به نام خدا

دستور کار کارگاه برنامه‌نویسی پیشرفته

نیم‌سال دوم ۹۷-۹۸

جلسه هفتم

آشنایی با تحلیل و طراحی نرم‌افزار و طراحی کلاس‌ها و مستند سازی

مقدمه

در جلسات قبلی شما یاد گرفتید که چگونه برای کلاس‌ها متد تعریف کنید، آنها را پیاده‌سازی کنید و از آنها در جاهای دیگر استفاده کنید. در این جلسات، کلاس ها بصورت طراحی شده به شما داده می‌شد و شما فقط نیازمند پیاده‌سازی متدهای آن بودید. در این جلسه قصد داریم تا کمی با تحلیل و طراحی شی‌گرایی آشنا شویم.

در تحلیل و طراحی شی‌گرا، توسعه‌گران تلاش می‌کنند تا فرایند(ها)ی که در دنیای واقعی می‌خواهند پیاده کنند را بصورت انتزاعی مدل کنند. مدلسازی به معنای ساده کردن موجودیت‌هایی است که قرار است در مساله‌ی ما با یکدیگر تعامل داشته باشند. موجودیت‌ها در دنیای واقعی دارای ویژگی‌های بسیاری هستند و معمولا به پیاده‌سازی تمام آنها در پروژه نیازی نیست (پیاده‌سازی آنها می‌تواند کار را بی دلیل پیچیده کند). به عنوان مثال در سیستم دانشگاه، دانشجو یک موجودیت به حساب می‌آید. این دانشجو ویژگی‌هایی مانند نام، نام خانوادگی، شماره دانشجویی، رنگ چشم، قد و علایق شخصی را دارد. برای طراحی سامانه انتخاب واحد دانشگاه، داشتن ویژگی‌های نام، نام خانوادگی و شماره دانشجویی دانشجو مهم است. در این سامانه پیاده‌سازی متدهایی که بیانگر قد و رنگ چشم دانشجو و علایق شخصی او هستند در اینجا بیهوده است و انجام آنها می‌تواند پروژه را بی دلیل دشوار و احتمال شکست آن را بالا ببرد. لذا فاز تحلیل و طراحی در پروژه‌های نرم افزاری اهمیت بسیاری دارد. در این جلسه سعی داریم تا یک مساله را خودمان مدلسازی کنیم و متدهای مربوط به هر کلاس را خود مشخص و تعریف کنیم.

بعد از طراحی کلاس‌ها و متدهای مربوط به آن‌ها باید با روش توسعه آزمون محور متدهای نوشته شده خود را آزمایش کنیم تا از عملکرد آن اطمینان خوبی پیدا کنیم. در صورتی که در فرایند آزمون به مشکلی برخوردیم فرایند اشکالیابی که در دو جلسه‌ی قبل راجع به آن صحبت شده بود را اجرا می‌کنیم تا اشکالزدایی را انجام دهیم.

جاواداک [1]

نکته: از این جلسه به بعد دانشجویان باید برای تمامی متدها و کلاس‌های خود، از جاواداک برای مستندسازی استفاده کنند.

از آنجایی که بسیاری از پروژه‌های نرم‌افزاری توسط تیم‌های چند نفره درست می‌شود، هر قسمت از برنامه توسط افراد مختلفی پیاده‌سازی می‌شود و بقیه اعضای تیم ممکن است از جزییات پیاده‌سازی قسمت‌های دیگر بی‌خبر باشند. همچنین ممکن است عضوی از تیم بعد از مدتی از تیم جدا شود و دیگر در دسترس نباشد. در صورتی که نیاز باشد از کدهای این اشخاص استفاده شود یا تغییراتی در آنها ایجاد شود نیاز است تا این کدها مستندسازی شده باشند. مستندسازی به برنامه نویسان کمک می‌کند تا بدون خواندن کامل کدی که میخواهند از آن استفاده کنند، جزییات مورد نیاز برای استفاده کردن از آن کد را بدست بیاورند.

جاواداک یکی از ابزارهای تولیدکننده مستند است که از آن برای توصیف کلاس‌ها و متدها استفاده می‌شود. در درس برنامه‌نویسی مقدماتی و پیشرفته با comment ها آشنا شدید که قسمتی از کد هستند که فقط نقش توصیف کد را دارند و به زبان ساده و قابل فهم نوشته می‌شوند. جاواداک نوع ساخت یافته ای از comment ها هستند که قبل از نام متدها و کلاس‌ها نوشته می‌شوند. این کامنت‌ها توسط ابزار جاواداک قابل فهم است و میتوان از آنها خروجی html نیز گرفت.

برای مثال در قطعه کد زیر یک توضیحی برای کلاس HelloStudents نوشته شده. در ابتدا یک خط درباره خود کلاس و کاربرد آن نوشته شده. سپس با استفاده از یک سری tag های از پیش تعیین شده به توصیف بقیه ویژگی‌های کلاس می‌پردازد. به عنوان مثلا تگ @author نویسنده‌ی کلاس را توصیف می‌کند. به نظر شما علت نوشتن این تگ چه می‌تواند باشد؟

```java
/**
* HelloStudents simply displays "Hello, AP students!" on the standard output.
* @author Amir Kalbasi
* @version 1.0
* @since 2018-10-17
*/
public class HelloStudents {
	public static void main(String[] args) {
		// Prints Hello, AP students! On the standard output.
		System.out.println("Hello, AP students!");
	}
```

[1] Javadoc

```
}
```

لیست کامل تگ‌های جاوا در لینک زیر قابل مشاهده است:

https://www.tutorialspoint.com/java/java_documentation.htm

یک جمله خلاصه که توضیحی برای متد و کلاس است باید برای هر کدام از متدها و کلاس‌ها نوشته شود.

برای کلاس‌ها از تگ‌های مختلفی من جمله @author, @version, @since می توان استفاده کرد.

قطعه کد زیر نیز جاواداک را برای کلاس و متدهای آن نشان می‌دهد.

```
import java.util.ArrayList;
/**
* The Stack class represents a last-in-first-out stack of objects.
*
* @author Joseph Bergin
* @version 1.0, May 2000 Note that this version is not thread safe.
*/
public class Stack{
	/**
	* Pushes an item on to the top of this stack.
	* @param item the item to be pushed.
	*/
	public void push(Object item) {
		this.elements.add(item);
	}
	...
	/**
	* Tests if this stack is empty.
	*
	* @return true if this stack is empty and false otherwise.
	*/
	public boolean isEmpty() {
		return this.elements.isEmpty();
	}
}
```

برای متدها تگ‌های @param, @return, @throws در صورت وجود نیاز است که نوشته شود.برای توضیح هرکدام از تگ‌ها به جدول داده شده در لینک مراجعه کنید.

انجام دهید

فایل Lab6.zip را که برای جلسه‌ی پیش بود را دانلود کنید. سپس برای خود کلاس و متدهای کلاس Course مستندات مناسب بنویسید.

تحلیل و طراحی

شروع فرایند تحلیل و طراحی با توصیف پروژه شروع می‌شود. در اینجا ما سعی داریم تا یک سامانه فروش بلیط هواپیما را پیاده‌سازی کنیم. توصیف سامانه معمولا توسط افرادی انجام می‌شود که مشتریان آن سامانه هستند. در اصطلاح به این افراد سودبران[۲] سامانه می‌گویند. توصیف سامانه رزرو بلیط هواپیما بصورت زیر است:

سامانه مدیریت بلیط باید لیست پروازهای فرودگاه را در اختیار مشتریان قراردهد و همچنین می‌تواند پروازی را به لیست پروازها اضافه کند یا پروازی را از لیست پروازها حذف کند. اطلاعات پرواز عبارت است از: روز و ساعت پرواز، مبدا و مقصد پرواز، شرکت هواپیمایی، نوع‌هواپیما و شماره پرواز. سپس کاربر می‌تواند یکی از پروازها را انتخاب کند و لیست کل صندلی‌ها و صندلی‌های خالی هواپیما را مشاهده کند. ویژگی‌های هر صندلی عبارت است از: شماره صندلی، نوع صندلی (business class ، first class یا economic class). سپس کاربر می‌تواند صندلی مورد نظر خود را انتخاب کند و آنرا رزرو کند. کاربر سامانه دارای ویژگی‌های نام، نام خانوادگی و ID است. ویژگی‌های بلیطی که کاربر رزرو می‌کند عبارت است از: شماره صندلی، شماره پرواز، مقدار باری که مسافر با خود می‌تواند به همراه داشته‌باشد و اینکه آیا مسافر مایل به دریافت غذا است یا خیر. بعد از آنکه مشتری توانست بلیط را رزرو کند باید توانایی مشاهده بلیط‌های رزرو کرده خود را نیز داشته باشد.

گام اول در طراحی شی‌گرایی استخراج موجودیت‌ها و کلاس‌های سامانه است. بعضی از این موجودیت‌ها می توانند کلاس‌های پروژه ما بشوند. همانطور که در قسمت قبلی اشاره کردیم، موجودیت‌ها در مساله دارای ویژگی‌هایی هستند. برای استخراج موجودیت‌ها باید به تعریف مساله رجوع کرد و واژگانی که اسم هستند را به عنوان موجودیت در نظر گرفت.برای مثال در این مساله پرواز یک نوع موجودیت است که با رنگ قرمز مشخص شده. ویژگی‌های این کلاس نیز مشخص شده که زیر آن‌ها خط کشیده شده است.

سامانه مدیریت بلیط باید لیست پروازهای فرودگاه را در اختیار مشتریان قراردهد. اطلاعات پرواز عبارت است از: روز و ساعت پرواز، مقصد پرواز، شرکت هواپیمایی نوع‌هواپیما و شماره پرواز.

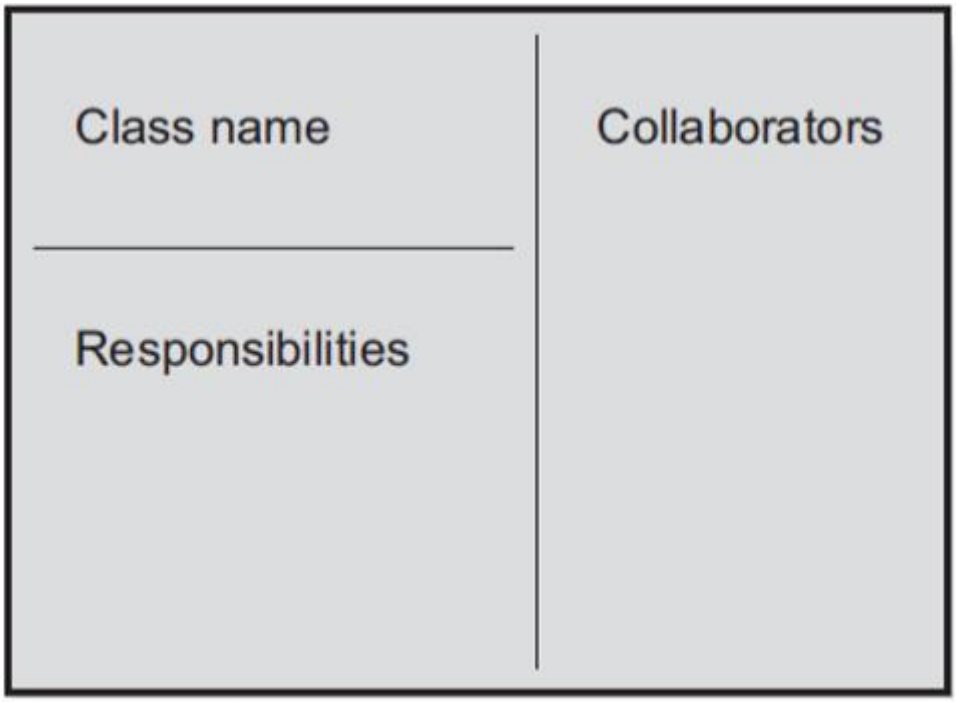
[۲] Stakeholder

انجام دهید

به کمک مدرس کارگاه لیستی از بقیه موجودیت‌های سامانه بالا را درست کنید.

توصیف مسولیت‌ها

بعد از تشخیص کلاس‌ها، نوبت به مسولیت‌های هر کلاس می‌رسد. برای انجام این کار از نمودار CRC استفاده می‌کنیم. این نمودار به ما کمک می‌کند تا بتوانیم یک شمای کلی از کلاس و مسولیت‌های آن و کلاس‌های مرتبط با آن داشته باشیم. اجزای نمودار CRC به صورت زیر است:



برای هر کلاس، یک کارت CRC لازم است. اجزای این شکل هر کدام قسمت‌هایی از کلاس را نمایش می‌دهند که عبارتند از:

- Class name: نام کلاس در این قسمت نوشته می‌شود.
- Collaborators: کلاس‌های دیگری که برای پیاده‌سازی لازم بوده و با این کلاس در ارتباط هستند.
- Responsibilities: لیست وظایف کلاس است که باید در قالب متدها پیاده‌سازی شود.

در تحلیل و طراحی نرم‌افزار بحث طراحی متدها است. ابتدا باید روی کاغذ لیست متدهایی که قرار است برای هر کلاس نوشته شود استخراج شود. به عنوان مثال برای کلاس پرواز برای

Class name می‌توان نام Flight را انتخاب کرد. بعد از انتخاب نام باید وظایف آنرا نوشت. همانطور که در قسمت قبلی اشاره شد وظایف کلاس پرواز عبارت است از:

- نشان دادن ساعت و روز پرواز
- نشان دادن مبدا و مقصد پرواز
- نشان دادن شهر مقصد
- نشان دادن شرکت هواپیمایی
- نشان دادن نوع هواپیما و شماره پرواز

سپس از روی وظایف کلاس، کلاس‌های مرتبط با آن کلاس را یادداشت می‌کنید. به عنوان مثال کلاس پرواز با کلاس هواپیما ارتباط دارد چرا که یکی از وظایف آن نشان دادن هواپیمایی است که قرار است در ساعت پرواز به شهر مقصد پرواز سفر کند.

انجام دهید

به کمک مدرس کارگاه برای ۳ کلاس دیگر این سامانه نمودار CRC آنها را رسم کنید و آنرا به مدرس کارگاه گزارش دهید.

پیاده‌سازی

بعد از آنکه متدها را طراحی کردیم باید کلاس‌ها را پیاده کنیم. برای پیاده سازی کلاس‌ها از روش توسعه آزمون محور استفاده می‌کنیم. برای این کار باید ابتدا متدهای کلاس‌ها را استخراج کنیم و بقیه قسمت‌ها همانند جلسه قبل است. منظور از متدهای کلاس متدهای public هستند که رابط کلاس با سایر کلاس‌ها را مشخص می‌کند. متدهای private معمولا هنگام نوشتن کد برای جلوگیری از تکرار کد و افزایش خوانایی آن استفاده می‌شوند. استخراج متد بدین صورت است که باید هریک از وظایفی که در قسمت قبلی در نمودار CRC بدست آمده را به یک یا چند متد نگاشت کرد. به عنوان مثال برای کلاس پرواز برای پیاده سازی نشان دادن ساعت و روز پرواز می‌توان از متد زیر استفاده کرد:

```
public Date getDateAndTime();
```

برای نشان دادن هواپیما:

```
public Airplane getAirplane();
```

نشان دادن مقصد پرواز:

```
public String getDestination();
```

نکته: از این به بعد برای نشان دادن از System.out.println استفاده نکنید. برای آنکه چیزی را چاپ کنید، ابتدا آنرا بصورت String بازگردانید و در کلاس کاربر خود آنرا چاپ کنید.

انجام دهید

به کمک مدرس کارگاه برای سه کلاسی که در قسمت قبلی طراحی کردید متدهای مناسب بنویسید. سپس برای آن‌ها تست کیس‌های مناسب بنویسید. در نهایت نیز آن‌ها را پیاده‌سازی کنید تا زمانی که تست کیس‌ها پاس شود.