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

B:5
A:0

هنگامی که آبجکت جدیدی از کلاس داخلی A ساخته میشود، ابتدا inline initialization انجام میشود و سپس constructor مقداردهی میکند. در نتیجه مقدار integerValue از 20 به 5 تغییر میکند و مقدار intValue همان صفر می ماند.

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

B:5
B:5
B:0

ابتدا initialization در کلاس پدر یعنی A صورت میگیرد و مقدار integerValue از 20 به 5 تغییر میکند و مقدار intValue صفر میماند. ولی متدهای print() و printValue در کلاس B که override شده اند، اجرا میشود.

B:15
B:15
B:10

سپس با اجرای constructor کلاس B مقدار intValue به 10 و مقدار integerValue به 15 تغییر میکند و متدهای printValue و print از کلاس B فراخوانی میشوند.

```
        Classes.A c = b;

        b.print();
```

B:10

مقدار intValue از طریق فراخوانی متد print در کلاس B چاپ میشود.

```
c.print();
```

B:10

اشاره گر c به همان آبجکتی که از طریق اشاره گر b ارجاع میشود، اشاره میکند و چون در یک polymorphism متد موجود در آبجکت اجرا میشود تابع print از کلاس B فراخوانی میشود.

```
((Classes.A) b).print();
```

B:10

تابع print در کلاس B فراخوانی میشود. casting فوق چه اعمال شود چه نشود، تفاوتی ندارد.

```
b.printSuper();
```

A:10

متد print از کلاس پدر یعنی A فراخوانی میشود.

```
a.printCaller();
```

A:10

پس از فراخوانی constructor کلاس B که مقدار intValue را به 10 تغییر داد، از آنجایی که یک متغیر static است، در کلاس A نیز مقدار آن به 10 تغییر میکند.

```
b.printCaller();
```

B:15
B:15
B:15

در واقع ابتدا متد printValue در B و سپس از طریقsuper همین متد دو بار از کلاس A فراخوانی میشود.

```
c.printCaller();
```

B:15
B:15
B:15

در polymorphism متدهای آبجکت اهمیت دارد نه اشاره گر. لذا نتیجه این قسمت با کامند قبلی تفاوتی نمیکند.

```
incrementValue(a);
a.printCaller();
```

A:11

چون پارامتر متد فوق یک رفرنس است، پس متد incrementValue اولی در کلاس Problem1 اجرا میشود و در اثر اجرای تابع، مقدارهای intValue و integerValue یک واحد افزایش می یابند.

```
incrementValue(b);
b.printCaller();
```

B:16
B:16
B:16

توضیحات فوق در مورد این کامند نیز صدق میکند

```
incrementValue(c);
c.printCaller();
```

B:17
B:17
B:17

اشاره گر c به همان آبجکتی که اشاره گر b رفرنس میدهد، اشاره میکند و این باعث میشود مقدار integerValue و intValue باز هم یک واحد بیشتر شود.

```
incrementValue(b.intValue, b.integerValue);
b.printCaller();
```

B:17

```
            B:17
            B:17
از آنجایی که پارامتر متد دو متغیر int است، دومین متد incrementValue
در کلاس فراخوانی میشود و چون صرفا یک کپی از مقدارهای صحیح کلاس b به
تابع داده شده، تغییرات اعمال شده در تابع در ویژگی های کلاس b تاثیری
ندارد و صرفا متغیرهای محلی که به تابع پاس شده افزایش می یابند.
        c.printCaller();
            B:17
            B:17
            B:17
                        توضیحات فوق در این قسمت نیز صدق میکند.
    }
}
```

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 از ارث بری ایجاد یک سلسله مراتب از کلاس ها است که هر زیر کلاس، ویژگی ها و رفتارهای ابرکلاس خود را به ارث برد و نیازی به پیاده سازی یک متد تکراری در هر زیرکلاس نباشد.

زمانی از ترکیب استفاده میکنیم که یک کلاس در ساختار خود یک یا تعدادی اشیا از کلاس های دیگر را داشته باشد. اما زمانی که یک کلاس ویژگی های یک کلاس دیگر را داشته و به عبارتی نمونه های آن زیرمجموعه ای از نمونه های کلاس دیگر باشد، از ارث بری استفاده میکنیم.

با کلیدواژه super در اولین دستور از سازنده زیرکلاس میتوان سازنده ابرکلاس را فراخوانی کرد.وگرنه هنگام نمونه گیری از زیرکلاس سازنده ای بدون پارامتر از ابرکلاس به طور ضمنی فراخوانی میشود.

برحسب نوع و تعداد پارامترهای سازنده یک کلاس میتوان سازنده موردنظر خود را فراخوانی کرد.

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؟

زمانی که میخواهیم صرفا ویژگی ها و عملکردهای یک کلاس را لیست کنیم ولی نحوه انجام آنها را توصیف نکنیم چون بسته به نوع زیرکلاس متفاوت است و یا میخوهایم یک کلاس از چند کلاس ارث بری کند، از رابط استفاده میکنیم.

رابط یک کلاس کاملا انتزاعی است و تمام متدهای آن به طور ضمنی انتزاعی هستند و اصلا نمیتوان داخل ان هیچ متدی را پیاده سازی کرد. همچنین تمام متغیرها به طور ضمنی فاینال و استاتیک هستند.

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 انتزاعی حداقل یک متد انتزاعی دارد و این باعث میشود که نتوان به طور مستقیم از آن آبجکت ساخت و تنها از زیرکلاس های غیرانتزاعی آن میتوان نمونه گیری کرد.

رابط یک کلاس کاملا انتزاعی است و تمام متدهای آن به طور ضمنی انتزاعی هستند و اصلا نمیتوان داخل ان هیچ متدی را پیاده سازی کرد ولی در کلاس انتزاعی یک متد میتواند انتزاعی نبوده و پیاده سازی شود.

سوال ۵

override کردن تابع و متغیر چه تاثیری در عملکرد متد در یک کلاس فرزند می‌گذارد؟

اگر این متغیر و متد در یک نمونه از کلاس فرزند فراخوانی شود، فرم تغییریافته در کلاس فرزند اجرا خواهد شد.
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 **فراخوانی متد (که اگر استاتیک نباشد از کلیدواژه this هم میتوان استفاده کرد.)**
- متدی داخل کلاس فرزند **به کمک کلیدواژه super**
- خارج از دو کلاس **نمونه گیری از کلاس پدر و فراخوانی متد**

سوال ۶

توضیح دهید که منظور از چندریختی در شی گرایی چیست و چه مزیتی ایجاد می‌کند.
چندریختی قابلیتی است که یک ارجاع از جنس کلاس پدر به اشیایی از زیرکلاس ها ارجاع دهد. در نتیجه یک فراخوانی متد مشترک باعث رفتار متفاوتی در اشیا متفاوت خواهد شد.

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؟
از طریق متدها میتوان کارکردهای متنوعی در یک کلاس ایجاد کرد. به طور کلی متدها صفر یا چند پارامتر ورودی گرفته و پس از یک عملیات معین، صفر یا چند خروجی برمیگردانند.
بهتر است نامگذاری توابع به صورتی که بین برنامه نویسان یک زبان قرارداد شده انجام گیرد و همچنین براساس کارکرد متد نامگذاری شوند. همچنین از کپی کردن عملیات مشخص در چند متد بایستی خودداری کرد و به جای آن، یک تابع ایجاد کرده و آن را چندبار در برنامه فراخوانی کنیم.

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؟
سه نوع: استاتیک، غیراستاتیک و بی نام
کلاس بی نام: برای اجرای یک رفتار خاص به این صورت که در محل ایجاد شی کلاس را تعریف میکنیم.
کلاس استاتیک: به ارجاعی به شیئی از کلاس بیرونی دسترسی ندارد و هنگام نمونه سازی هم به شیئی از کلاس بیرونی نیاز نیست.
کلاس غیراستاتیک: یک ارجاع با پسوند .this به کلاس بیرونی وجود دارد و برای نمونه سازی از کلاس داخلی ابتدا باید یک شی از کلاس بیرونی ایجاد کنیم.
به کمک کلیدواژه super میتوان به متد کلاس بیرونی دسترسی داشت. در صورتی که این کلیدواژه ذکر نشود، متد موجود در کلاس درونی فراخوانی میشود.

سوال ۹

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

- تابع و متد **در هیچ یک از زیرکلاس ها نمیتواند override شود.**
- تعریف کلاس **هیچ کلاسی نمیتواند آن را extend کرده و زیرکلاس آن باشد.**
- یک متغیر از نوع شی **هویت متغیر ثابت است و به یک آبجکت ثابتی ارجاع میدهد.**
- یک متغیر از نوع پایه **مقدار متغیر ثابت بوده و تغییر نمیکند.**

سوال ۱۰

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

- تابع و متد **میتوان بدون اینکه آبجکتی از کلاس ساخت، متدهای استاتیک را فراخوانی کرد و این متدها به کلاس مرتبط هستند نه به اشیا ساخته شده از کلاس.**
- تعریف کلاس **به ارجاعی به شیئی از کلاس بیرونی دسترسی ندارد و هنگام نمونه سازی هم به شیئی از کلاس بیرونی نیاز نیست.**
- یک متغیر از نوع شی **بدون ساختن هیچ آبجکتی از کلاس، قابل دسترسی است و در هر آبجکتی که از کلاس ساخته شود، رفرنس استاتیک به یک شی واحد اشاره میکند.**
- یک متغیر از نوع پایه **بدون ساختن هیچ آبجکتی از کلاس، قابل دسترسی است و مستقل از اشیا فقط یک خانه از حافظه را اشغال میکند.**