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();
        Classes.B b = new Classes.B(10);
        Classes.A c = b;
//c "as a A class obj" and b "as a B class obj" both refer to same storage
unit

        b.print();
        c.print();
        ((Classes.A) b).print();
        b.printSuper();
        a.printCaller();
        b.printCaller();
        c.printCaller();
        incrementValue(a); //a.intvalue=1 , a.integervalue=21
        a.printCaller();
        incrementValue(b); //c,b.intvalue=11 , c,b.integervalue=16
        b.printCaller();
        incrementValue(c); //c,b.intvalue=12 , c,b.integervalue=17
        c.printCaller();
        incrementValue(b.intValue, b.integerValue); //c,b.intvalue=13 ,
c,b.integervalue=18
        b.printCaller();
        c.printCaller();
    }
}
```

Commented [1]: B: 10

Commented [2]: print method in class B

Commented [3]: A: 10

Commented [4]: print method in class A

Commented [5]: A: 10

Commented [6]: upcasting b to A class for this line

Commented [7]: A : 10

Commented [8]: printsuper is print method of inner class A

Commented [9]: A : 0

Commented [10]: B: 15
B: 15

Commented [11]: A: 10

Commented [12]: A: 1

Commented [13]: B: 16
B: 16

Commented [14]: B: 17

Commented [15]: B: 18
B: 18

Commented [16]: B: 18

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؟

با ارثبری میتوان کلاس های مرتب تری ساخت ینی کلاس هارا جوری تعریف کرد که به ساختار دنیای واقعی شبیه تر باشه که در آن سوپر کلاسها مشخصاتی دارند که بین تمام اشیا خود و زیرکلاسهایشان مشترک باشند. این باعث میشود که از کپی کردن کد پرهیز شود(ترکیب هم این فایده را دارد) و با تعریف کلاس های انتزاعی قوانینی برای تعریف زیرکلاسها مشخص کنیم.زمانی از ارثبری استفاده میکنیم که رابطه is a برقرار باشد

زمانی از ترکیب استفاده میکنیم که رابطه has a برقرار باشد

مثلا هر شغل نیاز به کارمند دارد پس در تعریف کلاس شغل از اشیا کارمند استفاده میکنیم

یا هر یک از سگ، گربه، ماهی و ... یک نوع حیوان هستند پس از سوپرکلاس حیوان ارث بری میکنند

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؟

از رابط ها وقتی استفاده میشود که ما بخواهیم کلیات رفتار اشیا را مشخص کنیم یا طراحی یک سری از کلاسها را توصیف کنیم. یا گاهی اوقات صرفا برای تعریف یک پروتکل استفاده میشود و در آن متد انتزاعی ای نداریم.

در رابط ها ویژگی و متد تعریف نمیکنیم(در حالی که در کلاس انتزاعی این کار را ممکن بود انجام دهیم) البته اگر متغیری تعریف کنیم ان متغیر ثابت استاتیک و پابلیک خواهد بود و در جاوا هشت هم میتوان به شکلی متد هایی تعریف کرد. ویژگی خوبی که رابط ها دارند این است که یک کلاس میتواند از چند رابط استفاده کند که اگر رابط ها درست تعریف شده باشند هم مشکلی پیش نمی آید ولی کلاس ها فقط از یک کلاس (چه معمولی و چه انتزاعی) میتوانند ارث بری کنند تا از تداخل سوپرکلاسها در رفتار و ویژگی های کلاس مورد نظر جلوگیری 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 انتزاعی کلاس هایی هستند که سازنده ندارند یعنی ما برای آنها شی تعریف نمیکنیم بلکه باید زیرکلاس هایی داشته باشند تا برای آنها شی تعریف شود. در این کلاسها میتوان متد هایی تعریف کرد که انتزاعی باشند و برنامه نویس باید آنها را در کلاسهایی که ازین کلاس ارث می‌برند بصورت کامل تعریف کند. تفاوتی که با رابط ها دارند این است که ما میتوانیم در یک کلاس انتزاعی متد هایی تعریف کنیم که انتزاعی نباشند و صرفا بخاطر انتزاعی بودن این کلاس برای آن شی تعریف نکنیم اما رابط ها بدون جزییات تعریف میشوند و در آن متد ها تنها بصورت بدون body نوشته میشوند. درمورد تفاوت در ارثبری هم در سوال بالا توضیح داده شده.

سوال ۵

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

باعث میشود که مطابق بحث چندریختی تابع یا متغیر مورد نظر عملکردی متفاوت از آنچه در سوپرکلاسش تعریف شده بود داشته باشه.

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← بای دیفالت اگه متد رو صدا کنیم به نتیجه مطلوب میرسیم
- متدی داخل کلاس فرزند← استفاده از پیشوند super
- خارج از دو کلاس← بسته به اینکه شی یا کلاسی که میخواهیم از آن ، متد را صدا بزنیم از super. استفاده میکنیم یا نمیکنیم. (با توجه به چندریختی) مثلا <<کلاس پدر.اسم متد استاتیک>> یا<<شی‌پسر.سوپر.متد اوررایدشده>>

سوال ۶

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

چندریختی یعنی همین که مثلا یک متد با توجه به نحوه‌ی استفاده از آن میتواند چندحالت گوناگون داشته باشه.چندریختی باعث میشود که از کد ها استفاده ی مجدد (reuse) کنیم. یعنی مثلا بتوانیم یک تابع را با توجه به پارامتر های ورودی به چند روش تعریف کنیم ولی با یک اسم ثابت. یا مثلا یک تابع را در کلاسهای مختلف ثبت کنیم و هربار با توجه به ارجاعی که شی دارد آن متد به یک روش خاص انجام شود .دراین حالت نیازی نیست که برای فراخوانی متد آدرس دقیق آن متد را صدا بزنیم و خود برنامه با توجه به کاری که ازش درزمان اجرا میخوایم میفهمه که از مثلا ده تا متدی که با اسم مشابه نوشتیم منظورمون دقیقا کدوم متده.

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؟

راستش سوالو خوب متوجه نشدم!خیلی بیسیکه! خب ما اگه توابع و متد هارو تعریف نکنیم هردفعه که یه کاری با یه چیزی توی برنامه بخوایم بکنیم باید کل کد تابعو دوباره نویسی کنیم و اگه بخوایم از چیزایی مثل ترکیب کلاس ها استفاده کنیم هم که دیگه واقعا خدابخیر کنه! بخوایم تابع بنویسیم هم نیازه که اسامی درستی برای اسم متد یا تابع و متغیر ها و ... استفاده کنیم و از کامنت یا چیزی مث جاواداک برای مستند سازی استفاده کنیم. و اینکه سعی کنیم از الگوریتم های پیچیده حتی الامکان دوری کنیم، تمیز کد بزنیم و مفهوم برنامه به مفاهیم دنیای واقعی نزدیک باشند. ایشالا که همینارو ازم خواسته بودین!

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؟

کلاس درونی دونوع استاتیک و غیر استاتیک دارد که نوع استاتیک شی ای میسازد که با ویژگی های استاتیک تابع کار دارد و نوع غیراستاتیک با مشخصات یک شی از آن کلاس کار دارد. برای ساخت شی استاتیک مثل سوال یک کوییز عمل میکنیم برای ساخت شی غیراستاتیک هم مثلا مینویسیم() outer.inner x= object.inner

در صورت override شدن یک متد یا متغیر توسط یک کلاس درونی چگونه می‌توان به نسخه override شده از کلاس بیرونی دسترسی پیدا کرد؟

اگه داخل کلاس داخلی بخوایم از متد بالاتری استفاده کنیم از .super استفاده میکنیم. در خارج از کلاس داخلی هم که متد دست‌نخورده باقی می‌ماند.

سوال ۹

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

- تابع و متد← غیرقابل بازنویسی(override) میشود
- تعریف کلاس← فکر کنم پس از گرفتن اولین شی دیگر شی ای نمیگیرد
- یک متغیر از نوع شی← محل حافظه ارجاع داده شده تغییر نمیکند
- یک متغیر از نوع پایه ← مقدار متغیر تغییر نمیکند

سوال ۱۰

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

- تابع و متد ← باعث میشود که تابع مربوط به کل کلاس و متغیر های استاتیک شود و با فراخوانی کلاس صدا زده میشود نه با فراخوانی شی
- تعریف کلاس ← کلاس استاتیک برای زیرکلاس ها تعریف میشود و باعث میشود که به ویژگی های غیر استاتیک کلاس دسترسی نداشته باشد . به عبارتی دیگر به شی ای از کلاس بیرونی اشاره نکند و به خود کلاس ارجاع داشته باشد
- یک متغیر از نوع شی← یک متغیر برای هر شی تعریف میشود
- یک متغیر از نوع پایه← یک متغیر برای کل کلاس تعریف میشود