شی‌گرایی

کلاس‌های لازم را به گونه‌ای پیاده‌سازی کنید که متد `main` زیر اولاً بدون خطا کامپایل شده و ثانیاً خروجی اجرای آن دقیقاً به شکلی که گفته شده است باشد.

```java
public static void main(String[] args) {
    A[] elements = { new D(), new A(), new C(), new B() };
    for (int i = 0; i < elements.length; i++) {
        System.out.println(elements[i].method1());
        System.out.println(elements[i].method2());
        System.out.println();
    }
}
```

خروجی مورد انتظار:

```
D1
D1B2

A1
A2

C1
C1B2

A1
A1B2
```

در پیاده‌سازی خود، موارد زیر را باید رعایت کنید. رعایت نکردن هر یک از این موارد، موجب کسر امتیاز می‌شود:

- به جز کلاس `A` ، سایر کلاس‌ها تنها یک متد می‌توانند داشته باشند.
- هیچ کلاس و متد اضافه‌ای ایجاد نکنید.
- هیچ ویژگی (*field*)ای در هیچ کلاسی ایجاد نکنید.

بیمارستان

مدیر یک بیمارستان از شما خواسته تا برنامه‌ای برای او بنویسید که به کمک آن بتواند پرسنل خود را مدیریت کند. پرسنل بیمارستان شامل پزشک، پرستار و خدمه است. همچنین، برای خود مدیر هم کلاسی باید در نظر بگیرید. تمام پرسنل اعم از خود مدیر باید فیلدی به نام `workingHours` از نوع `int` داشته باشند که ساعات کاری آن‌ها را بیان می‌کند. همچنین باید دارای دو فیلد `ID` از نوع `String` و `name` از نوع `String` باشند و فقط مدیر لیست تمام کارکنان را داشته باشد.

متد های زیر را برای پرسنل پیاده‌سازی کنید:

- متد `int findSalary(Personnel p)` : متدی برای محاسبه‌ی حقوق که برای هر نوع پرسنل از فرمولی مجزا به‌دست می‌آید (پیاده‌سازی فرمول آن دلخواه است ولی باید با ساعات کاری رابطه‌ی مستقیم داشته باشد؛ مثلاً `2 * workingHours` قابل قبول است.)
- متد `String compareSalary(Personnel p)` : تمام افراد باید این متد را داشته باشند که فرد دیگری را به‌عنوان پارامتر می‌گیرد و حقوق آن را با فرد مقایسه می‌کند و عبارت `more` ، `less` یا `equal` را به‌همراه میزان اختلاف برمی‌گرداند.
- متدهای `Constructor` : هنگام ساختن نمونه از هر کلاس باید `name` و `workingHours` آن را مشخص کنید و اگر ساعات کار را وارد نکنید، برنامه نباید ارور دهد و باید خود یک مقدار پیشفرض را در نظر بگیرد. همچنین، `ID` نباید به‌صورت مستقیم از خارج کلاس قابل دسترسی باشد و باید از طریق *getter* و *setter* به آن دسترسی داشته باشید. هنگام *set* کردن `ID` یک نفر، اگر فردی با `ID` او در سیستم وجود داشته باشد، نباید اجازه‌ی این کار داده شود و ارور مناسب چاپ شود (برنامه متوقف شود).
- متد `void hire(Personnel p)` : فقط مدیر اجازه استفاده از آن را دارد. مدیر باید بتواند از کلاس خود هر کدام از پرسنل (نمونه‌ها) را به کمک متد `hire` استخدام و در سیستم خود ذخیره کند.
- متد `void fire(String ID)` : با فراخوانی این متد، پرسنل با `ID` داده شده از لیست مدیر اخراج می‌شود و جای خالی او با کارمند بعدی پر می‌شود. برای راحتی کار، تعداد پرسنل را محدود در نظر بگیرید (استفاده از `Collection` ها مجاز نیست).

شما باید علاوه بر پزشک، جراح هم داشته باشید که تمام ویژگی‌های پزشک را دارد، اما حقوق او علاوه بر ساعات کار به تعداد جراحی‌های او نیز وابسته است.

در نهایت، در کلاس main برنامه و کارآرایی همه ی متدها را تست کنید.

بانک

می‌خواهیم چند جزء کوچک از سیستم بانکی را شبیه‌سازی کنیم. چهار کلاس برای شعبه، مشتری، حساب و کارت اعتباری با توضیحاتی که برای ویژگی‌هایشان در زیر آمده است، بسازید.

شعبه

یک کلاس با نام `Branch` ایجاد کنید که سه ویژگی زیر را داشته باشد:

- کد (`code`): یک رشته از حروف و اعداد است که کد شعبه را نشان‌می‌دهد.
- شهر (`city`): نام شهری که شعبه در آن قرار دارد را نشان می‌دهد.
- درجه (`rate`): یک عدد صحیح از بین اعداد ۱ و ۲ و ۳ است که درجه‌ی شعبه را نشان می‌دهد.

حساب

یک کلاس با نام `Account` ایجاد کنید که دو ویژگی زیر را داشته باشد:

- شماره حساب (`accountNumber`): رشته‌ای از اعداد است که شماره حساب را نشان می‌دهد.
- مانده‌ی حساب (`balance`): یک عدد اعشاری است که میزان مانده‌ی حساب را نشان می‌دهد.

کارت اعتباری

یک کلاس با نام `CreditCard` ایجاد کنید که دو ویژگی زیر را داشته باشد:

- شماره کارت (`cardNumber`): رشته‌ای است از اعداد که شماره‌ی کارت را نشان می‌دهد.
- اعتبار (`credit`): یک عدد اعشاری است که میزان اعتبار باقی‌مانده در کارت را نشان می‌دهد.

مشتری

یک کلاس با نام `Customer` ایجاد کنید که سه ویژگی زیر را داشته باشد:

- نام (`name`): رشته‌ای از حروف است که نام مشتری را نشان می‌دهد.
- حساب (`account`): شی‌ای از جنس `Account` است که حساب مشتری را نشان می‌دهد.
- کارت اعتباری (`creditCard`): شی‌ای از جنس `CreditCard` است که کارت اعتباری مشتری را نشان می‌دهد.

پس از این که کلاس‌ها و ویژگی‌هایشان را تعریف کردید، متدهای لازم برای مقداردهی و فراخوانی ویژگی‌ها (*setter* و *getter*) را نیز بنویسید.

هم‌چنین، یک متد با نام `calCustomerBalance` در کلاس مشتری ایجاد کنید که مجموع اعتبار باقی‌مانده در حساب و کارت اعتباری مشتری را محاسبه کرده و برمی‌گرداند.

نکات

- در هر کلاس، ویژگی‌ها را به صورت یک بخش خصوصی (*private*) و متدها را به صورت یک بخش عمومی (*public*) تعریف کنید.
- در هر کلاس، برای تمامی ویژگی‌ها باید دو متد *setter* و *getter* را تعریف کنید.
- نام‌گذاری *setter* و *getter*ها را به درستی انجام دهید؛ مثلاً برای ویژگی `name` ، نام متد مقداردهی آن باید `setName` و نام متد فراخوانی مقدار آن باید `getName` باشد.
- در این سؤال فرض شده که کارت اعتباری و حساب برای مشتری جدا از هم هستند، یعنی میزان باقی‌مانده‌ی آن‌ها ربطی به یکدیگر ندارد.
- نوع (*type*) هر یک از ویژگی‌ها که صراحتاً بیان نشده‌اند را باید با توجه به توضیحات داده‌شده خودتان تشخیص دهید و از نوع مناسبی استفاده کنید.

پشته

درمورد ساختمان داده‌ی پشته (*stack*) مطالعه کنید و کلاس پشته را با توجه به توابع زیر پیاده‌سازی کنید.

تضمین می‌شود که بیش‌تر از **۱۰۰ عضو** به پشته اضافه نشود. **استفاده از کلاس** `Stack` **جاوا مجاز نیست.**

```
public boolean empty();
```

اگر پشته خالی باشد، مقدار `true` را برمی‌گرداند.

```
public int size();
```

این متد، اندازه‌ی پشته را بر می‌گرداند.

```
public String push(String element);
```

این متد، عنصر `element` را به پشته اضافه می‌کند و همان را برمی‌گرداند.

```
public String pop();
```

این متد، عنصر سر پشته را برمی‌گرداند و آن را حذف می‌کند. اگر پشته خالی باشد، رشته‌ی `"error"` را برمی‌گرداند.

```
public String peek();
```

این متد، عنصر سر پشته را بدون حذف کردن برمی‌گرداند. اگر پشته خالی باشد، رشته‌ی `"error"` را برمی‌گرداند

```
public int search(String element);
```

به دنبال عنصر `element` می‌گردد. اگر آن‌را پیدا کند، اندیسش را برمی‌گرداند و در غیر این‌صورت، `-1` را برمی‌گرداند. اندیس را برای اولین عضو اضافه شده صفر، برای دومین عضو اضافه شده ۱ و به همین ترتیب برای آخرین عضو اضافه شده به اندازه‌ی یکی کم‌تر از اندازه‌ی پشته در نظر بگیرید.

```
public void clear();
```

پشته را خالی می‌کند.

```
public String toString();
```

عناصر پشته را به شکل رشته و به فرم زیر برمی‌گرداند:

```
[element1, element2, element3]
```