به نام خدا

محمد رضا حسن زاده **۹۷۳۹۰۲۱** **تمارین برنامه نویسی پیشرفته**

سوال اول:

حالت یا State : هر شیای یکسری خصوصیات و یا ویژگی هایی(attributes) دارد که به آنها fields گفته می‌شود. به عنوان مثال انسان، نام دارد. به مجموعه مقادیر این ویژگی ها و خصوصیت ها، حالت یا State آن شی گفته می‌شود.به طور مثال انسان(شی) علی است. نام خانوادگی او رضایی است و سن او نیز ۱۲ سال است. یک انسان دیگر (شی) نامش حسین است، نام خانوادگی او دربندی است و او ۴۵ سال سن دارد. هر دو انسان هستند اما با حالت ها یا State های متفاوت.

رفتار Behavior : هر شیای که ما تعریف می‌کنیم یک رفتاری دارد. یعنی یک کاری می‌تواند انجام دهد یا به عبارتی یک operations برای او تعریف می‌شود. رفتار یک نوع شی عملکرد آن را تعیین می‌کند. به طور مثال انسان می‌تواند راه برود. این رفتار این شی (انسان) است که می‌تواند راه برود.

هویت Identity : هویت ویژگی یک شی را توصیف می‌کند که آن را از دیگر اشیا متمایز می‌کند. به عبارت دیگر یک شی با هویت موجودی مستقل است نسبت به State و ارزش های خود به طور مثال چند شی مثلا چند انسان را در نظر بگیریم این انسان ها با یکدیگر یکی نیستند و هرکدام هویت مخصوص به خود را دارند شاید با یکدیگر State های یکسانی داشته باشند مثلا نام هر دو علی باشد نام خانوادگی آن ها حسینی باشد سن هر دو ۴۵ سال باشد اما به طور مثال آدرس خانه هایی که در آن زندگی می‌کنند با یکدیگر متفاوت اند.

سوال دوم:

برنامه نویسی ساخت یافته (Structured programming): یک پارادایم برنامه نویسی است که طبق آن برنامه نویس قدمها و روالهایی را که لازم است تا برنامه به جواب برسد را مشخص میکند. در این روش از برنامه نویسی، انجام یک روال به روالهای کوچکتر تقسیم میشود و به این ترتیب یک برنامه با شکسته شدن به ریز برنامه های کوچکتر سعی میکند تا عملکرد مد نظر را پیاده سازی کند.

برنامه‌نویسی شی گرا (Object-Oriented Programming) یک شیوه برنامه‌نویسی است، که ساختار یا بلوک اصلی اجزای آن، شی‌ها می‌باشند. در این شیوه برنامه‌نویسی، برنامه به شی گرایش پیدا می‌کند، به این معنا که داده‌ها و توابعی که بر روی این داده‌ها عمل می‌کنند، تا حد امکان در قالبی به نام شیء و در کنار یکدیگر قرار گرفته، جمع‌بندی شده و یک واحد (یا یک شی) را تشکیل می‌دهند و نسبت به محیط بیرونِ خود، کپسوله می‌شوند. از این طریق، توابع خارج از آن شیء، امکان ایجاد تغییر در داده‌های درون شی را نخواهند داشت.

مزایای برنامه نویسی شی گرایی نسبت به ساخت یافته:

۱ - قابلیت سازمان دهی بهینه تر کدها

۲ - قابلیت تقسیم برنامه به برنامه‌های کوچک‌تر اما مستقل. برنامه اصلی به صورت یک exe در می‌آید که دیگر قسمت‌های مستقل برنامه را فراخوانی می‌کند.

۳ - عدم نیاز به نوشتن کدهای تکراری و قابلیت‌هایی که قبلاً پیاده‌سازی شده‌اند و صرف جویی در استفاده از منابع.

۴ - به دنیای واقعی نزدیک است و طراحی سیستم های پیچیده و بزرگ را ساده تر می کند.

۵ - انواع داده های ترکیبی و پیچیده را با استفاده از مفاهیمی مانند وراثت پشتیبانی می کند.

۶ - با امکاناتی مانند Encapsulation کار تیمی را ساده تر می سازد.

معایب برنامه نویسی شی گرایی نسبت به ساخت یافته:

۱ - سرعت اجرای برنامه های نوشته شده با پارادیم شی گرایی به طور معمول از برنامه های نوشته شده با پاردایم ساخت یافته پایین تر است.

۲ - در پارادایم شی گرایی به طور معمول تعداد خط کدها بیشتر است.

۳ - فرایند تفکر در برنامه نویسی شی گرا ممکن است برای همه‌ی افراد کار آسان و راحتی نباشد و افراد برای عادت کردن به آن نیازمند زمان باشند. به طور مثال مفاهیمی مانند پلی مورفیسم

۴ - به دلیل آنکه بسیاری از مسائل به واقعیت شباهت دارند با پارادایم شیءگرایی حل آنها بسیار راحت تر است اما مسائل و مشکلات زیادی هم وجود دارند که نمیتوان آنها را با این رویکرد حل کرد.

سوال سوم:

در زبان برنامه نویسی C ، استرینگ ها صرفا آرایه‌ای یا پوینتری از کاراکتر ها هستند که پایان آن ها با NULL بسته می‌شوند. ساختن هر استرینگ به مانند ساختن یک آرایه در این زبان برنامه نویسی می‌باشد.

اما در زبان برنامه نویسی java استرینگ ها دیگر یک آرایه نیستند. بلکه شی هایی از کلاس java.lang.String هستند. این استرینگ ها داده های کاراکتری را نمایش می‌دهند ولی پیاده سازی این کلاس به برنامه نویس نشان داده نمی‌شود. در جاوا نمی‌توانیم با استرینگ ها به مانند آرایه ها برخورد بکنیم ولی اگر نیاز باشد می‌توانیم آن ها را به آرایه های کاراکتری با استفاده از متد toCharArray تبدیل بکنیم.

تابع **charAt**:

این تابع یک ورودی صحیح می‌گیرد و کاراکتری را که در آن اندیس هست برمی‌گرداند به عنوان مثال اگر "salam" را در یک استرینگی داشت هباشیم و روی آن تابع charAt(۲) را صدا بزنیم به ما خروجی را کاراکتر 'l' می‌دهد.

```
String temp = new String("salam");
System.out.print(temp.charAt(2));
OUTPUT : 'l'
```

تابع **replace**:

این تابع روی یک استرینگ صدا زده می‌شود و در ورودی خود دوتا کاراکتر به عنوان ورودی می‌گیرد سپس جای کاراکتر اول را با کاراکتر دوم در استرینگی که صدا زده شده عوض می‌کند و به عنوان خروجی یک استرینگ دیگر می‌دهد.

```
String temp = "salam";
System.out.print (temp.replace('a', 'o'));
OUTPUT : "solom"
```

تابع **equals**:

این تابع یک استرینگ را به عنوان ورودی می‌گیرد و سپس روی یک استرینگ دیگری صدا زده می‌شود و بررسی می‌کند که آیا محتوا های ذخیره شده داخل این دو متغیر با یکدیگر برابرند یا خیر و خروجی را به صورت یک بولیون به ما می‌دهد.

```
String temp_1 = "salam";
String temp_2 = "salam";
System.out.print (temp_1.equals(temp_2));
OUTPUT : true
```

تابع **split** :

این تابع روی یک استرینگ صدا زده می‌شود و به عنوان ورودی یک استرینگ می‌گیرد که به آن رجکس (یک الگویی برای تشخیص و جدا سازی رشته ها) گفته می‌شود و ورودی دوم یک عدد صحیح می‌گیرد به عنوان limit. این تابع با الگوی رجکس داده شده استرینگی که روی آن صدا زده شده را تبدیل به یک آرایه‌ای از استرینگ ها می‌کند. آن عدد صحیح می‌تواند در سه بازه قرار داشته باشید اگر بزرگتر از صفر باشد الگو را به تعداد دفعاتی که معلوم شده به عنوان limit روی استرینگ اجرا می‌کند و استرینگ اصلی را به اندازه‌ی limit به آرایه‌ای از رشته ها توسط الگو جدا می‌کند. اگر مقدار limit کمتر از صفر باشد الگو تا آخر رشته اجرا می‌شود و رشته را از یکدیگر جدا می‌کند. اگر مقدار limit برابر با صفر باشد الگو تا آخر رشته اجرا می‌شود و رشته را از یکدیگر جدا می‌کند و همچنین آرایه هایی را که خالی هستند و تشکیل شده اند را پاک می‌کند.

```
String str = "salam@bar@shoma";
String[] arrOfStr = str.split("@", 2);
for (String a : arrOfStr) {
    System.out.println(a);
}
```

OUTPUT : salam

bar@shoma

تابع **concat** :

این تابع یک استرینگ به عنوان ورودی می‌گیرد و این استرینگ را به آخر استرینگی که روی آن صدا زده شده را اضافه می‌کند و به عنوان یک استرینگ درگز آن را خروجی می‌دهد.

مانند مثال زیر:

```
String s = "salam";
s = s.concat("! is the best.");
System.out.println(s);
```

OUTPUT: salam! is the best.

تابع **format** :

این تابع به مانند printf در زبان c یک رشته به عنوان ورودی می‌گیرد که در آن فرمت هایی مانند %s یا %f و .. وجود دارد. این تابع یک آرگومان هایی را هم به عنوان ورودی می‌گیرد تا با این فرمت ها جایگزین کند سپس استرینگ فرمت شده را خروجی می‌دهد.

```
String name="sonoo";
String sf1=String.format("name is %s",name);
String sf2=String.format("value is %f",32.33434);
System.out.println(sf1);
System.out.println(sf2);
```

OUTPUT: name is sonoo

value is 32.334340

تابع **indexOf** :

این تابع اندیس اولین جایی را که ورودی در آن اتفاق افتاده است را برمی‌گرداند. اگر ورودی در رشته مجود نباشد ۱- برمی‌گرداند به عنوان مثال:

```
String str = "Hello world, welcome to the universe.";
int n = str.indexOf("welcome");
System.out.print(n);
```

OUTPUT:13