به نام خدا

دستور کار کارگاه برنامه‌نویسی پیشرفته

جلسه چهارم

آشنایی با مستندسازی، کتابخانه‌ها و مدل حافظه در جاوا

مقدمه

در این جلسه قصد داریم با تعدادی از کتابخانه‌های پرکاربرد جاوا و پیداکردن روش استفاده از آن‌ها با استفاده از مستندات جاوا آشنا شویم. پس از آن با بعضی از ساختمان داده‌های موجود در این زبان آشنا می‌شویم و نکاتی را پیرامون مدل حافظه در جاوا مطرح می‌کنیم.

نکات آموزشی

۱. آشنایی با مستندات جاوا (JavaDoc) و نحوه تولید آن‌ها
۲. آشنایی با تکمیل خودکار در محیط توسعه یکپارچه
۳. آشنایی با روش استفاده از کتابخانه‌ها در جاوا
۴. آشنایی با تعدادی از کتابخانه‌های معروف و پرکاربرد در جاوا
۵. نکاتی پیرامون مدل حافظه در جاوا

آشنایی با مستندات جاوا

زبان برنامه‌نویسی جاوا معمولا برای پروژه‌هایی با ابعاد بزرگ استفاده می‌شود که به وسیله‌ی تیم‌های برنامه‌نویسی توسعه می‌یابند. از این رو لازم است روشی برای انتقال اطلاعات و نحوه‌ی استفاده از کلاس‌ها و متدهای نوشته‌شده توسط هر برنامه‌نویس به دیگران وجود داشته باشد. این عمل توسط مستندسازی کدها انجام می‌شود. از طرفی، هنگامی که برنامه‌نویسان برنامه‌های خود را به صورت کتابخانه در اختیار دیگران قرار می‌دهند، لازم است چگونگی فراخوانی توابع و متدهای استفاده‌شده در آن برای استفاده‌کنندگان به نحوی مشخص شود که بدون نیاز به اطلاع از جزئیات و نحوه پیاده‌سازی، بتوان به سادگی از آن‌ها در کاربردهای مختلف استفاده کرد. یکی از اهداف دستور این جلسه آشنایی با روش استفاده از این مستندات و تولید آن‌ها برای برنامه‌هایی است که در این درس پیاده‌سازی می‌شوند.

کتابخانه‌های جاوا همراه با یک فایل مستند ارائه می‌شوند که در آن روش استفاده از کلاس‌های موجود در کتابخانه، توضیح واسط (interface)های موجود، روش فراخوانی متدها، ورودی و خروجی هر متد و شرح کلی عملکرد مربوط به آن توضیح داده شده است. این مستندات برای کتابخانه‌های معروف جاوا در اینترنت موجود است و در سایت‌هایی مانند: https://www.oracle.com[1] و https://www.tutorialspoint.com/java/ یافت می‌شود (نسخه‌ای از مستندات رسمی موجود در سایت اوراکل از http://ceit.aut.ac.ir/~ghaffarian/files/jdk-8u161-docs-all.zip قابل دانلود است.

یکی از مهم‌ترین ابزارهای نگارش مستند در جاوا، JavaDoc است. این ابزار که در JDK موجود است، برای ساخت مستند کاربرد دارد. روش استفاده از این ابزار به این صورت است که ابتدا در کد خود با استفاده از یک دستور زبان[2] خاص توضیحات را وارد کرده، سپس با اجرای JavaDoc مستندات را در قالب یک فایل html تولید می‌کنید. این دستورات به صورت کامنت لابلای کد نوشته می‌شوند و توسط کامپایلر بررسی نمی‌شوند. نوع سوم از کامنت‌گذاری که در جدول ۱ آماده است، برای نوشتن این دستورات به کار می‌رود.

جدول ۱ - انواع کامنت گذاری در جاوا

Sr.No.	Comment & Description
1	**/* text */** The compiler ignores everything from /* to */.
2	**//text** The compiler ignores everything from // to the end of the line.
3	**/** documentation */** This is a documentation comment and in general it's called **doc comment**. The **JDK javadoc** tool uses *doc comments* when preparing automatically generated documentation.

[1] منبع رسمی زبان برنامه نویسی جاوا این سایت است ولی به دلیل تحریم‌ها فعلا امکان دسترسی به آن‌ها از آدرس‌های ایران نیست.
[2] Grammar

این دستورات قبل از قطعه کدی که قصد توضیح آن را داریم (مثال: قبل از تعریف کلاس، قبل از تعریف متدها و فیلدها) نوشته می‌شود. برای نمونه، اگر قصد توضیح عملکرد یک کلاس خاص را داریم، در بالای کد (مانند کد ۱) توضیحات را می‌نویسیم.

```java
/**
 * The HelloWorld program implements an application that
 * simply displays "Hello World!" to the standard output.
 *
 * @author  Sepehr Sabour
 * @version 1.0
 * @since   2018-01-15
 */
public class HelloWorld {

  public static void main(String[] args) {
      // Prints Hello, World! on standard output.
      System.out.println("Hello World!");
   }
}
```

کد ۱- نمونه یک متد مستندسازی شده

همچنین می توانید با استفاده از تگ‌های مخصوص زبان html نیز توضیحات خود را تکمیل کنید. در این صورت می‌توانید قالب فایل تولیدشده را نیز بهبود دهید. نمونه‌ای از یک فایل مستندات مربوط به کلاس Scanner را در فایل پیوست‌شده مشاهده کنید.

تگ‌ها در مستندات برای بیان آن‌چه از پیش تعریف شده است، استفاده می‌شوند و روش یکسانی را برای معرفی این موارد فراهم می‌آورند. برای مثال، تگ @param جهت معرفی پارامترهای یک متد است که اصولا بیان آن در مستندسازی هر متد الزامی است. در جدول ۲ تگ‌های شناخته‌شده برای تولید مستند همراه با توضیحات و کاربرد هر کدام آورده شده است.

جدول ۲-تگ‌های استفاده شده در JavaDoc

Tag	Description	Syntax
@author	Adds the author of a class.	@author name-text
{@code}	Displays text in code font without interpreting the text as HTML markup or nested javadoc tags.	{@code text}

{@docRoot}	Represents the relative path to the generated document's root directory from any generated page.	{@docRoot}
@deprecated	Adds a comment indicating that this API should no longer be used.	@deprecated deprecatedtext
@exception	Adds a **Throws** subheading to the generated documentation, with the classname and description text.	@exception class-name description
{@inheritDoc}	Inherits a comment from the **nearest**inheritable class or implementable interface.	Inherits a comment from the immediate surperclass.
{@link}	Inserts an in-line link with the visible text label that points to the documentation for the specified package, class, or member name of a referenced class.	{@link package.class#member label}
{@linkplain}	Identical to {@link}, except the link's label is displayed in plain text than code font.	{@linkplain package.class#member label}
@param	Adds a parameter with the specified parameter-name followed by the specified description to the "Parameters" section.	@param parameter-name description
@return	Adds a "Returns" section with the description text.	@return description
@see	Adds a "See Also" heading with a link or text entry that points to reference.	@see reference
@serial	Used in the doc comment for a default serializable field.	@serial field-description | include | exclude
@serialData	Documents the data written by the writeObject( ) or writeExternal( ) methods.	@serialData data-description

@serialField	Documents an ObjectStreamField component.	@serialField field-name field-type field-description
@since	Adds a "Since" heading with the specified since-text to the generated documentation.	@since release
@throws	The @throws and @exception tags are synonyms.	@throws class-name description
{@value}	When {@value} is used in the doc comment of a static field, it displays the value of that constant.	{@value package.class#field}
@version	Adds a "Version" subheading with the specified version-text to the generated docs when the -version option is used.	@version version-text

پس از نوشتن مستندات و گذاشتن تگ‌های مورد نظر، با استفاده از دستور javadoc مستند مربوطه را تولید می‌کنیم.

> javadoc <MySourceFileName> .java -d <Destination Directory>

> javadoc -sourcepath <Source Directory> -d <Destination Directory>

روش ساده‌تر استفاده از IDE برای تولید JavaDoc است. در IntelliJ IDEA، از منوی Tools، با انتخاب گزینه Generate JavaDoc می‌توان به سادگی فایل html مستندات پروژه را ایجاد کرد.

نمونه‌ای از مستند تولیدشده برای کد ۱ را در پیوست دستور کار قابل مشاهده است.

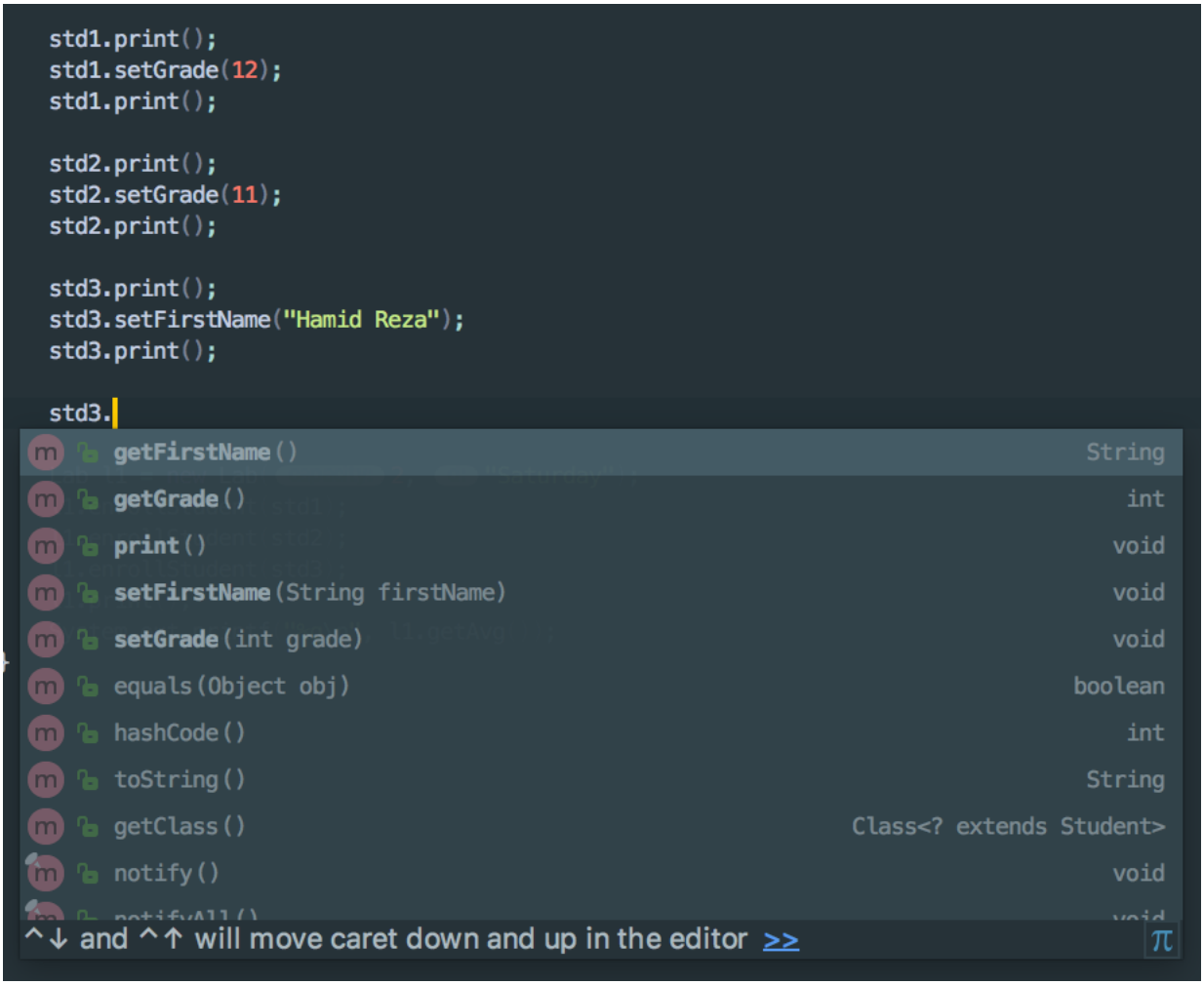
```java
import java.io.*;

/**
 * <h1>Add Two Numbers!</h1>
 * The AddNum program implements an application that
 * simply adds two given integer numbers and Prints
 * the output on the screen.
 * <p>
 * <b>Note:</b> Giving proper comments in your program makes it more
 * user friendly and it is assumed as a high quality code.
 *
 * @author  Sabour Sepehr
 * @version 1.0
 * @since   2018-01-17
 */
public class AddNum {
  /**
   * This method is used to add two integers. This is
   * a the simplest form of a class method, just to
   * show the usage of various javadoc Tags.
   * @param numA This is the first paramter to addNum method
   * @param numB  This is the second parameter to addNum method
   * @return int This returns sum of numA and numB.
   */
  public int addNum(int numA, int numB) {
    return numA + numB;
  }

  /**
   * This is the main method which makes use of addNum method.
   * @param args Unused.
   * @exception IOException On input error.
   * @see IOException
   */

  public static void main(String args[]) throws IOException {
    AddNum obj = new AddNum();
    int sum = obj.addNum(10, 20);

    System.out.println("Sum of 10 and 20 is :" + sum);
  }
}
```

کد ۱ - نمونه یک کلاس مستندسازی شده

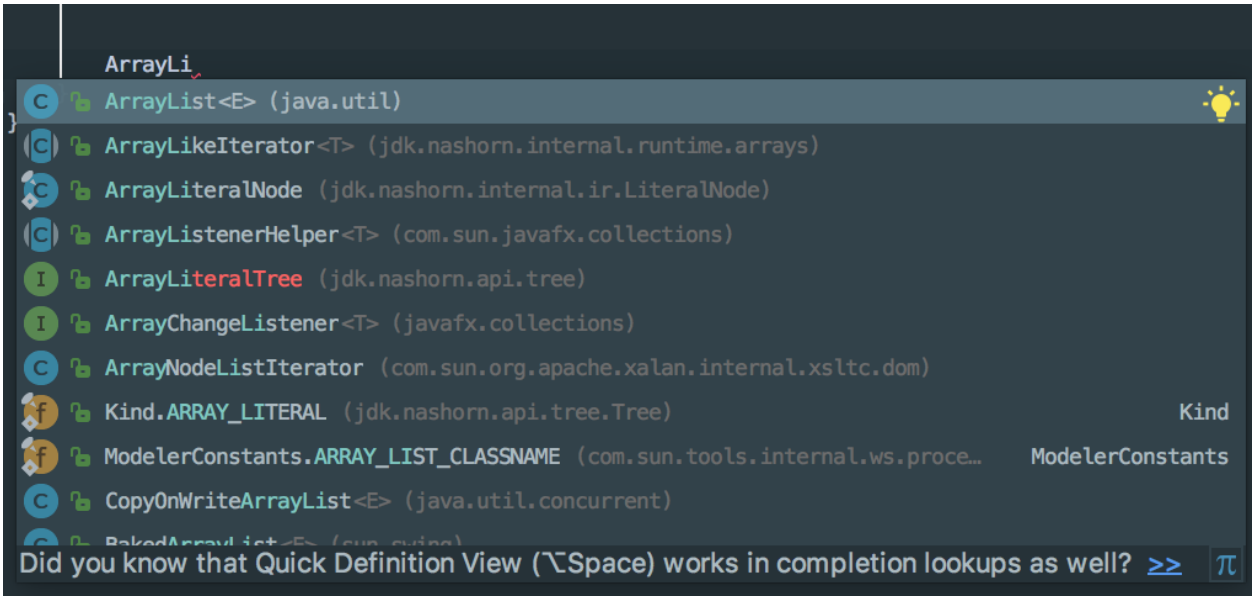
انجام دهید: برای تمرین بیشتر روش مستندسازی در جاوا، تمرین این جلسه را توسط روش گفته شده مستندسازی کنید. لازم به یادآوری است که مستندسازی تمرین‌ها و پروژه‌های این درس بعد از این جلسه الزامی است.

آشنایی با تکمیل خودکار در محیط توسعه یکپارچه

یکی از ویژگی‌هایی که IDEهای معروف مانند IntelliJ در اختیار توسعه‌دهندگان قرار می‌دهد، تکمیل خودکار کد است؛ به این ترتیب که شما بخشی از متد، فیلد، نوع و موارد دیگری را که می‌خواهید، نوشته و با فشردن دکمه‌های Ctrl + Space، IDE پیشنهادهایی برای تکمیل آن به شما ارائه می‌کند (پیشتر shortcutهای کاربردی محیط IntelliJ در مودل قرار گرفته است).



شکل ۱ – پیشنهادهای ارائه‌شده توسط IDE برای تکمیل کد



شکل ۲- پیشنهادهای ارائه‌شده توسط IDE برای تکمیل کد

با استفاده از این ویژگی، شما می‌توانید اطلاعاتی از کتابخانه‌هایی که برای شما آشنا نیستند، به دست آورید و مدت زمان نوشتن کد را کاهش دهید. در صورتی که شما نمی‌دانید چگونه می‌توانید با HashMap کار کنید، اگر یک instance از آن بسازید، با فشردن Ctrl + Space می‌توانید متدهایی که بیشتر کاربرد دارند را مشاهده کنید و به این ترتیب از متد مورد نظر استفاده کنید.

شکل ۳ - پیشنهادهای ارائه‌شده توسط IDE برای تکمیل کد

آشنایی با روش استفاده از کتابخانه‌های در جاوا

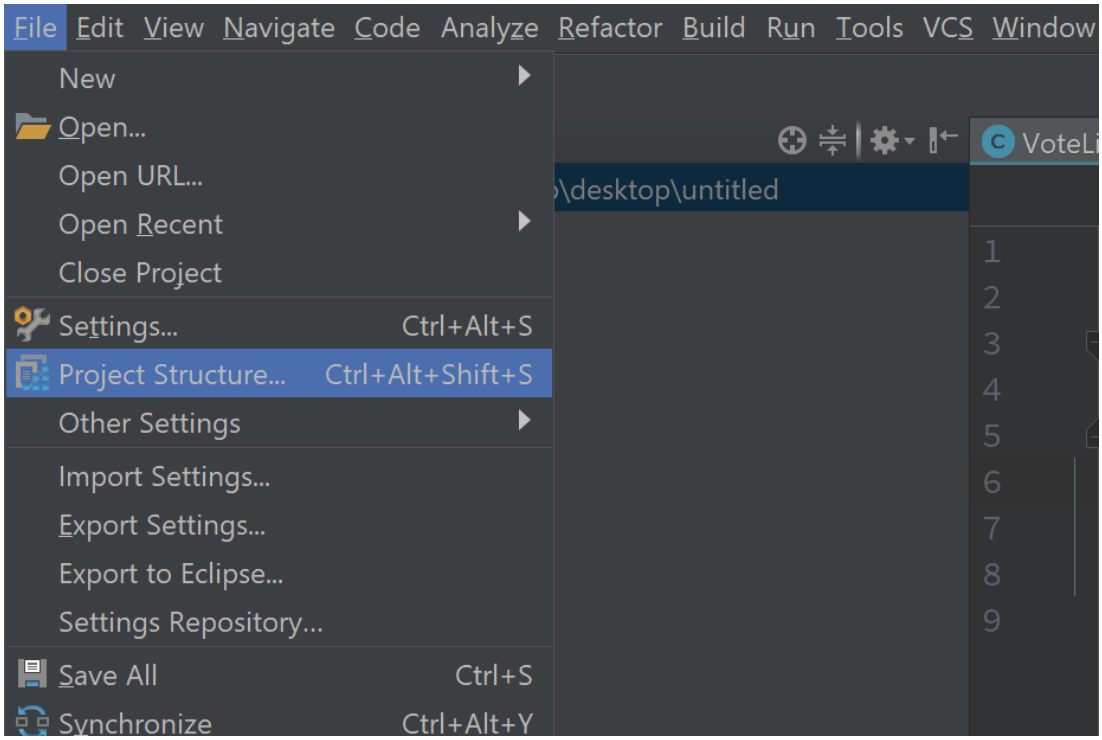
برنامه‌نویسان جاوا برای افزایش کارایی و کاهش زمان توسعه برنامه‌های خود نیاز دارند که از کدهای آماده و نوشته‌شده توسط دیگران استفاده کنند. این کدهای آماده اغلب به صورت کتابخانه موجود است و این کتابخانه‌ها معمولا به صورت فایل با پسوند .jar در اختیار آن‌ها قرار می‌گیرد که یا به طور مستقیم دانلود شده و به پروژه اضافه می‌گردد و یا توسط ابزارهایی مانند Maven و Gradle (که به این ابزارها اصطلاحا سیستم‌های خودکارسازی ساخت[3] می‌گویند) در دسترس قرار می‌گیرند. در این دستور کار افزودن مستقیم یک کتابخانه شرح داده می‌شود و سایر ابزارها در جلسات آینده بررسی می‌شوند.

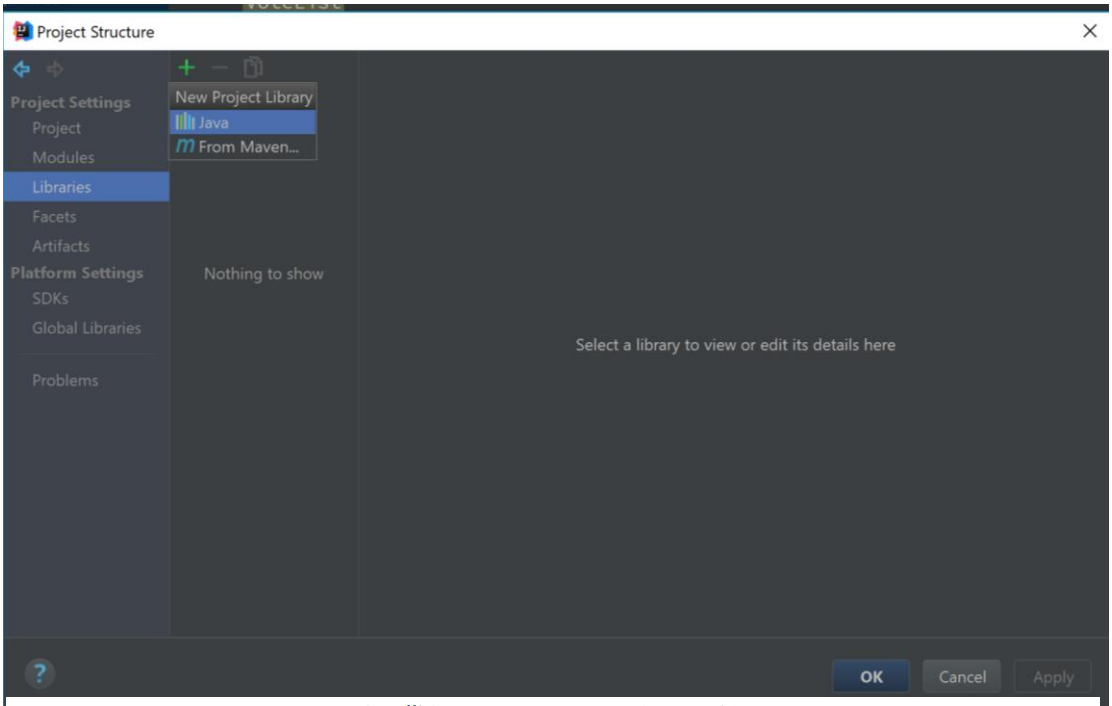
یکی از راه‌های افزودن فایل jar. توسط IDEها است. افزودن این نوع فایل توسط Intellij در شکل‌های ۴ تا ۶ آورده شده است. از جمله مجموعه کتابخانه‌های معروف و پرکاربرد موجود می‌توان به Google Guava و Apache Commons اشاره کرد.

انجام دهید: کتابخانه‌های موجود در این دو مجموعه را جستجو و بررسی کنید.

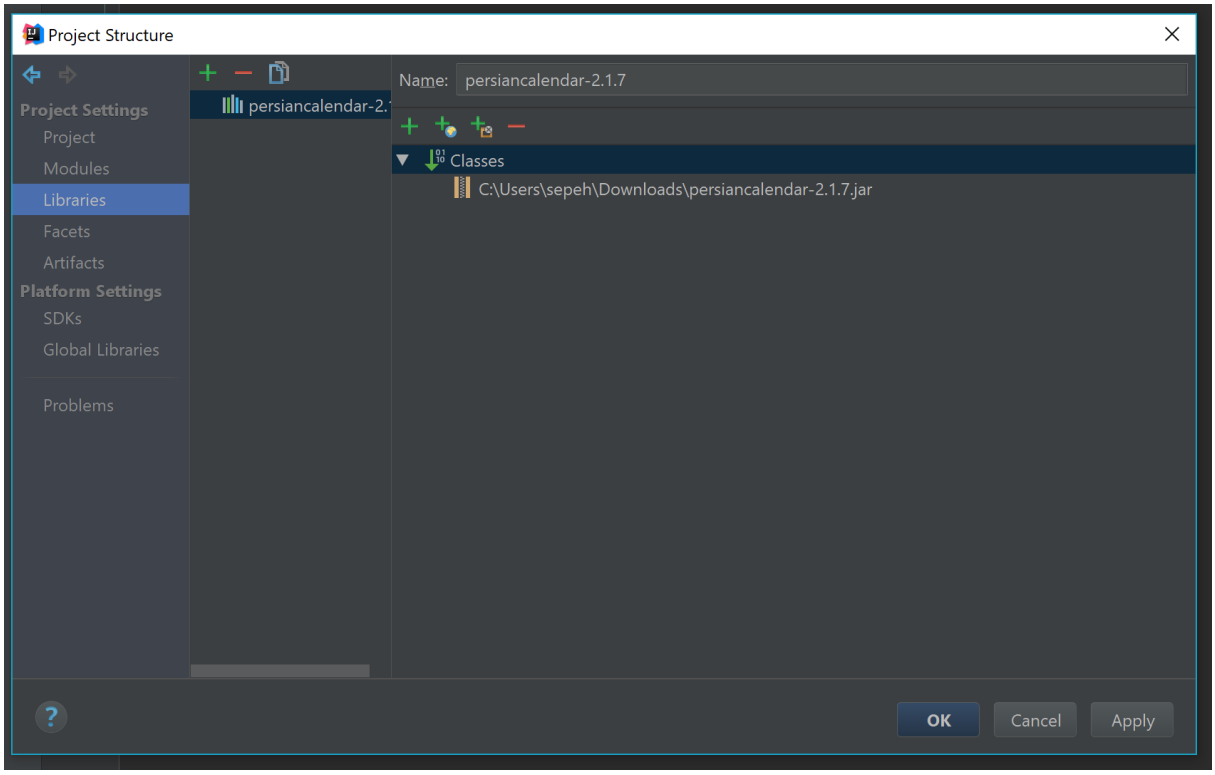
بعضی از کتابخانه‌های پرکاربرد در JDK به طور پیش‌فرض وجود دارد. برای مثال کتابخانه java.util از این دسته کتابخانه‌ها است. این کتابخانه شامل کلاس‌های کاربردی مانند ساختمان داده‌های مختلف و تولید داده‌های تصادفی است. برای استفاده از این کتابخانه کافیست java.util را در ابتدای فایل خود import کنید. در این جلسه از تعدادی از این کتابخانه‌ها استفاده خواهیم کرد.

[3] Build Automation System





شکل ۵ - افزودن کتابخانه توسط IntelliJ



شکل ۶ - افزودن کتابخانه توسط IntelliJ

انجام دهید: پیاده‌سازی یک نرم‌افزار رای‌گیری

از شما خواسته شده است تا یک نرم‌افزار رای‌گیری با استفاده از زبان جاوا بنویسید. در این نرم‌افزار، فرد می‌تواند یک رای‌گیری ایجاد کرده و پس از ساختن رای‌گیری، سایرین می‌توانند آرای خود را ثبت نمایند. رای‌گیری می‌تواند دارای دو مدل باشد:

۱- هر فرد تنها بتواند یک رای بدهد.

۲- هر فرد بتواند چندین رای بدهد.

در ابتدا کلاس‌ها و متدهای مربوط به این نرم‌افزار را طراحی کرده و سپس آن‌ها را پیاده‌سازی می‌کنیم.

در این نرم‌افزار به یک کلاس Voting نیاز است که در آن وضعیت رای‌گیری (در حال انجام یا پایان‌یافته)، نوع رای‌گیری (تک رای و چند رای)، پرسش رای‌گیری، گزینه‌های رای‌گیری و آرای اخذشده نگه‌داری می‌شود. این کلاس باید شامل متدهایی باشد که تعداد آرای اخذشده، نتیجه تا این لحظه و افراد رای‌دهنده را بازگرداند.

علاوه بر این، یک کلاس VotingSystem نیاز است که در آن رای‌گیری‌های ساخته و ذخیره می‌شوند. این کلاس شامل لیستی از رای‌گیری‌های فعال است و باید متدهایی داشته باشد که با آن‌ها بتوان یک رای‌گیری را ایجاد و حذف کرد. همچنین دو متد برای شروع و پایان رای‌گیری نیز نیاز است.

هنگام شروع نرم‌افزار از کاربر خواسته می‌شود یا به حالت رای‌گیری برود یا به حالت مدیریت برود. در حالت مدیریت می‌تواند یک رای‌گیری را ایجاد، فعال، غیرفعال و حذف نماید. همچنین باید بتواند تعداد رای‌گیری‌های موجود، وضعیت آن‌ها، تعداد رای داده‌شده و تعداد افراد شرکت‌کننده در هر کدام را دریافت نماید.

در حالت رای‌گیری نیز کاربر با انتخاب یکی از رای‌گیری‌ها و واردکردن اسم خود، وارد گزینه‌های مربوط به رای‌گیری شده و می‌تواند رای خود را ثبت کند. اگر رای‌گیری از نوع چند رای باشد، کاربر چند گزینه انتخاب می‌کند ولی اگر از نوع تک رای باشد، فقط یک گزینه می‌تواند انتخاب کند. در هر دو مدل رای‌گیری، کاربر پس از ثبت رای دوباره نمی‌تواند رای بدهد. پس باید بررسی کنید که اسم فرد رای‌دهنده قبلا وجود نداشته باشد. همچنین کاربر باید بتواند بین حالت رای‌گیری و مدیریت، جابه‌جا شود.

برای ذخیره‌سازی رای‌ها لازم است از Collectionهایی مانند ArrayList، HashMap، HashSet و کتابخانه Random استفاده کنید.

در این قسمت کتابخانه‌های گفته‌شده را بیشتر توضیح می‌دهیم:

- ArrayList: این کتابخانه به شما کمک می‌کند تا بتوانید اشیا و مقادیر مختلف را در یک لیست (آرایه) ذخیره کنید. این کتابخانه امکانات زیادی برای جستجو و به‌روزرسانی مقادیر درون آن به شما می‌دهد. برای دانستن روش استفاده از این کتابخانه به این لینک مراجعه کنید. **از این کتابخانه برای ذخیره‌سازی رای‌گیری‌ها استفاده می‌کنیم.**
- HashMap: این کتابخانه به شما کمک می‌کند تا بتوانید نگاشتی از یک شی به شی دیگری را نگه دارید. برای دانستن روش استفاده از این کتابخانه به این لینک مراجعه کنید. **برای نگاشت هر رای‌دهنده به رای داده‌شده توسط وی از این کتابخانه استفاده می‌کنیم.**
- HashSet: این کتابخانه به شما امکان پیاده‌سازی یک مجموعه از اشیا را می‌دهد، به نحوی که امکان اضافه‌کردن شی تکراری به آن نیست. برای دانستن روش استفاده از این کتابخانه به این لینک مراجعه کنید. **برای جلوگیری از رای‌دادن چندباره کاربران از این کتابخانه استفاده می‌کنیم.**
- Random: **در رای‌گیری‌های تک رای کاربران می‌توانند با انتخاب گزینه "انتخاب تصادفی" یک گزینه را به طور تصادفی انتخاب نمایند. برای ایجاد گزینه تصادفی، از کتابخانه Random استفاده کنید.** کلاس Random برای شما راهی برای تولید اعداد تصادفی ایجاد می‌کند. شما می‌بایست در ابتدا یک نمونه از این کلاس ساخته و پس

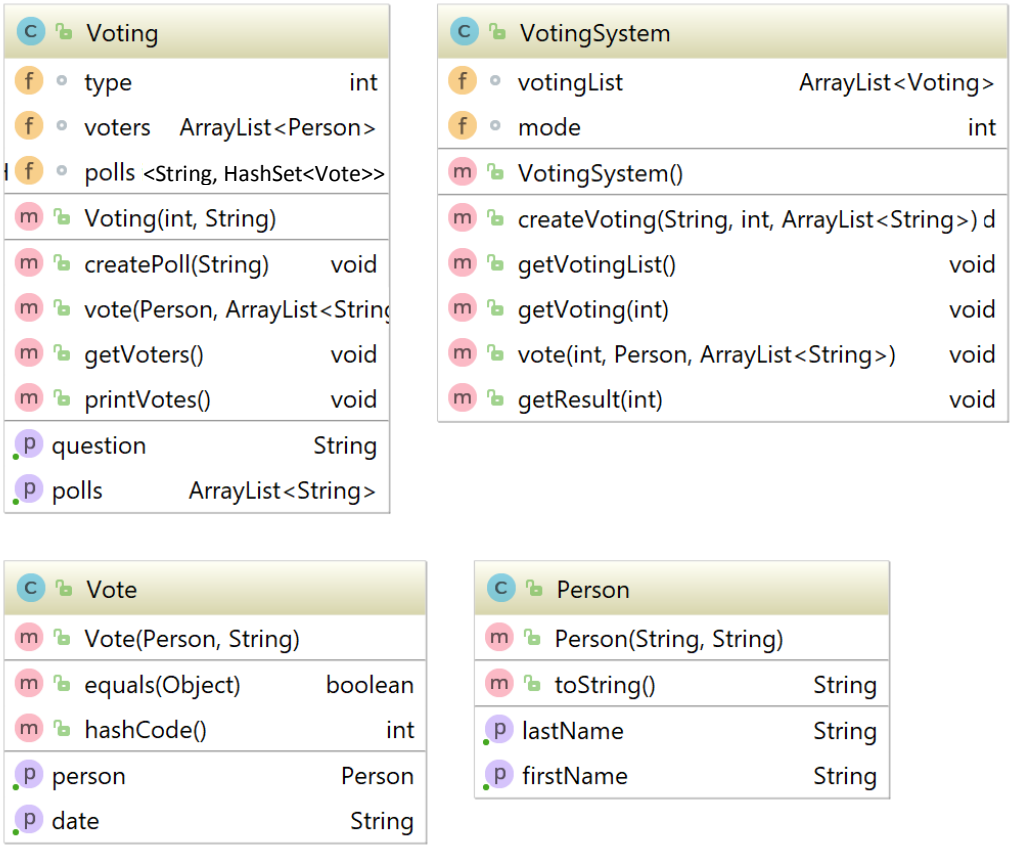
از آن، با استفاده از متدهای موجود در این کلاس، که نمونه‌هایی از آن در ادامه آمده است، انواع داده‌های تصادفی را تولید کنید.

```
Random r = new Random();
r.nextBoolean(); // Generate random boolean
r.nextInt(); // Generate random integer
r.nextInt( bound: 10); // Generate random integer in [0, 10)
```

شکل ۷ – مثال‌هایی از متدهای موجود در کلاس Random برای تولید مقدار تصادفی

در نرم‌افزار رای‌گیری لازم است که تاریخ رای داده‌شده به هجری خورشیدی ذخیره شود. این نوع تاریخ‌نگاری به طور پیش‌فرض در جاوا وجود ندارد. به همین دلیل کتابخانه مربوط به تاریخ هجری خورشیدی (موجود در پیوست) را به پروژه اضافه کنید و از آن استفاده کنید. استفاده از این کتابخانه بسیار ساده است. برای آشنایی با این کتابخانه به این لینک مراجعه کنید.

یک مهندس خوب نرم‌افزار، قبل از پیاده‌سازی برنامه، مسئله را به خوبی تحلیل و طراحی می‌کند. از همین رو، قبل از شروع برنامه‌نویسی، کلاس‌ها و اشیای مورد نیاز از آن‌ها، فیلدهای آن‌ها و متدها را طراحی می‌کنیم. شکل زیر نمونه‌ای از تحلیل و طراحی‌ای است که برای این برنامه انجام شده است. با توجه به این طراحی، برنامه را بنویسید.



نکاتی پیرامون مدل حافظه در جاوا

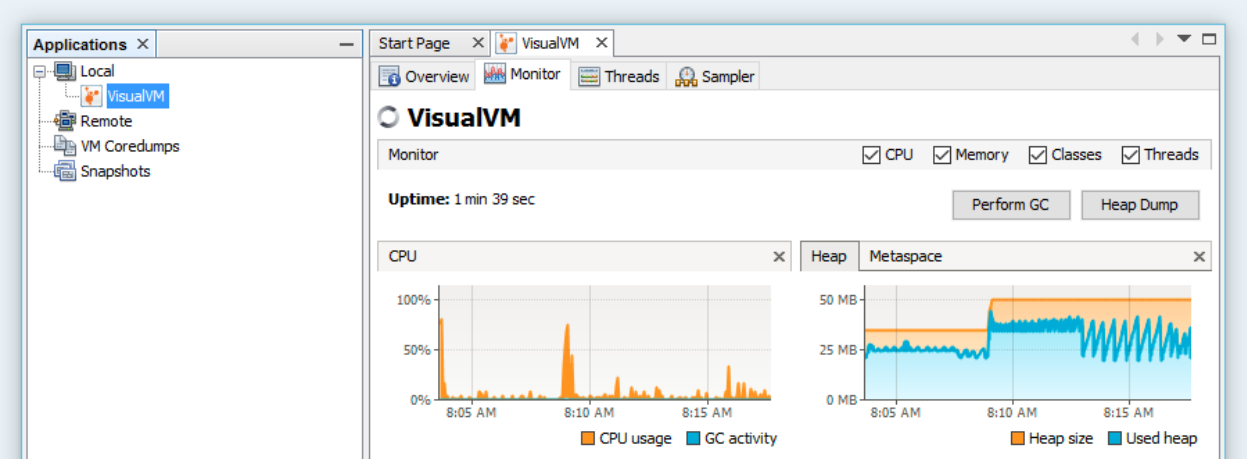
در کلاس با مدل حافظه در جاوا، عملکرد هر قسمت از آن و ارتباط بین متغیرها و نمونه‌ها آشنا شده‌اید. در این جا قصد داریم برخی نکات مرتبط با مبحث حافظه در جاوا را ذکر کنیم:

۱- اشیای تغییرناپذیر (immutable objects): متدهای موجود در یک کلاس اغلب برای تغییر وضعیت یک نمونه ساخته‌شده از آن کلاس طراحی می‌شوند. اگر این متدها، مقادیر فیلدهای یک شی را تغییر دهند، وضعیت و حالت شی تغییر کرده است و این تغییرات، برای سایر اشیایی که به آن شی دسترسی داشته باشند، قابل درک است. در برنامه‌نویسی به زبان جاوا می‌توان اشیایی ایجاد کرد که تغییرناپذیر باشند. متدهای این اشیا، تغییری در مقادیر فیلدها ایجاد نمی‌کنند. یکی از معروف‌ترین مثال‌های این‌گونه اشیا، نمونه‌های ساخته‌شده از کلاس String هستند. اشیایی که از این کلاس ساخته می‌شوند، تغییرناپذیر هستند، به این معنی که فراخوانی متدهای آن‌ها، وضعیت شی را تغییر نمی‌دهد؛ بلکه یک شی جدید می‌سازد و آن را برمی‌گرداند. به مثال زیر توجه کنید:

```
String s = "Hello world";
String sr = s.replace( oldChar: 'H', newChar: 'C');
System.out.println(s); // Hello world
System.out.println(sr); // Cello world
```

انجام دهید: مستندات متدهای concat، toUpperCase، toLowerCase، compareTo، replaceAll، split، subString و trim از کلاس String را مطالعه کنید.

۲- عملکرد Garbage Collector جاوا: همانطور که می‌دانید، هر شی‌ای که ساخته می‌شود، بخشی از حافظه Heap را به خود اختصاص می‌دهد. این حافظه می‌بایست در زمانی که استفاده‌ای از آن شی نداریم، برای استفاده در ادامه برنامه آزاد شود. این کار همان وظیفه‌ی Garbage Collector است و این کار را از طریق شمارش اشاره‌گرهایی که به آن بخش از حافظه تخصیص‌یافته برای یک شی وجود دارند، انجام می‌دهد. زمانی که هیچ اشاره‌گری به یک شی اشاره نکند، آن شی از حافظه پاک می‌شود. برای اینکه بتوانیم این عملیات را از نزدیک ببینیم، از ابزاری با نام VisualVM استفاده می‌کنیم. این ابزار به صورت پیش‌فرض به همراه JDK نصب می‌گردد و محصولی از شرکت Oracle است.



شکل ۸ - نمای نرم افزار VisualVM

انجام دهید: یک‌بار دیگر نرم‌افزار رای‌گیری را اجرا کنید و همزمان تغییرات حافظه را از طریق این ابزار مشاهده کنید.