فرهاد امان 9931006

1)

الف) در برنامه نویسی شی گرا هر شی که ساخته می شود دارای identity مختص به خود است که با هیچ شی دیگری برابر نیست حتی اگر دو شی از نظر نظر نوع و مقدار دقیقا شبیه به هم باشند . این تفاوت معمولا در محل ذخیره سازی شی نمایان می شوند دو شی با identity متفاوت در بخش های متفاوتی از حافظه ذخیره می شوند. در جاوا معمولا امکان فهمیدن مکان ذخیره اشیاء وجود ندارد در واقع ما دسترسی به آدرس نداریم.

State نشان دهنده مقادیر یک شی است به عنوان مثال دو رشته که مقدار یکسانی دارند دارای stateیکسان اما identity متفاوت می باشند.

Behavior رفتار های یک شی را نشان می دهد یا در واقع اعمالی که آن شی قادر به انجام آن است. نمونه های مختلف از یک کلاس دارای behavior های یکسان هستند. در جاوا متد ها تعیین کننده رفتار های اشیاء هستند.

ب) در جاوا تعامل بین اشیاء به وسیله متد های غیر private صورت می گیرد . هنگامی که یک شی از متد شیئی دیگر استفاده می کند تعامل اشیاء صورت می گیرد این تعامل معمولا یک طرفه است. متد های getter و setter برای تعامل اشیائ هستند زیرا فیلده ای یک شی معمولا به صورت private می باشند .

ج) هنگامی که در یک کلاس چند متد با نام یکسان اما پارامتر های متفاوت داریم overloading اتفاق می افتد. در این حالت حتما باید تعداد پارامتر ها یا نوع پارامتر ها تغییر کند. و تغییر دادن نوع خروجی به تنهایی نمی تواند باعث overloading شود. زیرا باعث ایجاد گنی در کد می شود .

به پروسه تبدیل یک نوع به نوع دیگر casting می گویند . در جاوا دو نوع casting وجود دارد .

Widening در این روش یک نوع به نوع بزرگتری تبدیل می شود مثلا int به long تبدیل می شود. این نوع از casting به صورت خودکار و ضمنی انجام می شود.

Narrowing در این روش یک نوع به نوع کوچکتری تبدیل می شود. مثل double به float این روش به صورت دستی و صریح انجام می شود.

برنامه نویسی ماژولار یک روش برنامه نویسی است که در آن نرم افزار از قسمت های مجزایی به نام ماژول ساخته شده است. یک ماژول با یک استاندارد و قالب مشخص طراحی می شود و انجام یکسری از فعالیت‌ها را

بر عهده دارد هر ماژول کاملا جدا از بخش های دیگر است و عملا به صورت مجزا قابلیت اجرا شدن دارد همچنین تغییر دادن آن مشکلی در اجرای بخش های دیگر برنامه ایجاد نیم کند.

در برنامه نویسی شی گرا abstraction عبارت است از پنهان کردن جزئیات پیاده سازی از دید کاربر.

در واقع کاربر تنها کاربرد و روش استفاده از یک شی را می داند نه اینکه چطور این کار را انجام می دهد.

د) هنگامی به یک شی immutable می گوییم که state های داخلی آن بعد ساخته شدن قابل تغییر نباشند. استفاده از این اشیا باعث می شود که امکان تداخل به 0 برسد و اشیاء مختلف تنوانند روی هم تاثیر مخرب بگذارند . یه مثال برای این اشیاء String است بعد از تعریف شدن یک string امکان تغییر دادن کاراکتر های مختلف آن وجود ندارد .

2)

1) خیر تنها در صورتی جاوا سازنده پیشفرض خود را استفاده می کند که ما سازنده ای برای کلاس خود در نظر نگرفته باشیم.
2) بله ArrayList به دلیل داشتن سایز پویا معمولا کندتر از آرایه معمولی است .
3) خیر ArrayList تنها امکان نگه داشتن اشیاء را دارد در جاوا کلاس هایی خاص برای نگه داری primitive type ها وجود دارد که می توانیم از آن ها استفاده کنیم .
4) بله این امکان با استفاده از overloading وجود دارد.
5) بله با استفاده از this می توان به هر عضو از current object اشاره کرد.
6) بله در جاوا متغیر ها باید قبل از استفاده حتما مقدار دهی شوند.
7) خیر به این متغیر ها local می گویند که در تنها در متدی که در آن تعریف شده اند قابل استفاده اند.
8) بله اگر هم در تعریف هم در سازنده مقدار دهی انجام شود مقدار نهایی مقدار سازنده خواهد بود .

3)

الف)

1. بله این امکان وجود دارد اما باید توجه کنیم که در هر صورت باید تغییر در پارامتر های متد عمال شود که ابن کار میتواند با تغییر تعداد پارامتر ها یا نوع پارامتر ها یا هر دو صورت گیرد سپس میتوانیم نوع خروجی متد را هم تغییر دهیم و امکان تغییر نوع خروجی به تنهایی و بدون تغییر پارامتر های ورودی وجود ندارد.

2. از سازنده برای مقدار دهی اولیه شی استفاده می شود در حالیکه متد کاربرد ها و رفتار های شی را تعیین میکند.

از متد به صورت صریح استفاده می شود. در صورتیکه از سازنده به صورت ضمنی استفاده می شود.

سازنده return type ندارد در حالیکه متد حتما دارای return type است هرچند void

در صورتیکه سازنده وجود نداشته باشد جاوا از سازنده پیشفرض استفاده می کند اما در متد چنین چیزی وجود ندارد.

نام سازنده حتما مانند نام کلاس است در حالیکه متد چنین نیست .

3. بله این امکان درست مانند overload کردن متد ها وجود دارد با توجه به اینکه سازنده هیچ return type ای ندارد تنها تغییر دادن پارامترهای ورودی امکان پذیر است . میتوان با تغییر در تعداد پارامتر ها یا تغییر نوع آن ها overload کرد هنگام ایجاد یک شی جدید از کلاس میتوان با دادن ورودی های مختلف از سازنده های متفاوت استفاده کرد.

ب)

1. نام کلاس با حرف بزرگ شروع می شود.
2. Id باید به صورت int باشد نه String پس از داخل”” خارج می شود.
3. سازنده هیچ نوع خروجی ندارد پس عبارت void حذف می شود.
4. FirstName باید به صورت firstName و last_name باید به صورت lastName باشد.
5. باید از عبارت this برای اشاره به فیلد های شی فعلی استفاده کنیم.

```
public class Main {

    public static void main(String[] args) {
        Person p1 = new Person("Ted", "Mosby", 123456);
        p1.express();
        p1.express("Happy");
        p1.print();
    }

    static class Person {
        private String firstName;
        private String lastName;
        private int id;
```

```
        public Person(String firstName, String lastName, int id){
            this.firstName = firstName;
            this.lastName = lastName;
            this.id = id;
        }

        public void express(){
            System.out.println("I feel neutral");
        }

        public int express(String state){
            System.out.println("I feel " + state + " today");
            return 0;
        }

        public void print(){
            System.out.println("person{" +
                    "name='" + firstName + '\'' +
                    ", lname='" + lastName + '\'' +
                    ", id=" + id +
                    '}');
        }
    }
}
```

4)

الف)

1. Arrayاز خانه های متوالی در حافظه تشکیل شده و دسترسی به هر خانه بدون نیاز به پیمایش خانه های پیشین و با استفاده از index خنه اماکن پذیر است. ArrayList یک ارایه پویا می باشد. که امکان تغییر سایز حذف و یا اضافه کردن المان به آن وجود دارد و البته از آرایه کند تر می باشد . LinkedList از تعدادی المان مجرا تشکیل شده که الزامی برای پشت سر هم بودن آن ها در حافظه وجود ندارد. در هر المان علاوه بر محتوا یک رفرنس به المان بعدی وجود دارد که به این صورت امکان پیمایش المان های LinkedList به وجود می آید .المان های LinkedList دارای index نیستند و برای دسترسی به یک المان باید تمام المان های پیش از آن نیز پیمایش شوند .

2. ابتدا میدانیم بخاطر متغیر بودن تعداد المان ها Array گزینه مناسبی نیست بین LinkedList و ArrayList باتوجه به اینکه سرعت LinkedList بیشتر از ArrayList می باشد و با توجه به ماهیت بیمارستان که سرعت زیادی مورد نیاز است LinkedList میتواند گزینه بهتری باشد.
3. بله این امکان در جاوا وجود دارد.
4. Length یک فیلد با مقدار ثابت در آرایه است که تعداد خانه های آرایه و همچنین مقدار فضای allocate شده از رم هنگام ایجاد آن آرایه را مشخص میکند که قابل تغییر نیست اما متد size در ArrayList با هربار صدا زده شدن تعداد المان های لیست را محاسبه و بر میگرداند و این مقدار هربار میتواند متفاوت باشد.