به نام خدا

تمرین دوم درس برنامه‌نویسی پیشرفته

نیم‌سال اول ۱۳۹۹-۱۴۰۰

۰. فایل مربوط به توضیحات نحوه ارسال تمرین‌ها را که در مودل قرار دارد، مطالعه کنید.

۱. تمامی فایل‌های کد را به همراه فایل متنی که در قالب pdf است (مورد سوم را بخوانید) به صورت یک فایل آرشیو zip (zip != rar) که به قالب زیر نام‌گذاری شده است، بارگذاری نمایید.

StudentNumber_FirstName_LastName.zip

9031066_Ehsan_Edalat.zip

۲. در سوال‌هایی که ورودی و خروجی مطلوب آن‌ها مشخص شده است، برنامه‌ی شما به صورت ماشینی تصحیح می‌شود. بنابراین رعایت نحوه ورودی‌گرفتن و نمایش خروجی اهمیت بسیاری دارد. دقیقا همان‌طور که از شما خواسته شده است ورودی‌ها را خوانده و خروجی‌ها را تولید کنید.

۳. پاسخ سوالات تشریحی را به صورت تایپ‌شده و در قالب یک فایل pdf (برای کل تمرین) تحویل دهید.

۴. در صورت مشاهده هرگونه تقلبی، طبق موارد گفته شده در قوانین درس برخورد خواهد شد.

۵. در صورت وجود هرگونه ابهام می‌توانید از طریق ربات تلگرامی AP_Admin_bot با تدریس‌یاران در ارتباط باشید.

مهلت تحویل: تا جمعه ۱۸ مهر ۱۳۹۹ ساعت ۲۳:۵۵ شب

فهرست سوالات

صفحه

سوال اول

در جلسات گذشته، شما در کلاس درس، با مفاهیم کلاس[1] و شی[2] آشنا شدید. در ادامه سوالاتی از این مفاهیم پرسیده می‌شود که شما باید به این سوالات به صورت کامل و واضح پاسخ دهید:

۱. کلاس چیست؟ چه تفاوتی با شی دارد؟

۲. سازنده[3] کلاس یک تابع[4] است، این تابع به چه شکل از باقی توابع کلاس تمیز داده می‌شود؟

۳. در ادامه قطعه کدی را مشاهده می‌کنید. در این کد قصد داریم تا در سازنده‌ی کلاس `Book` موجود در فایل `Library.java`، متغیر کلاس (یا به اصطلاح `field` کلاس) با نام `title` را مقداردهی کنیم. آیا کد زیر این عمل را به درستی انجام می‌دهد؟ در صورتی که در این کد، خطاهایی مشاهده می‌کنید، آن خطاها را اصلاح کنید.

```
public class Book {
    private String title; // Class field
    public book(String title){
        title = title
    }
}
```

[1] Class

[2] Object

[3] Constructor

[4] Function

سوال دوم

درستی یا نادرستی عبارات زیر را مشخص کنید و عبارات غلط را نیز اصلاح کنید.

آ) در صورتی که ما یک سازنده با چند آرگومان[5] ورودی تعریف کنیم، زبان جاوا نیز یک سازنده پیشفرض[6]، بدون آرگومان برای کلاس، درنظر می‌گیرد.

ب) سطح دسترسی سازنده کلاس، حتما باید از نوع `public` باشد.

پ) اعضای `protected` در یک کلاس، تنها برای اعضای همان کلاس قابل دسترسی هستند.

ت) متغیرهایی که در درون توابع تعریف می‌شوند، به عنوان `field` کلاس شناخته شده و دیگر توابع می‌توانند به آن‌ها دسترسی داشته باشند .

سوال سوم

در برنامه‌نویسی با جاوا، شما با مفاهیمی از قبیل `Primitive types`، `Overloading`، `Getter` و `Setter` آشنا می‌شوید. در ادامه سوالاتی از این مفاهیم از شما پرسیده می‌شود.

۱. `Primitive types` در جاوا چه مواردی شامل می‌شوند؟ این تایپ‌ها شامل چه مقادیری هستند و دامنه تغییرات آن‌ها چیست؟

۲. مفهوم `overloading` در توابع به چه معناست؟

۳. `Getter` و `Setter` در کلاس‌ها چه هستند و چه کاربردی دارند؟

[5] Arguments

[6] Default constructor

سوال چهارم

(پیاده‌سازی)

در این سوال قصد داریم تا با حلقه‌ها در جاوا آشنا شویم. کلاسی به نام Shape خواهیم داشت، این کلاس برای ترسیم و محاسبه مساحت اشکال هندسی استفاده می‌شود و توابعی برای رسم و محاسبه شکل دارد که در ادامه به ذکر آن خواهیم پرداخت. ورودی برنامه شامل لیستی از اعداد است که با space از هم جدا شده است. عدد اول نشان‌دهنده شکل هندسی مورد نظر است که یکی از موارد

- مربع: ۰
- مستطیل: ۱
- مثلث قائم‌الزاویه: ۲
- صلیب: ۳

را شامل می‌شود.

ورودی‌های بعدی با توجه به نوع شکل تعداد متفاوتی خواهد داشت که در ادامه ذکر می‌کنیم:

* مربع: یک عدد وارد می‌شود که نشان‌دهنده طول این مربع است.
* مستطیل: دو عدد وارد می‌شود که عدد اول نشان‌دهنده طول و عدد بعدی عرض آن است.
* مثلث قائم‌الزاویه: یک عدد وارد می‌شود که نشان‌دهنده ارتفاع مثلث است (این مثلث متساوی‌الساقین است).
* صلیب: دو عدد فرد بزرگتر از یک ($3 \geq$) که اولی نشان‌دهنده قطر عمودی و عدد دوم نمایان قطر افقی است.

از شما خواسته می‌شود تا این اشکال را با «*» و در ادامه‌ی آن تعداد * به کار رفته را نمایش دهید.

توابعی که باید پیاده‌سازی کنید موارد زیر هستند.

```
public void drawSquare(int length)
public void drawRectangle (int length, int width)
```

```
public void drawTriangle (int height)
public void drawCross(int length, int width)
```

در کلاس Main، نمونه‌ای از کلاس Shape بسازید و توابع بالا را فراخوانی کنید.

برای درک بهتر به مثال‌های زیر توجه کنید:

مثال یک:

```
Input:  0 2
Output:
**
**
4
```

*** مربعی با طول ۲ ساخته می‌شود که دارای ۴ کاراکتر * است.**

مثال دو:

```
Input:  1 2 3
Output:
***
***
6
```

مثال سه:

```
Input:  2 3
Output:
*
**
***
6
```

مثال چهار:

```
Input:  3 5 3
Output:
 *
 *
***
 *
 *
7
```

*** به شکل صلیب دقت کنید که دو قطر در نقاط وسط متقاطع هستند.**

سوال پنجم

(پیاده‌سازی)

یک فروشگاه برای مدل‌کردن ذخیره‌سازی هر یک از اجناس خود از کلاسی با مشخصات زیر استفاده می‌کند (نام این کلاس را `Item` در نظر بگیرید):

۱. کلاس `Item` دارای فیلدهای نام کالا (`name`)، نام تولیدکننده کالا (`producer`)، قیمت اولیه کالا (`price`)، تخفیف (`discount`) و تعداد موجودی آن در انبار (`amount`) است.

۲. متدهایی برای افزایش (`increment`) و کاهش (`decrement`) تعداد موجودی در کلاس `Item` وجود دارد که با گرفتن تعداد، به مقدار لازم به موجودی یک کالا اضافه یا کم می‌کند. تابعی نیز برای اعمال مقدار تخفیف وجود دارد (`setDiscount`). همچنین متد دیگری برای نمایش (`print`) نام کالا به همراه نام تولیدکننده آن، قیمت اولیه، قیمت قابل پرداخت (پس از اعمال تخفیف) و مقدار موجودی آن در نظر بگیرید.

۳. constructor کلاس Item، نام کالا، قیمت اولیه و نام تولیدکننده را به عنوان پارامترهای ورودی گرفته و به فیلدهای کلاس انتساب می‌دهد و تعداد موجودی و مقدار تخفیف به طور پیش‌فرض صفر در نظر گرفته شوند.

۴. در این کلاس تابعی برای محاسبه قیمت قابل پرداخت برای کالا بنویسید (خروجی این تابع int می‌باشد).

این کلاس را پیاده‌سازی کرده و برنامه‌ای بنویسید که با استفاده از این کلاس تعداد ۵ عدد شکلات فرمند و ۳ عدد پفک چاکلز را نگه‌داری کند، برای شکلات‌های فرمند ۳۰ درصد تخفیف درنظربگیرد و در پایان، همه نمونه‌های Item ساخته‌شده را نمایش دهد.

سوال ششم

(پیاده‌سازی)

پیاده سازی برنامه مدیریت خط تولید و انبار یک کارخانه .

در این سوال شما باید برنامه‌ای بنویسید تا حسابدار یک شرکت به راحتی بتواند به میزان تولیدات هرخط تولید کارخانه و نوع کالای تولید شده در آن دسترسی داشته باشد. همچنین او باید بتواند تعداد خطوط تولید را اضافه یا کم کند . همچنین باید میزان موجودی هر کالا در انبار را بررسی و بتواند آن را در صورت لزوم افزایش یا کاهش دهد.

موجودیت‌ها و کلاس‌های این برنامه را در ادامه ذکر خواهیم کرد. شما باید تمامی مواردی که گفته شده است را پیاده‌سازی کنید.

کلاس محصول Product: محصولاتی که شرکت تولید می‌کند از این جنس است.

* **فیلدها:**

- نام name

- قیمت محصول price

این مقادیر در سازنده کلاس مقداردهی می‌شوند.

کلاس خط تولید `ProductionLine`: این کلاس خط تولید یک محصول است.

*** فیلدها:**

- نام خط تولید `name`
- محصولی که تولید می‌کند `type` (این فیلد از جنس `Product` است)

*** توابع:**

- `print`

وضعیت خط تولید را نشان می‌دهد. برای مثال خط تولید با نام `LineName1` و محصول `ProductName1` خروجی زیر را نمایش می‌دهد:

```
Name: LineName1 | Product: ProductName1
```

کلاس انبار `Store`: این کلاس محصولات شرکت را ذخیره می‌کند.

*** فیلدها:**

- لیستی ازمحصولات `products` (این فیلد یک `ArrayList` از `Product` است)
- لیستی از تعداد هر نوع کالا `prices`

*** توابع:**

- `addProduct`

با گرفتن یک اسم کالا، قیمت آن و تعداد آن، یک کالا به انبار اضافه می‌کند و یا در صورت موجود بودن آن کالا در انبار صرفا بر تعداد موجودی آن می‌افزاید.

- `RemoveProduct`

با گرفتن نام کالا و تعداد، در صورت موجود بودن در انبار، آن را از انبار کم و درصورت اتمام، از لیست محصولات حذف می‌کند.

- `CalcValue`

این تابع ارزش کل محصولات در انبار را نشان می‌دهد.

کلاس `FinancialSystem`: کلاس مدیریت کارخانه

❋ فیلدها:

- لیستی از خطوط تولید `productionLines` (این فیلد یک `ArrayList` از `ProductionLine` است)

- نمونه ای از انبار `store` (از جنس `Store`)

❋ توابع:

- `printAllProductionLines`

تمامی خطوط تولید را نشـان می‌دهد، برای مثال شـرکتی که دارای دو خط تولید با نام‌های `LineName1` با محصـول `ProductName1` و `LineName2` با محصول `ProductName2` هست به شکل زیر نمایش داده می‌شود:

```
Name: LineName1 | Product: ProductName1

Name: LineName2 | Product: ProductName2
```

این تابع باید از تابع `print` در کلاس `ProductionLine` استفاده کند.

- `addProductionLine`

با گرفتن اسم خط تولید و نام کالای تولید شده و قیمت هر کالا به خطوط تولید شرکت می‌افزاید.

- `removeProductionLine`

با گرفتن اسم یک خط تولید آن را از لیست خطوط تولید حذف می کند.

- `printAllStorageProducts`

تمـامی کـالا هـای موجود در انبـار وظرفیـت آن را نمـایش می‌دهـد بـه عنوان مثـال انبـار دارای محصـــولی بـه نـام `ProductName1` با ظرفیت ۱۰ عدد است و محصول با نام `ProductName2` با ظرفیت ۵ تا است.

```
ProductName: ProductName1 | Quantity: 10

ProductName: ProductName2 | Quantity: 5
```

- `addProductToStore`

با گرفتن یک اسم کالا، قیمت آن و تعداد آن، یک کالا به انبار اضافه می‌کند و یا در صورت موجود بودن آن کالا در انبار صرفا بر تعداد موجودی آن می‌افزاید. این تابع، باید تابع `addProduct` را در کلاس `Store` فراخوانی کند.

- `removeProductFromStore`

با گرفتن نام کالا و تعداد در صورت موجود بودن در انبار آن را از انبار کم و درصورت اتمام، از لیست محصولات حذف می‌کند. این تابع، باید تابع `RemoveProduct` را از کلاس `Store` فراخوانی کند.

- `printValueOfStore`

مجموع کل قیمت های کالاهای درون انبار را نمایش می‌دهد. این تابع، باید خروجی تابع `CalcValue` در `Store` را نمایش دهد.

شما باید در کلاس `Main` نمونه‌ای از کلاس `FinancialSystem` بسازید و توابع آن را فراخوانی کنید.