برنامه نویسی پیشرفته

زمستان و بهار ۹۹-۱۳۹۸ - دانشکده علوم ریاضی

دانشگاه صنعتی شریف

با توجه به شرایط خاص پیش آمده تیم درس برنامه نویسی پیشرفته تصمیم گرفتند که یک پرسش‌نامه بدون تاثیر در ارزیابی برای بررسی میزان پیشرفت مطالعه برگزار کند. هدف از این پرسش‌نامه بررسی پیشرفت عملکرد و مطالعه شما است.
هدف از این پرسش‌نامه این موارد است:

- جبران فاصله ایجاد شده میاد دانشجوها با همدیگر که امکان ارائه بازخورد پیشرفت مطالعه به یکدیگر را ایجاد می‌کند.
- جبران فاصله ایجاد شده میان دانشجوها و تیم درس برای دریافت بازخورد پیشرفت تحصیلی
- بازخورد هر دانشجو به خود در رابطه با پیشرفت مناسب در مطالعه و یادگیری مفاهیم از طریق منابع درس
- جهت‌دهی به اشکالاتی که شما ممکن هست هنوز در جریان وجود نقطه ضعف خود در این رابطه نباشید و رفع آنها در جلسه‌های رفع اشکال آنلاین
- دقت کنید که این پرسش‌نامه تنها مرجع برای بازخورد پیشرفت مناسب شما در مطالعه منابع نیست. تمرین‌ها و پروژه نیز سهم بزرگی در این مساله دارند. پس حتما پس از پاسخ به سوال‌ها و ارسال پاسخ‌ها، اشکال‌ها و ابهام‌هایی که داشتید در جلسه‌های آنلاین رفع اشکال در میان بگذارید و رفع کنید و از این فرصت استفاده کنید.

توضیحات

- نتیجه این پرسش‌نامه تاثیری در ارزیابی نهایی این درس ندارد.
- این پرسش‌نامه برای اطمینان بیشتر از اینکه مسیر درس را درست طی می‌کنید طراحی شده.
- اگر نیاز به بررسی صحت پیشرفتتان دارید حتما در این پرسش‌نامه شرکت کنید.
- در صورتی که با مطالب درس به درستی پیش آمده باشید می‌توانید به تمام سوال‌ها پاسخ دهید.
- در صورتی که به بخشی از هر سوال تسلط ندارید یا احتیاج به بررسی صحت پاسخ‌ها دارید حتما در جلسه‌های رفع اشکال شرکت کنید و اشکال یا ابهام‌های خود را رفع کنید.
- سعی کنید جواب‌ها کوتاه و دقیق باشند که مرور جواب در جلسه رفع اشکال سریع‌تر انجام شود.
- از آنجایی که این پرسشنامه برای یادگیری طراحی شده می‌توانید در پر کردن سوال‌ها با هر فردی مشورت و همفکری کنید.

نحوه انجام پرسش‌نامه

- برای پاسخ به این پرسش‌نامه یک نسخه از این فایل را از منو فایل و گزینه گرفتن یک کپی برای خود ایجاد کنید و جواب‌های آن را در همین فایل بنویسید.
- پس از جواب دادن به سوال‌ها آن را در قالب PDF دانلود کنید.
- فایل PDF در یک ریپازیتوری github بارگذاری کنید.
- آدرس این ریپازیتوری را در یک فایل یک خطی با پسوند جاوا داخل کوئرا و در بخش پرسشنامه بررسی پیشرفت بارگذاری کنید.

سوال‌ها

سوال ۱

خروجی این برنامه را بدست بیاورید و به ازای هر خط توضیح دهید که چرا به این خروجی رسید؟

```
class Classes {
   static class A {
       static int intValue = 0;
       int integerValue = 20;

       A() {
           integerValue = 5;
           printValue();
           print();
       }

       void printCaller() {
           print();
       }

       void printValue() {
           System.out.println("B:" + integerValue);
       }

       void print() {
           System.out.println("A:" + intValue);
       }
   }

   static class B extends A {
       B(int v) {
           intValue = v;
           integerValue = 15;
           printValue();
           print();
       }

       void print() {
           System.out.println("B:" + intValue);
       }

       void printSuper() {
           super.print();
       }

       void printCaller() {
           printValue();
           super.printValue();
       }

       void printValue() {
           System.out.println("B:" + integerValue);
```

```
            super.printValue();
        }
    }

    static public class C extends A {
        void printCaller() {
            System.out.println("B:" + integerValue);
        }

        void print() {
            System.out.println("A:" + intValue);
            super.printCaller();
        }
    }
}

class Problem1 {
    public static void incrementValue(Classes.A object) {
        object.intValue++;
        object.integerValue++;
    }

    public static void incrementValue(int firstValue, int secondValue) {
        firstValue++;
        secondValue++;
    }
    public static void main(String[] args) {
```

اجرا کردن constructor داخل جاوا با مراحل ذکر شده در
https://docs.oracle.com/javase/specs/jls/se8/html/jls-12.html#jls-12.5
انجام می‌شود که به اختصار به توضیح آن می‌پردازم:
ابتدا پارامتر های ورودی constructorرا به عنوان متغیر های جدیدی برای constructor تعریف می‌کند.
سپس چک می‌کند که آیا این constructor داخل خود constructor دیگری از همان کلاس‌را صدازده است یا خیر، اگر صدا زده بود به اجرای constructorدوم میپردازد و به همین ترتیب روال تکرار می‌شود
اگر constructorشرط 2را نداشت، بررسی می‌کند که آیا این کلاس از کلاس دیگری به جز object ارث برده است یا خیر، اگر ارث برده بود، با اجرای constructor کلاس پدر آن می‌پردازد
سپس متغیر هایی که در سطح کلاس تعریف شده اندرا برای constructor تعریف می‌کند
در مرحله آخر خط به خط دستور های داخل constructorرا اجرا می‌کند.

```
        Classes.A a = new Classes.A();
```

در اینجا آبجکتی از کلاس A ساخته شده پس مراحل 1 و 4 و 5 اجرا می‌شود
در اجرای این مراحل چون تعریف integerValue در مرحله 5 نسبت به تعریف آن در مرحله 4 جلوتر انجام شده، این مقدار در printValue چاپ می‌شود.

```
        Classes.B b = new Classes.B(10);
```

در این قسمت آبجکتی از کلاس B ساخته می‌شود که خود از کلاس A ارث برده است؛ پس مراحل اجرا 1 3 4 5 است به این صورت که ابتدا constructor کلاس پدر آن(A) اجرا شده و بعد constructor کلاس B، اما نکته ای که وجود دارد این است که دو تابع print و printValue چون داخل B، override شده‌اندزمانی که constructor کلاس A این توابع‌را فراخوانی می‌کنند، توابع در کلاس B اجرا می‌شوند(این قسمت برای من روشن نیست خیلی که چرا این اتفاق می‌افتد)

```
        Classes.A c = b;
```

متغیر c در اینجا در واقع تبدیل شده متغیر b از نوع آبجکت کلاس B به کلاس A است، اما چون سازنده آن کماکان کلاس B است، تفاوت متغیر c نسبت به b در سطح دسترسی آن ها است به این صورت که متغیر c به متد های کلاس B دسترسی ندارد اما متغیر b به کلاس های A و B دسترسی دارد. این نکته نیز قابل ذکر است که متغیر c متد هایی که در کلاس B، override شده-اندرا می‌تواند مطابق کلاس B اجرا کند.

```
        b.print();
```
اجرای تابع print کلاس B

```
        c.print();
```
اجرای تابع print کلاس B چرا که این متد از override،A شده است

```
        ((Classes.A) b).print();
```
این دقیقا خط بالایی است.

```
        b.printSuper();
```
مقدار A=10 چاپ می‌شود چرا که مقدار intValue در constructor کلاس B تغییر کرده است

```
        a.printCaller();
```
متد print کلاس A صدازده می‌شود اما با توجه به static بودن متغیر intValue و تغییر آن در سازنده کلاس B که قبلا صدازده شده است، مقدار 10 چاپ می‌شود

```
        b.printCaller();
```
متد printValue کلاس های A و B صدازده می‌شود با مقدار integerValue = 15

```
        c.printCaller();
```
دقیقا مثل بالایی

```
        incrementValue(a);
```
مقادیر intValue و integerValue آبجکت کلاس A زیاد می‌شود

```
        a.printCaller();
```
مقدار intValue چاپ می‌شود که چون static هست، برای همه آبجکت ها عوض می‌شود.

```
        incrementValue(b);
```
مقادیر intValue و integerValue آبجکت کلاس B زیاد می‌شود توجه شود که به این دلیل که آبجکت c نیز به این آبجکت ها دسترسی دارد، برای آن هم عوض می‌شود

```
        b.printCaller();

        incrementValue(c);
```
مقادیر intValue و integerValue آبجکت کلاس B زیاد می‌شود

```
        c.printCaller();


        incrementValue(b.intValue, b.integerValue);
```
در این قسمت تغییری‌رخ نمی‌دهد چرا که این مقادیر در این کلاس به صورت local تغییر می‌کند و تاثیر آن اعمال نمی‌شود

```
        b.printCaller();
        c.printCaller();
    }
}
```

سوال ۲

توضیح دهید که هدف از ارث بری در شی گرایی چیست. چه زمان از composition و چه زمان از inheritance استفاده می‌کنیم؟ چگونه می‌توانیم از سازنده پدر را فراخوانی کنیم؟ چگونه می‌توانیم سازنده دیگری از خود کلاس را فراخوانی کنیم؟

هدف نهایی این است که کد کمتری نوشته شود، چرا که برای مثال توابع و متغیر هایی در کلاس های مرتبط با هم تکرار می‌شوند، با ارث بری تنها یک بار این توابع نوشته می‌شوند.
زمانی که کلاس ما با کلاس دیگری رابطه is-a داشته باشد از inheritance و زمانی که رابطه has-a داشته باشد از composition استفاده می‌کنیم. برای مثای خانه یک ساختمان است(is-a) که آشپزخانه دارد(has-a).Composition در واقع یک design technique است اما inheritance یک مکانیزم است که در خود جاوا پیاده سازی شده.

خط اول سازنده، از this.anotherConstructor(parameters) استفاده می‌کنیم.

سوال ۳

توضیح دهید که چرا از رابط‌ها (interface) استفاده می‌کنیم. چه محدودیت‌هایی نسبت به یک کلاس دارند و چرا امکان پیاده‌سازی متد در آنها داده شده است؟

Interface ها در واقع پنجره هایی هستند که می‌توانیم از این پنجره ها به کلاسی که آن هارا implement می‌کند نگاه کنیم. به این معنا که اگر برای مثال interface مقایسه را در نظر بگیریم، کلاس هایی که این interfaceرا implement می کنندرا می‌توانیم با هم مقایسه کنیم، یعنی باید به نحوی پیاده سازی شوند که قابل مقایسه باشند. برای‌رسیدن به این هدف، هر interface باید برنامه نویس‌را مجبور به پیاده سازی توابعی کند که این قابلیت‌را فراهم سازند، قالب این توابع در خود interface باید تعریف شود.

محدودیت interface ها نسبت به کلاس این است که توابع و فیلد های آن نمیتوانند body داشته باشند، به این معنا که‌صرفا باید در آن قالب تعریف کرد و کدی نباید در خود آن اجرا شود.

سوال ۴

کلاس انتزاعی (abstract) چیست و چه زمانی در مدل‌سازی از یک کلاس انتزاعی استفاده می‌کنیم؟ این نوع کلاس چه تفاوتی با رابط(interface) دارد؟

کلاس های abstract کلاس هایی هستند که به تنهایی قابل اجرا نیستند، به این صورت که توابعی در آن هستند که پیاده سازی آن بر عهده کلاسی است که آن‌را به ارث می‌برد یا اینکه آبجکتی می‌خواهد از آن بسازد. برای مثال اگر کلاس Animal می‌تواند تابعی مانند breathe داشته باشد چون در همه حیوانات به یک صورت انجام می‌شود(تقریبا!) اما تابع eat آن برای هر حیوانی متفاوت است، پس این تابع abstract خواهد بود که هر حیوانی که این کلاس‌را ارث می‌برد تعریف منحصر خودرا داشته باشد.

Interfaceصرفا یک طرز نگاه و یک قابلیت است و مانند کلاس نمی‌شود از آن آبجکت ساخت و یا کدی در آن نوشت، اما کلاس abstract قابلیت های یک کلاس‌را به طور کامل دارد به صورتی که تنها یک‌سری از توابع آن پیاده سازی نشده‌اند.

سوال ۵

override کردن تابع و متغیر چه تاثیری در عملکرد متد در یک کلاس فرزند می‌گذارد؟ چطور می‌توانیم پس از override شدن یک متد در کلاس فرزند در هر کدام از مکان‌های زیر به نسخه هم نام آن متد در کلاس پدر دسترسی پیدا کنیم؟

- متدی داخل کلاس پدر
- متدی داخل کلاس فرزند

- خارج از دو کلاس

با override کردن تابع یا متغیر در کلاس فرزند می‌توانیم تابع موجود در کلاس پدر را به طرز دیگری پیاده سازی کنیم. به طوری که مثلا تابع در کلاس پدر را می‌توانیم Default و تابع override شده‌را customized در نظر بگیریم.

متدی داخل پدر:
بستگی به این دارد که آبجکتی که ساخته شده از نوع کدام کلاس باشد، اگر آبجکت از نوع کلاس پدر باشد که چون اصلا کلاس پسر را نمی‌شناسد هر فراخوانی متدی در همان کلاس پدر خواهد بود اما اگر آبجکت کلاس فرزند باشد:

راهی برای دسترسی به تابع override شده کلاس پدر وجود ندارد!
https://stackoverflow.com/questions/15668032/how-to-call-the-overridden-method-of-a-superclass

متدی داخل فرزند:
با استفاده از super.method() می‌توانیم به کلاس پدر دسترسی پیدا کنیم.

خارج از دو کلاس:
باید آبجکتی از نوع کلاس پدر با constructor پدر را بسازیم تا بتوانیم به توابع آن دسترسی داشته باشیم.

سوال ۶

توضیح دهید که منظور از چندریختی در شی گرایی چیست و چه مزیتی ایجاد می‌کند.

منظور از چندریختی در شی‌گرایی این است که یک آبجکت می‌تواند از نوع کلاس های پدر خود باشد. مزیت این موضوع این است که تصور کنید کلاسی وجود داشته باشد که آبجکتی از نوع Car بگیرد و بخواهد تابع accelerate آن‌را صدا بزند، با استفاده از چندریختی ما می‌توانیم آبجکت کلاس برای مثال BMW را تبدیل به نوع Car کنیم و آن‌را به این کلاس پاس بدهیم. اگر تابع accelerate کلاس Car، override شده باشد، در زمان اجرای این تابع، accelerate عه BMW صدا زده می‌شود. اگر چندریختی نبود، باید تابع accelerate را برای تک تک کلاس هایی که از Car ارث برده بودند را پیاده می‌کردیم.

https://www.quora.com/What-are-the-benefits-of-polymorphism-in-object-oriented-programming

سوال ۷

چرا از توابع و متدها در زبان برنامه نویسی استفاده می‌کنیم؟ در طراحی برنامه و شکستن آن به توابع و متدهای مختلف چه نکته‌هایی را باید رعایت کرد که خوانایی آن بیشتر شود و پیچیدگی اضافی نداشته باشیم؟

دلیل اصلی استفاده از توابع و متد های کوتاه تر شدن کد است. به این صورت که قسمتی از کد را که زیاد تکرار می‌شود را داخل تابعی قرار می‌دهیم تا قابل استفاده مجدد باشد و هر بار که نیاز به استفاده از آن داشتیم دوباره آن‌را کامل ننویسیم.
نکته هایی که می‌توان رعایت کرد می‌تواند شامل موارد زیر باشد:

1- زمانی که قطعه کدی داریم که شاید ربط مستقیمی به اسم تابعی که در آن اجرا می‌شود نداشته باشد را داخل تابع دیگری بنویسیم. برای مثال ایجاد فایل در تابع اجرای بازی
2- نام‌گذاری توابع به صورتی باشد که عملکرد آن‌را به خوبی توضیح دهد.
3- نام‌گذاری توابع با حروف کوچک شروع و کلمه های میانی با حروف بزرگ نوشته شوند:runFast()
4- اگر نام متد نمی‌تواند توضیح مناسبی باشد، حتما کامنت گذاری شود.

سوال ۸

کلاس درونی (inner class) چه انواعی دارد و هر کدام چه کاربردی در مدل‌سازی و توصیف موجودات دارد؟ چگونه می‌توانیم یک شی از هر نوع ایجاد کنیم؟ در صورت override شدن یک متد یا متغیر توسط یک کلاس درونی چگونه می‌توان به نسخه override شده از کلاس بیرونی دسترسی پیدا کرد؟

چهار نوع کلاس داخلی وجود دارد:

1- Member
2- Static member
3- Local
4- Anonymous

member-1:
این کلاس ها مانند کلاس های عادی داخل کلاس نوشته می‌شوند و می‌توان آن ها را public, private, ... تعریف کرد. استفاده این کلاس ها برای زمانی است که مثلا تنها انسان است که توانایی آنالیز را دارد(!) پس کلاس analyze داخل کلاس انسان قرار می‌گیرد. دسترسی به این کلاس از طریق آبجکت کلاس بیرونی است.

static member-2:
این کلاس ها با کلید واژه static داخل کلاس دیگری نوشته می‌شوند و برای دسترسی به آن ها نیازی نیست که شی ای از کلاس بیرونی آن ساخته شود. استفاده همچین کلاسی برای من روشن نیست مگر اینکه دلیل استفاده از آن این باشد که کلاس درونی مرتبط با کلاس بیرونی باشد، پس تعریف کردن آن داخل این کلاس readability کد را زیاد کند.

local-3:
کلاسی است که داخل یک تابع تعریف می‌شود و فقط آن تابع می‌تواند از آن استفاده کند. این کلاس تنها به متغیر های final متد دسترسی دارد(نمیدونم چرا!)

anonymous-4:
این کلاس ها بیشتر برای eventListener ها به کار می‌روند، به این صورت که معمولا در ورودی توابع مستقیما آبجکتی از آن new کرده و توابع آن را پیاده سازی می‌کنیم.

https://www.cis.upenn.edu/~matuszek/General/JavaSyntax/inner-classes.html

سوال ۹

کلمه کلیدی final روی هر کدام از موارد زیر چه تاثیری دارد؟

- تابع و متد

تابع یا متد قابلیت override شدن را نخواهد داشت.

- تعریف کلاس

کلاس را نمی‌شود به ارث برد.

- یک متغیر از نوع شی

متغیر تنها به یک قسمت از حافظه می‌تواند اشاره کند و نمی‌توان آدرس آن را عوض کرد.

- یک متغیر از نوع پایه

مقدار آن پس از تعیین دیگر عوض نمی‌شود.

سوال ۱۰

کلمه کلیدی static روی هر کدام از موارد زیر چه تاثیری دارد؟

- تابع و متد

به این معنا که تابع یا متد مرتبط به آبجکت کلاس نیستند و مستقیما به خود کلاس ربط دارند.

- تعریف کلاس

فقط به کلاس های درونی قابل اعمال است و به این صورت که کلاس درونی دیگر مرتبط به کلاس بیرونی خود نیست و به متغیر های آن دسترسی ندارد و برای آبجکت ساختن و صدازدن آن باید از outerClass.innerClass استفاده کرد

- یک متغیر از نوع شی

این متغیر مرتبط با خود کلاس می‌شود و می‌توان با استفاده از اسم کلاس به آن دسترسی پیدا کرد. اما باید توجه به مقدار دهی آن نیز داشت.

- یک متغیر از نوع پایه

این متغیر مرتبط با خود کلاس می‌شود و می‌توان با استفاده از اسم کلاس به آن دسترسی پیدا کرد.