

به نام خدا

پاسخ تمرین دوم درس برنامه‌نویسی پیشرفته

نیم‌سال اول ۱۳۹۹-۱۴۰۰

۰. فایل مربوط به توضیحات نحوه ارسال تمرین‌ها را که در مدل 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 ورودی گرفتن و نمایش خروجی اهمیت بسیاری دارد. دقیقاً 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.

صفحه

فهرست سوالات

- سوال اول.....۳
- سوال دوم.....۶
- سوال سوم.....۶
- سوال چهارم.....۸
- سوال پنجم.....۱۱
- سوال ششم.....۱۱

سوال اول

در جلسات گذشته، شما در کلاس درس، با مفاهیم کلاس^۱ و شی^۲ 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؟

شی یک موجودیت فیزیکی، منطقی و مجرد است در حالی که کلاس منحصرًا یک موجودیت منطقی و مجرد است. در واقع کلاس یک نقشه یا الگو است که شی از روی آن ساخته می‌شود. در جدول زیر تفاوت‌ها گفته شده است، ذکر ۴ مورد از این تفاوت‌ها کفایت.

No.	آبجکت	کلاس
۱	آبجکت یک نمونه حقیقی (فیزیکی) ساخته شده از روی کلاس است.	کلاس یک نقشه یا الگو انتزاعی (template) است که آبجکت از روی آن ساخته می‌شود.
۲	آبجکت یک موجودیت حقیقی و فیزیکی مانند خودکار، موبایل، تخت و غیره ... می‌باشد.	کلاس گروهی از آبجکت‌های مشابه است.
۳	کلاس یک موجودیت فیزیکی و نمونه‌ی حقیقی از یک کلاس است.	کلاس یک موجودیت منطقی و انتزاعی است.
۴	آبجکت از طریق کلیدواژه ی new از روی یک کلاس ساخته شده و در حافظه ی heap قرار می‌گیرد مانند این دستور: <code>Student s1=new Student();</code>	کلاس را با استفاده از کلیدواژه ی class تعریف می‌کنند مانند این دستور: <code>class Student</code>
۵	ممکن است بر اساس نیاز برنامه تعداد زیادی آبجکت از روی یک کلاس ساخته شود.	کلاس یکبار اعلان می‌شود.
۶	یک آبجکت به هنگام ساخته شدن، بخشی از حافظه را به خود تخصیص می‌دهد.	زمانی که توسعه دهنده یک کلاس را تعریف می‌کند، این کلاس جایی را در حافظه اشغال نمی‌کند.
۷	روش‌های مختلفی برای ایجاد آبجکت در جاوا وجود دارد که یکی از آن‌ها، استفاده از کلیدواژه ی new می‌باشد. از جمله روش‌های دیگر می‌توان به استفاده از متد <code>newInstance()</code> ، <code>clone()</code> ، متد <code>factory</code> و <code>deserialization</code> اشاره کرد.	برای تعریف کلاس یک راه بیشتر وجود ندارد و آن هم استفاده از کلیدواژه ی class می‌باشد.

^۱ Class

^۲ Object

۲. سازنده^۳ کلاس یک تابع^۴ است، این تابع به چه شکل از باقی توابع کلاس تمیز داده می‌شود؟

در زیر چند مورد تفاوت سازنده با یک تابع نوشته شده است.

شماره	سازنده	تابع
۱	سازنده برای ساختن و مقداردهی به یک شی است.	تابع برای اجرای دستورالعمل‌های خاصی است.
۲	به وسیله سیستم و با کلیدواژه new صدا زده می‌شود.	با نوشتن و صدا زدن در کد فراخوانی می‌شود.
۳	Return type ندارد.	Return type دارد.
۴	نامش باید با نام کلاس یکی باشد.	نامش باید با نام کلاس متفاوت باشد.

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
    }
}
```

³ Constructor

⁴ Function

کد درست به شکل زیر است:

```
public class Library {  
    private String title; // Class field  
    public Library(String title){  
        this.title = title;  
    }  
}
```

ضمناً میتوانند، به جای تغییر نام سازنده به Library از تغییر نام فایل به Book استفاده کنند.

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^۵ ورودی تعریف کنیم، زبان جاوا نیز یک سازنده پیشفرض^۶، بدون آرگومان برای کلاس، در نظر می‌گیرد. (غلط، با نوشتن یک سازنده، جاوا، سازنده بدون آرگومان در نظر نمی‌گیرد.)

ب) سطح دسترسی سازنده کلاس، حتما باید از نوع `public` باشد. (غلط، لزومی ندارد که سطح دسترسی `public` باشد و برای مثال می‌تواند `private` باشد.)

پ) اعضای `protected` در یک کلاس، تنها برای اعضای همان کلاس قابل دسترسی هستند. (غلط، اعضای `protected` برای کلاس‌های فرزند نیز 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. (غلط، به عنوان متغیر `local` شناخته می‌شوند و فقط در `scope` خود قابل دسترسی هستند.)

سوال سوم

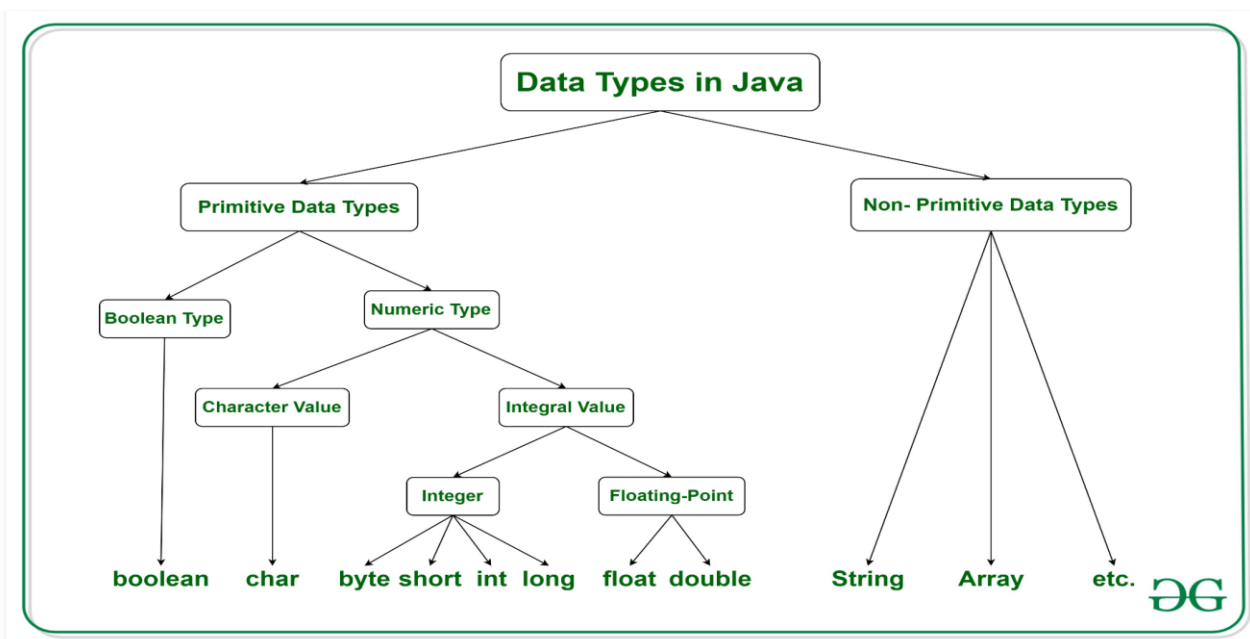
در برنامه‌نویسی با جاوا، شما با مفاهیمی از قبیل `Primitive types`، `Overloading`، `Getter` و `Setter` آشنا می‌شوید. در ادامه سوالاتی از این مفاهیم از شما پرسیده می‌شود.

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

شامل موارد زیر هستند:

^۵ Arguments

^۶ Default constructor



و دامنه آن‌ها موارد زیر است:

Primitive data are only single values and have no special capabilities.

TYPE	DESCRIPTION	DEFAULT	SIZE	EXAMPLE LITERALS	RANGE OF VALUES
boolean	true or false	false	1 bit	true, false	true, false
byte	twos complement integer	0	8 bits	(none)	-128 to 127
char	unicode character	\u0000	16 bits	'a', '\u0041', '\t01', '\', '\', '\n', '\', '\beta'	character representation of ASCII values 0 to 255
short	twos complement integer	0	16 bits	(none)	-32,768 to 32,767
int	twos complement integer	0	32 bits	-2, -1, 0, 1, 2	-2,147,483,648 to 2,147,483,647
long	twos complement integer	0	64 bits	-2L, -1L, 0L, 1L, 2L	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	IEEE 754 floating point	0.0	32 bits	1.23e100f, -1.23e-100f, .3f, 3.14F	upto 7 decimal digits
double	IEEE 754 floating point	0.0	64 bits	1.23456e300d, -1.23456e-300d, 1e1d	upto 16 decimal digits

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

overloading تابع، قابلیت است که به یک کلاس اجازه میدهد، تعداد دو یا بیشتر تابع با نام یکسان داشته باشد.

در **overload** یک تابع آرگومان های ورودی، دارای تفاوت های به شرح زیر هستند:

تعداد پارامترها.

نوع داده پارامترها.

ترتیب نوع داده پارامترها.

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

getter و setter دو روش متداول هستند که برای بازیابی و به روزرسانی مقدار یک متغیر استفاده می‌شوند.

کلاس variable را private اعلام می‌کند، از آنجا که متغیر private است، کدهای خارج از این کلاس مستقیماً به متغیر دسترسی پیدا نمی‌کنند،

در عوض، کد خارجی برای خواندن یا به روزرسانی متغیر باید getter، setVariable() و setter، setVariable() فراخوانی کند.

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) 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 مدل کردن 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 قبل 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 پیش فرض 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 در نظر بگیرد و در پایان، همه نمونه‌های `Item` ساخته شده 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 هر خط 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 از محصولات `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 صرفاً بر تعداد موجودی آن می‌افزاید.

- `RemoveProduct`

با گرفتن نام کالا و تعداد، در صورت موجود بودن در انبار، آن را از انبار کم و در صورت 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 نشان می‌دهد، برای مثال شرکتی که دارای دو خط تولید با نام‌های `LineName1` با محصول `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 می‌کند.

- `printAllStorageProducts`

تمامی کالا های موجود در انبار وظرفیت آن را نمایش می‌دهد به عنوان مثال انبار دارای محصولی به نام `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 صرفاً 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 در صورت 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.