسب و ۲ قلعه داشته باشد، تمامی مهره‌ها باید از کلاس `Element` ارث‌بری کنند (این کلاس شامل موارد مشترک بین مهره‌ها نظیر رنگ، مختصات و... می‌باشد). همچنین تمامی مهره‌های بازی باید از اینترفیس `Actions` که توضیح آن در زیر آمده پیروی کنند.

```
/**
 * Actions
 */
public interface Actions {
    void move ();
}
```

این تابع را هر مهره بر اساس قوانینی که در ادامه خواهیم گفت، بازنویسی می‌کند.

در این بازی زمین بازی به صورت خانه‌های یک‌درمیان سیاه و سفید و به صورت مستطیل ۵ * ۱۰ می‌باشد که مهره‌ها در ابتدای بازی در عرض زمین بطوری که هر ۵ خانه پر شوند قرار می‌گیرند (به پیاده‌سازی کلاس Board فکر کنید).

مهره‌های هربازیکن دارای رنگ یکسان سیاه یا سفید است و همیشه بازیکنی که به رنگ سفید است بازی را آغاز می‌کند.

نحوه حرکت :

- **سرباز:** صرفاً رو به جلو حرکت می‌کند و در هر مرحله فقط می‌تواند یک خانه به جلو برود! در صورت پر بودن خانه روبه‌رو توسط مهره حریف آن مهره را از بازی حذف می‌کند!! (توجه کنید در

این بازی سرباز به صورت مورب مهره حریف را حذف نمی‌کند و قوانین برای راحتی کار ایجاد شده‌اند).

- **اسب:** به صورت L و در جهت حرف حرکت می‌کند . به عنوان مثال اگر اسبی در خانه (1,0) باشد می‌تواند به خانه های (2,2) یا (2,1) برود که در این مختصات گوشه چپ و پایین صفحه مبدا می‌باشد. همچنین در صورتی که مهره حریف در آن مکان قرار گرفته باشد آن را از بازی حذف میکند!
- **قلعه:** حرکتش مشابه سرباز است با این تفاوت که اگر راهش توسط مهره خودی سد نشده باشد می‌تواند بدون محدودیت به جلو و عقب حرکت کند مثلا ۳ خانه به جلو یا ۲ خانه به عقب برود. در صورت برخورد با مهره حریف آن را از زمین بازی حذف و در همان خانه متوقف می‌شود.
- **شاه:** می‌تواند در جهات راست، چپ، بالا و پایین، تنها یک خانه جابه‌جا شود و در صورت برخورد با مهره حریف، آن را از بازی حذف می‌کند.

* توجه کنید که اگر مسیر حرکت مهره‌ای به وسیله مهره خودی یا حریف سد شده باشد، نمی‌تواند حرکت کند (مهره حریف اگر در مقصد باشد می‌توانید حذفش کنید). و برای اسب وجود داشتن مهره در مسیر مهم نیست.

بازی زمانی تمام می‌شود که تمام مهره‌های یکی از بازیکنان از بازی بیرون شوند.

نیازی به نمایش بازی به صورت گرافیکی در کنسول نیست (البته که نمایش کاربرپسند، نمره بیشتری خواهد داشت) و می‌توانید آن را به صورت زیر نمایش دهید:

```
(0,9):Castle(Black)   (1,9):Horse(Black)  (2,9):King(Black)  (3,9):Horse(Black)  (4,9):Castle(Black)
(0,8):Soldier(Black)  (1,8):Soldier(Black)  (2,8):Soldier(Black) (3,8):Soldier(Black)  (4,8):Soldier(Black)
(0,7):EmptyBlock      (1,7):EmptyBlock    (2,7):EmptyBlock  (3,7):EmptyBlock    (4,7):EmptyBlock
(0,6):EmptyBlock      (1,6):EmptyBlock    (2,6):EmptyBlock  (3,6):EmptyBlock    (4,6):EmptyBlock

                              ..........................................

                              ..........................................

(0,1):Soldier(White)  (1,1):Soldier(White)  (2,1):Soldier(White) (3,1):Soldier(White)  (4,1):Soldier(White)
(0,0):Castle(White)   (1,0):Horse(White)  (2,0):King(White)  (3,0):Horse(White)  (4,0):Castle(White)
```

کلاس‌های گفته شده کامل نیستند، شما باید بتوانید ساختار مناسبی را برای پروژه ارائه دهید و نمودار UML آن را نیز رسم کنید (برای کلاس‌بندی خود دلیل داشته باشید، قطعاً همه‌ی کد در یک کلاس، معماری خوبی نیست. اما معماری هر شخصی لزوماً با شخص دیگری یکی نمی‌شود).

رعایت ساختار شی‌گرایی، ارث‌بری و نوشتن جاواداک و کامنت الزامی است و در صورت عدم رعایت، نمره‌ای تعلق نخواهد گرفت.