به نام خدا

دستور کار کارگاه برنامه‌نویسی پیشرفته

نیم‌سال اول 97-98

جلسه چهارم

کار با کتابخانه‌های جاوا

مقدمه

جلسه گذشته با مفهوم modularization و abstraction آشنا شدید و در دستورکار با نوشتن ماژول‌های مختلف سطح دسترسی متفاوتی به کلاس‌های داخل آن ماژول دادید. در این جلسه قصد داریم تا با مفهوم کتابخانه آشنا بشویم و یک کتابخانه را به پروژه خود اضافه کنیم.

کتابخانه (library)

همانطور که در جلسه قبلی بیان کردیم، ساده‌سازی پیاده‌سازی نرم‌افزار را abstraction می‌گویند. در اول درس با ایجاد class توانستیم یک سری وظایف را در قالب متدهای public تعریف کنیم. سپس با تعریف package توانستیم تعدادی کلاس را در کنار یکدیگر قرار دهیم که هدف خاصی را دارند ولی نیازی به در دسترس بودن تمامی کلاس‌های آن پکیج نیست.

حال اگر یک روز شما تصمیم بگیرید که یک کدی در ابعاد بزرگ بنویسد که باید آنرا برای استفاده به سایر اعضای تیم بدهید، باید از سطح بالاتری از abstraction نسبت به class و package استفاده کنید. به عنوان مثال اگر تیم شما بخواهد یک سامانه حضور و غیاب با استفاده از دستگاه اثر انگشت بنویسد، آنگاه باید کلاس‌ها و توابعی نوشته شود که وظیفه‌ی ارتباط با سخت‌افزار را بر عهده داشته باشد. اگر این قسمت بر عهده شما باشد، باید یک package درست کنید که شامل یک سری کلاس باشد که باید با سخت‌افزار ارتباط برقرار کند و داده‌های آنرا از روی حافظه دستگاه بخواند یا بنویسد و کلاس‌های دیگری که سایر توسعه‌دهندگان از آنها استفاده کنند تا بتوانند داده‌های خود را به سخت افزار انتقال دهند. کلاس‌های دسته اول باید از دید توسعه‌دهندگان مخفی بماند و آنها فقط باید کلاس‌های دسته دوم را ببینند. برای این کار از یک package استفاده می‌کنید. حال فرض کنید این سخت‌افزار یک نمایشگر دارد که باید نام کسی که انگشت خود را گذاشته روی آن نمایش داده شود. برای این کار شما باید یک package جدید برای ارتباط با نمایشگر نیز ایجاد کنید. اگر این سخت‌افزار شما ویژگی‌های دیگری نیز داشته باشد، برای هرکدام شما باید یک package درست کنید. معمولا اینگونه است که برای توسعه چنین محصولاتی از تعداد زیادی package و کلاس استفاده می‌شود. در این صورت پیشنهاد می‌شود از کتابخانه استفاده شود.

انجام دهید

در این جلسه قصد داریم از کتابخانه‌ای استفاده کنیم که قابلیت کار با فایل‌های PDF را به برنامه‌نویسان جاوا می‌دهد. برای آنکه درکی از تعداد پکیج‌ها و کلاس‌های این کتابخانه داشته باشید ابتدا در لینک زیر یک نگاهی به کلاس‌ها و package های این کتابخانه بیاندازید.

http://www.java2s.com/Code/Jar/i/Downloaditext505jar.htm

در صورتی که بخواهید این تعداد کلاس و فایل را همانند جلسه قبلی در پوشه پروژه خود copy/paste کنید، آنگاه کدهای شما در میان کدهای نوشته شده دیگر گم خواهد شد. همچنین سایر برنامه‌نویس‌ها نیز قادر به شناسایی کد شما به سادگی نخواهند بود.

حال میخواهیم از این کتابخانه استفاده کنیم و یک فایل PDF بسازیم. برای این کار ابتدا فایل کتابخانه را از لینک زیر دانلود کنید.

http://central.maven.org/maven2/com/itextpdf/itextpdf/5.0.6/itextpdf-5.0.6.jar

لینک زیر نحوه‌ی اضافه کردن کتابخانه به Intellij را به شما نشان می‌دهد.

https://stackoverflow.com/questions/1051640/correct-way-to-add-external-jars-lib-jar-to-an-intellij-idea-project

حال میخواهیم یک برنامه کوچک بنویسیم که یک فایل PDF را بخواند و صفحه اول آنرا چاپ کند. برای این کار ابتدا فایل test.pdf را از moodle دانلود کنید و مسیر کامل آنرا کپی کنید. سپس یک package درست کنید و کلاس PDFReaderTest را در آن بسازید. در تابع main این کلاس قطعه کد زیر را وارد کنید.

```
try {
        PdfReader pdfReader = new PdfReader("test.pdf");
} catch (Exception e){
        e.printStackTrace();
}
```

مشاهده خواهید کرد که در ابتدا جاوا این کلاس را نمیشناسد و عبارت PdfReader را قرمز می‌کند. برای آنکه مشکل رفع شود باید پکیجی را که این کلاس در آن تعریف شده را import کنید. برای این کار عبارت زیر را در اول فایل و قبل از تعریف کلاس‌ها وارد کنید.

```
import com.itextpdf.text.pdf.PdfReader;
```

مشاهده خواهید کرد که خطای پروژه رفع شده. این عبارت کلاس PdfReader را از پکیج com.itextpdf.text.pdf به پروژه شما اضافه کرده. سپس قطعه کد زیر را در ادامه اضافه کرده و نتیجه را مشاهده کنید.

```
int pages = pdfReader.getNumberOfPages();
for (int i=1; i<=pages; i++) {
        String pageContent = PdfTextExtractor.getTextFromPage(pdfReader, i);
        System.out.println("Content on Page " + i + ": " + pageContent);
}
```

تغییرات خود را commit کنید.

حال می‌خواهیم با استفاده از این کتابخانه یک فایل PDF را بسازیم. کلاس دیگری به نام PDFWriteTest ساخته. نحوه‌ی نوشتن فایل pdf را از لینک زیر بخوانید:

https://codesjava.com/itext-create-pdf-file-in-java

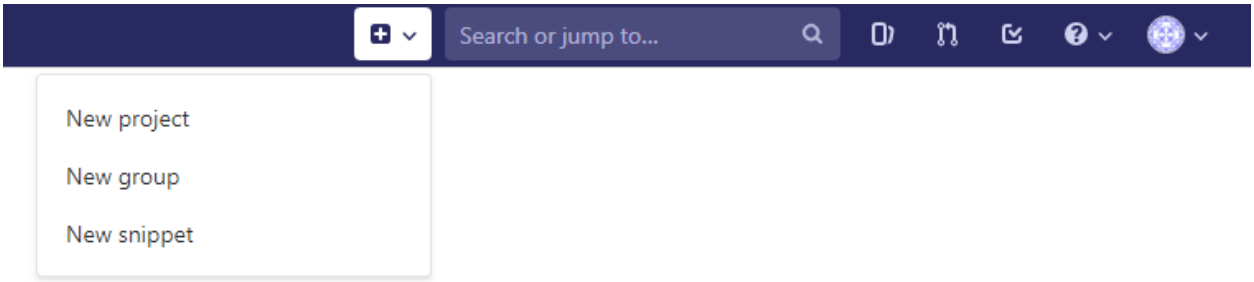
تغییرات خود را commit کنید.

Git Repository

در جلسه دوم با ابزار git و نحوه‌ی ذخیره‌ی تغییرات پروژه آشنا شدید. مدیریت تغییرات پروژه‌ی شما در پوشه‌ای به نام git. ذخیره شده است که این پوشه فقط در کامپیوتر شما قابل دسترسی است. git این قابلیت را به کاربران خود می‌دهد تا بتوانند کدهای خود را در جاهای دیگر نیز upload کنند. این کار علاوه بر آنکه باعث شده تا کدها به همراه تمامی commit ها و log ها و ... در جایی سالم بمانند، قابلیت کارگروهی را نیز به کاربران git می‌دهد (در جلسات آینده درباره این موضوع صحبت خواهیم کرد). برای این کار نرم‌افزار git قابلیت ساخت git repository را به کاربران می‌دهد که همانند فضای آپلود فایل است. در این فضا شما می‌توانید پروژه‌های git خود را آپلود کنید و در صورت نیاز در اختیار دیگران قرار دهید. دانشکده کامپیوتر نیز از ترم پیش یک git repository ایجاد کرده که دانشجویان می‌توانند در آنجا کدهای خود را آپلود کنند. برای این کار به آدرس زیر بروید:

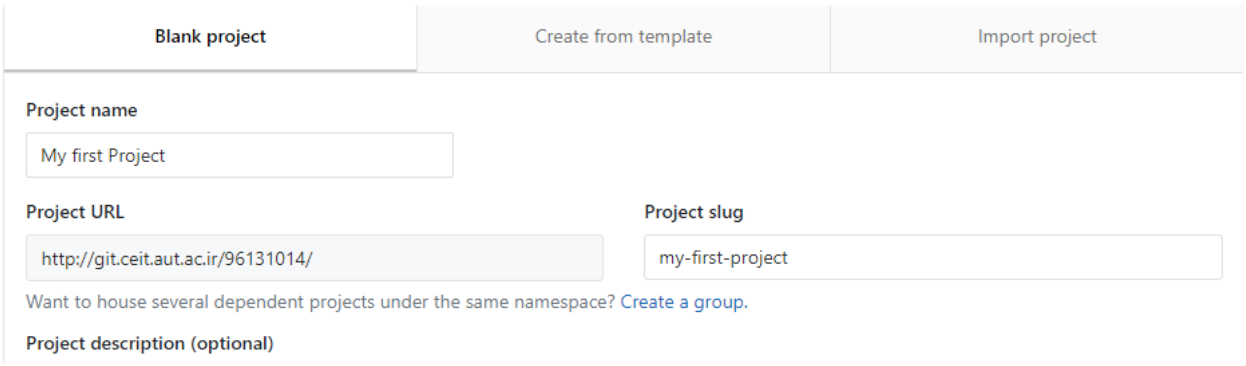
https://git.ceit.aut.ac.ir/

با اکانتی که وارد moodle می‌شوید (CEIT username همان شماره دانشجویی است و password رمز عبود برای وارد شدن به moodle) در این سایت login کنید و به فضای کاربری خود بروید. در منوی بالا گزینه ی "+" را زده و new project را انتخاب کنید.



شکل 1 منوی بالای gitlab

سپس مطابق شکل 2 یک عنوان برای پروژه خود انتخاب کنید.



شکل 2 ساخت پروژه در gitlab

سپس بر روی create project کلیک کنید.

در محیط git bash ابتدا مشخصات کاربری خود را بصورت global تنظیم کنید.

```
config --global user.name "your-student-id"
git config --global user.email "your-student-id@ceit.local"
```

حال آدرس repository خود را با دستور زیر به git اضافه کنید.

شکل 3 url پروژه git

مطابق شکل بالا آدرس پروژه‌ی خود را کپی کنید و دستور زیر را وارد کنید.

نکته : بجای پروتکل http از پروتکل https استفاده کنید.

```
git remote add origin <project-url>
```

بعد از اضافه شدن url پروژه باید commit های خود را با دستور زیر push کنید.

```
git push -u origin master
```