به نام خدا

دستور کار کارگاه برنامه‌نویسی پیشرفته
جلسه اول

مقدمه‌ای بر جاوا و برنامه‌نویسی ساخت‌یافته در آن

مقدمه

در نخستین جلسه از کارگاه برنامه‌نویسی پیشرفته قصد داریم با ساختار کلی زبان برنامه‌نویسی جاوا آشنا شویم. مطالبی که در این جلسه مورد بررسی قرار می‌گیرند عبارتند از:

- نصب و راه‌اندازی محیط توسعه یکپارچه (IDE[1])
- آشنایی با نحوه ایجاد یک پروژه
- ساختار کلی برنامه‌ها در زبان برنامه‌نویسی جاوا
- آشنایی با قواعد نحوی زبان جاوا
- مروری بر ساختارهای کنترلی و متغیرها در جاوا
- آشنایی با نحوه اجرای برنامه‌ها در جاوا
- مروری بر اشکال‌زدایی برنامه‌ها در محیط توسعه یکپارچه مورد استفاده
- نصب گیت و ساخت اکانت و پروژه

مراحل انجام کار

نصب و راه‌اندازی محیط توسعه یکپارچه

در این دوره از کارگاه‌های برنامه‌نویسی پیشرفته، از نرم‌افزار IntelliJ به عنوان محیط توسعه یکپارچه استفاده می‌نماییم . این نرم‌افزار، یکی از محصولات شرکت JetBrains است که امکان استفاده از نسخه Professional آن، برای دانشجویان به طور رایگان فراهم است.

می‌توانید نسخه Ultimate این نرم‌افزار را از اینجا دریافت نمایید. برای ایجاد یک حساب کاربری و راه‌اندازی نسخه Professional این محصول، لازم است به پایگاه اینترنتی

[1] Integrated Development Environment

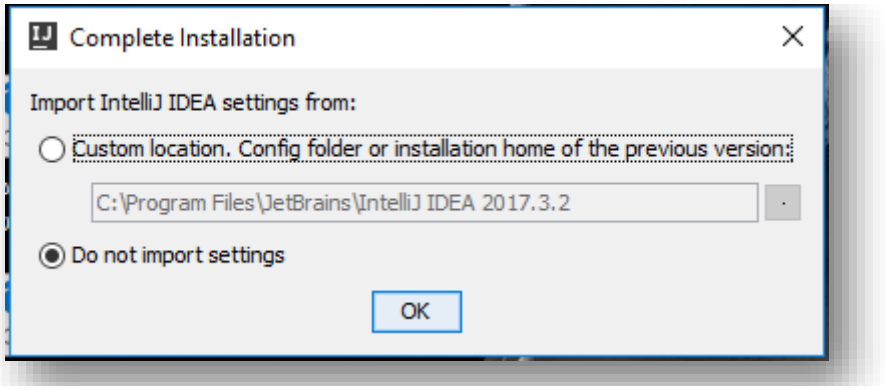
[2] بدیهی است با توجه به محدودیت‌های موجود، امکان بررسی تمامی ابعاد و ویژگی‌های این محصول در این دوره وجود ندارد. مطلوب است دانشجویان، مطالعات بیشتری در رابطه با محیط توسعه مورد استفاده، ابعاد، ویژگی‌ها و امکانات آن انجام دهند.

https://www.jetbrains.com/student مراجعه کرده و مراحل کار را از آن‌جا دنبال کنید. کافیست تا پس از نصب برنامه، با License ایجادشده برای شما، Register کنید.

برنامه‌های نوشته‌شده به زبان جاوا، برای اجرا، نیازمند Java SDK [۳] هستند. برای دریافت SDK مناسب با سیستم‌عامل خود می‌توانید به اینجا مراجعه نمایید. در رابطه با وظیفه SDK و نقش آن در فرایند اجرای برنامه‌های جاوا در ادامه این دستورکار بیشتر توضیح خواهیم داد.

مراحل نصب IntelliJ IDEA

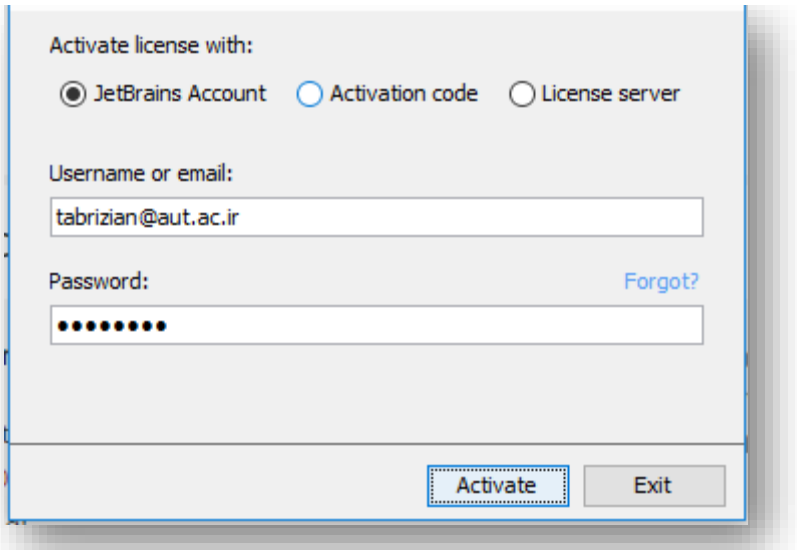
با اجرای فایل نصب نرم‌افزار، پنجره شکل۱ نمایش داده می‌شود. در صورتی‌که قبلاً از نسخه دیگری از این نرم‌افزار استفاده می‌کردید، می‌توانید با انتخاب گزینه اول، تمامی تنظیمات اعمال‌شده روی نسخه قبلی را برای نسخه جدید نیز استفاده نمایید. در غیر این صورت گزینه دوم را انتخاب نموده و به مرحله بعدی بروید.



شکل ۱ – صفحه اول پس از اجرای نصب

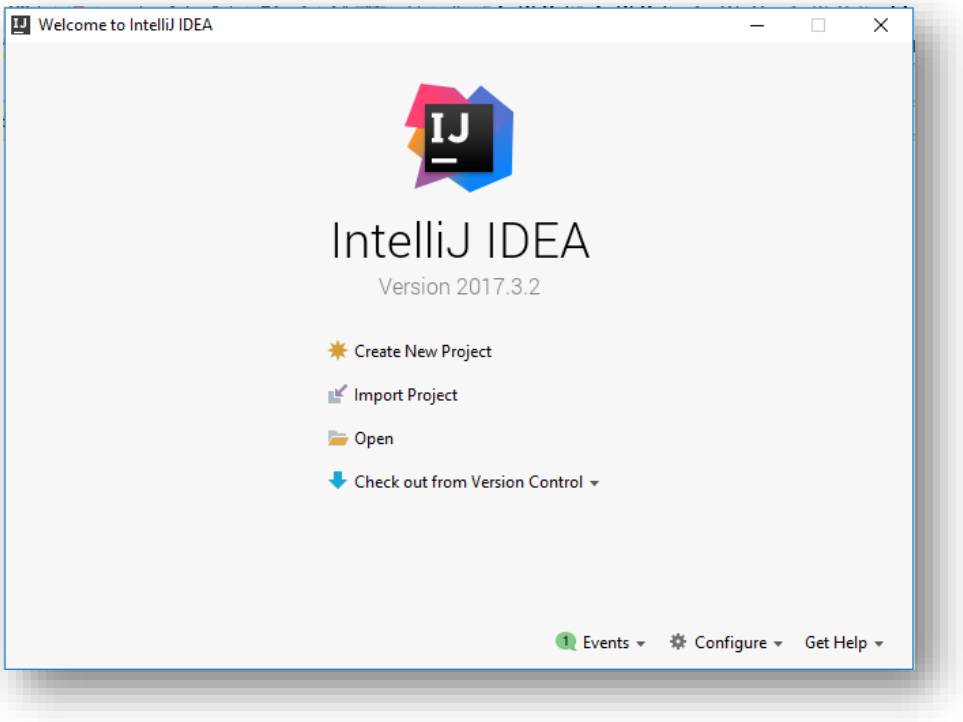
در مرحله بعدی و مطابق شکل ۲، پنجره‌ای برای فعال‌سازی نرم‌افزار نمایش داده می‌شود. در این پنجره می‌توانید با انتخاب گزینه فعال‌سازی از طریق حساب کاربری و وارد نمودن نام و رمز عبور حساب کاربری خود، نرم‌افزار خود را فعال نمایید.

۳ Standard Development Kit



شکل ۲ – فعال‌سازی نرم‌افزار

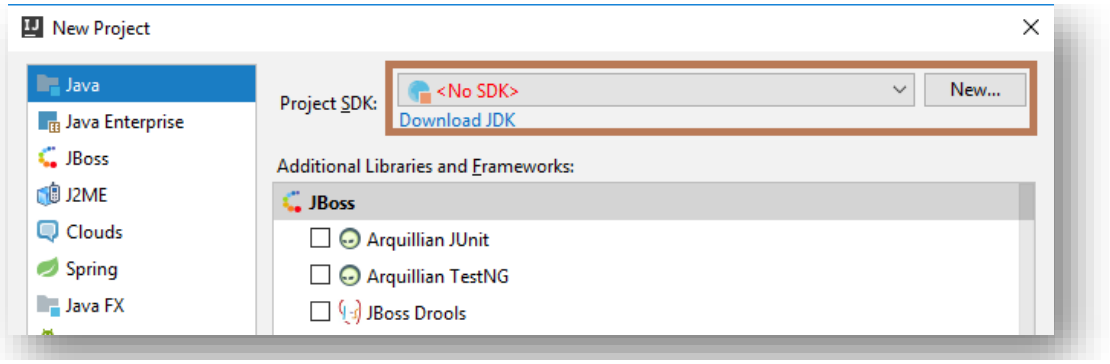
با اتمام فرایند نصب نرم‌افزار، پنجره شکل ۳ نمایش داده می‌شود. در این مرحله، فرایند نصب نرم‌افزار تکمیل شده است و از این پس می‌توانید شروع به برنامه‌نویسی نمایید.



شکل ۳ – صفحه ایجاد پروژه جدید

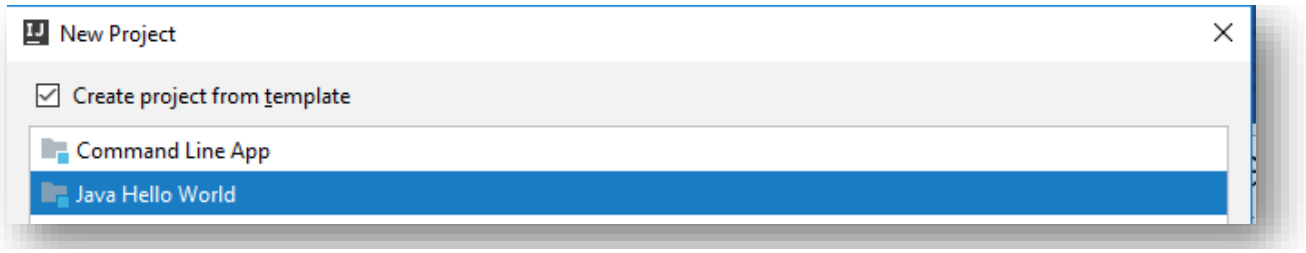
ایجاد پروژه جدید

با انتخاب گزینه Create New Project در پنجره شکل ۳، می‌توانید یک پروژه جاوای جدید ایجاد نمایید. با انتخاب این گزینه، پنجره‌ای مشابه شکل ۴ نمایش داده می‌شود. در منوی سمت چپ می‌توانید انواع پروژه‌های قابل ایجاد را مشاهده نمایید. در این منو، روی اولین گزینه، پروژه Java کلیک نمایید. در منوی سمت راست، باید SDK مورد استفاده در این پروژه را به محیط توسعه یکپارچه معرفی نمایید. با انتخاب گزینه New، مسیر نصب SDK را مشخص نموده و مراحل را تا انتهای کار دنبال نمایید.



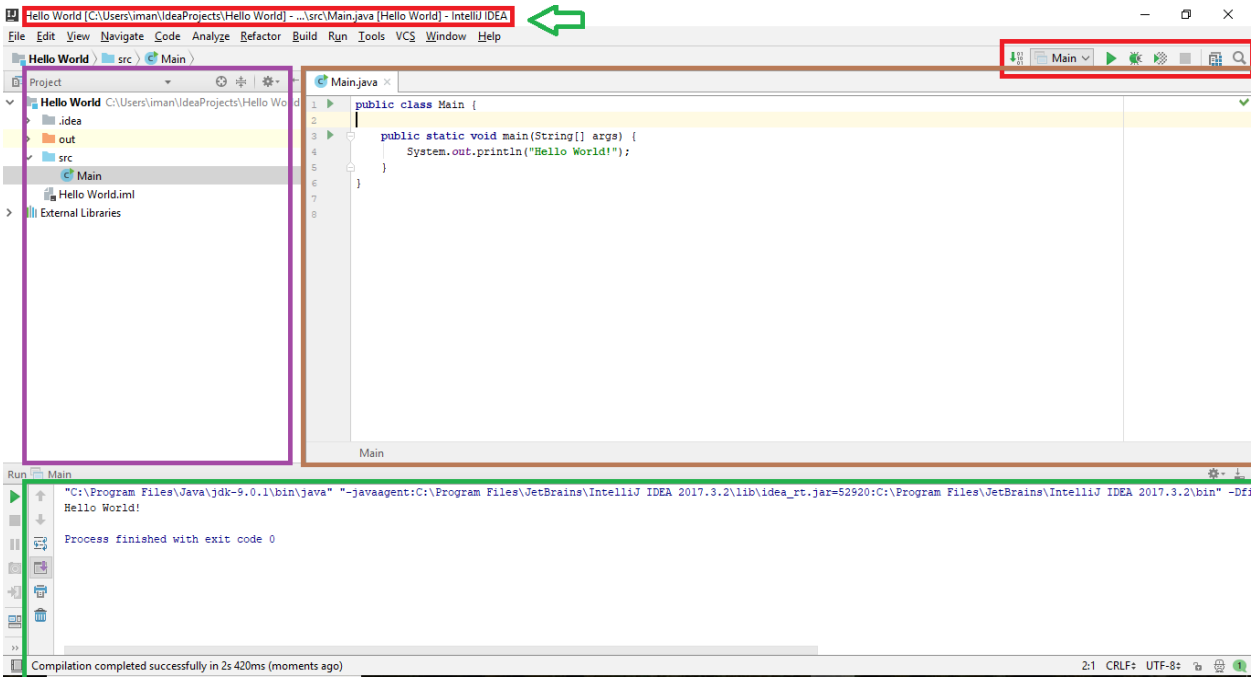
شکل ۴ – راه اندازی SDK

پس از نصب SDK، پنجره‌ای مشابه شکل ۵ نمایش داده می‌شود. در این پنجره می‌توانید قالب آماده‌ای را برای پروژه خود انتخاب نمایید. روی گزینه Java Hello World کلیک کرده و به مرحله بعدی بروید.



شکل ۱ – انتخاب نوع پروژه جدید

با ایجاد پروژه جدید، پنجره اصلی نرم‌افزار، مشابه با شکل ۶ نمایش داده می‌شود.



شکل ۶ – محیط IntelliJ IDEA

آدرس ریشه پروژه، در میله عنوان نمایش داده می‌شود. میله ابزار در زیر میله عنوان در بالای صفحه قرار دارد. کادر بنفش‌رنگ ساختار فایل پروژه، بسته‌ها و کلاس‌های تعریف‌شده را نمایش می‌دهد. کادر سبزرنگ که در پایین صفحه نشان داده شده است، خروجی برنامه به همراه Return code و اطلاعات مربوط به اجرای برنامه را نشان می‌دهد. کادر قهوه‌ای محیط ویرایش کد را نمایش می‌دهد.

با اتمام فرایند ایجاد پروژه، یک پوشه به نام پروژه در آدرس انتخابی شما ایجاد می‌شود که فایلی به اسم Main.java در آن وجود دارد. با بازکردن این فایل در محیط توسعه می‌توانید کد شکل ۷ را در آن مشاهده نمایید.

```java
public class Main {

    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

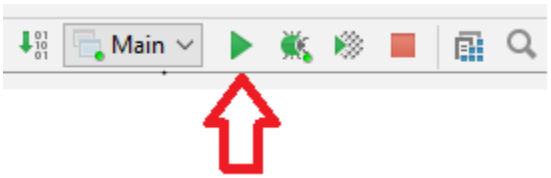
شکل ۷ – کد برنامه Hello World

۴ Title Bar

تابع main نوشته شده در این فایل، نقطه آغازین اجرای برنامه است. همان‌طور که در شکل ۷ مشخص است، ورودی این تابع یک آرایه رشته‌ای به نام args است. تمامی مقادیری که در هنگام اجرای برنامه به آن پاس داده می‌شوند، در این آرایه قرار می‌گیرند. در ادامه، نحوه استفاده از این آرگومان مورد بررسی قرار خواهیم داد.

دستور System.out.println، با دریافت یک رشته، آن را در کنسول چاپ می‌نماید. در جلسات بعدی، پس از آشنایی با ساختار کلاس‌ها، جزئیات بیشتری را در رابطه با این دستور مشاهده خواهید کرد.

حال برای اجرای برنامه کافی است تا دکمه Run را از منوی زیر اجرا کنید.



شکل ۸ – منوی اجرای پروژه

انواع داده در جاوا

جدول ۱، انواع داده‌های اولیه مورد استفاده در جاوا را به همراه[۵] اطلاعاتی در رابطه با آن‌ها نمایش می‌دهد.

جدول ۱ – جدول انواع داده

Type	Description	Initial Value	Size	Example Literals
boolean	true or false	false	۱ bit	true, false
byte	twos complement integer	۰	۸ bits	
char	Unicode character	\u۰۰۰۰	۱۶ bits	‘a’, ‘\u۰۰۴۱’, ‘\\’
short	twos complement integer	۰	۱۶ bits	
int	twos complement integer	۰	۳۲ bits	-۲, -۱, ۰, ۱, ۲
long	twos complement integer	۰	۶۴ bits	-۲L, -۱L, ۰L
float	IEEE ۷۵۴ floating point	۰.۰	۳۲ bits	۱.۲۳e۱۰۰f,.۳f, ۳.۱۴F
double	IEEE ۷۵۴ floating point	۰.۰	۶۴ bits	۱.۲۳۴۵۶e۳۰۰d, ۱.۲۳۴e-۳d, ۱e۱d

۵ Primitive Data Types

تعریف متغیر در جاوا

برای تعریف متغیر در جاوا، ابتدا نوع داده و سپس نام متغیرها را در یک خط مشخص می‌نماییم. شکل ۹، نمونه‌ای از تعریف متغیر در جاوا را نمایش می‌دهد.

```
int a, b, c;
a = ۱۲۳۴;
b = ۹۹;
c = a + b;
int[] arr = new int[۵۰];
arr[۰] = ۰;
```

شکل ۹ – تعریف متغیر و آرایه

ساختارهای کنترلی در جاوا

قواعد نحوی ساختار کنترلی if در جاوا، کاملاً مشابه با قواعد زبان C است. شکل ۱۰، نمونه‌ای از کاربرد این دستور در جاوا را نمایش می‌دهد. خروجی این قطعه کد چیست؟

```
public class Main {

    public static void main(String[] args) {

        int i = ۰;

        if (i == ۰)
            System.out.println("i is ۰");
        else
            System.out.println("i is not ۰");
    }
}
```

شکل ۱۰ – استفاده از ساختار کنترلی if

شکل ۱۱، نمونه‌ای از استفاده از ساختار switch-case در زبان جاوا را نمایش می‌دهد. خروجی این قطعه کد چیست؟

```
public class Main {
    public static void main(String[] args) {
        int day = ۴;
        switch (day) {
            case ۰:  System.out.println("Sunday");
                     break;
            case ۱:  System.out.println("Monday");
                     break;
            case ۲:  System.out.println("Tuesday");
                     break;
            case ۳:  System.out.println("Wednesday");
                     break;
            case ۴:  System.out.println("Thursday");
                     break;
            case ۵:  System.out.println("Friday");
                     break;
            case ۶:  System.out.println("Saturday");
                     break;
            default: System.out.println("invalid day");
        }
    }
}
```

شکل ۱۱ – ساختار Switch Case

حلقه‌ها

قواعد نحوی استفاده از ساختار While و For در زبان جاوا، کاملاً مشابه با قواعد این ساختارها در زبان برنامه‌نویسی C است. شکل‌های ۱۲ و ۱۳، به ترتیب نمونه‌هایی از استفاده از این ساختارها را در زبان جاوا مشخص می‌کند. در هر مورد بگویید خروجی قطعه کد نمایش داده شده چیست؟

```java
public class Main {
    public static void main(String[] args) {

        // print out special cases whose ordinal doesn't end in th
        System.out.println("۱st Hello");
        System.out.println("۲nd Hello");
        System.out.println("۳rd Hello");

        // count from i = ۴ to ۱۰
        int i = ۴;
        while (i <= ۱۰) {
            System.out.println(i + "th Hello");
            i = i + ۱;
        }

    }
}
```

شکل ۱۲ – ساختار while

```java
public class Main {
    public static void main(String[] args) {

        // print out special cases whose ordinal doesn't end in th
        System.out.println("۱st Hello");
        System.out.println("۲nd Hello");
        System.out.println("۳rd Hello");

        // count from i = ۴ to ۱۱
        for (int i = ۴; i <= ۱۱; i++) {
            System.out.println(i + "th Hello");
        }

    }
}
```

شکل ۱۳ – ساختار for

ساختار حلقه دیگری که در زبان جاوا وجود دارد، موسوم به For-Each است. این ساختار برای ایجاد یک حلقه روی یک آرایه از هر نوع و انجام یک عملیات خاص روی تک تک عناصر آن آرایه مورد استفاده قرار می‌گیرد. شکل ۱۴، نمونه‌ای از استفاده از این ساختار را نمایش می‌دهد. لازم به ذکر

است که For-Each تنها برروی عناصر قابل پیمایش قابل اجرا است. این به آن معناست که بر روی نوع‌های داده‌ی خاصی مانند آرایه قابل اجرا است.

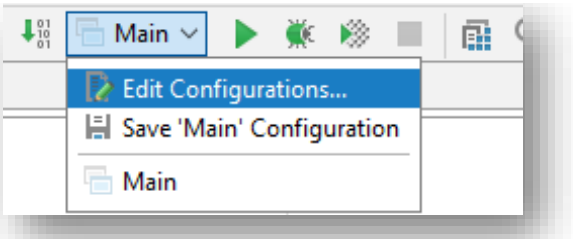
```
String myStr = "Salam";
for (char s : myStr.toCharArray()) {
    System.out.println(s);
}
```

شکل ۱۴- ساختار for-each

در شکل ۱۴، برای اینکه ما رشته را به نوعی قابل پیمایش تبدیل کنیم، بایستی در گام اول آن را به آرایه‌ای از کاراکترها تبدیل کرده و سپس برای هر کاراکتر موجود در این آرایه عمل مورد نظر خود را که همان چاپ کردن رشته است انجام دهیم.

پاس دادن Argument به برنامه‌های جاوا در محیط IntelliJ IDEA

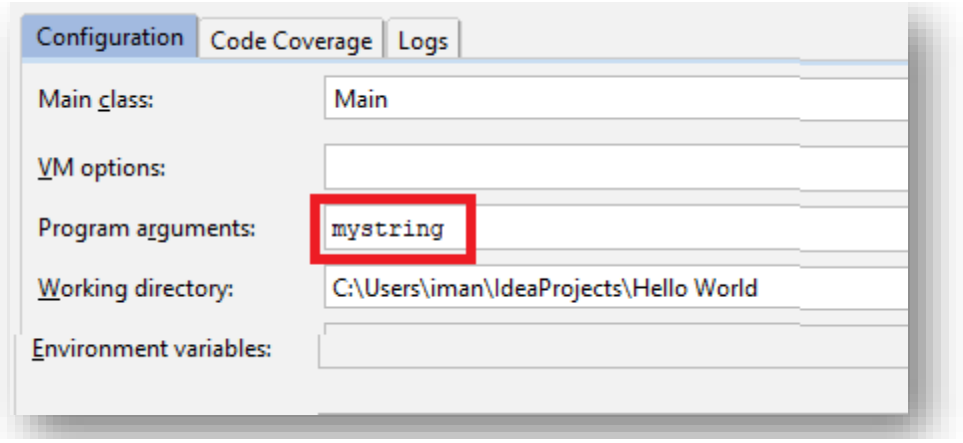
Argumentها متغیرهایی هستند که در زمان اجرا به برنامه پاس داده می‌شوند و برنامه متناسب با ورودی داده‌شده اجرا می‌شود. به عنوان مثال، فرض کنید به جای آن‌که شما در شکل ۱۴، تک تک حروف یک کلمه‌ی از پیش تعیین‌شده را چاپ کنید، قصد داشتید تا رشته موردنظر را به عنوان یکی از Argumentهای ورودی برنامه دریافت کنید. لازم به ذکر است که این روش با روش‌هایی مانند scanf در زبان C تفاوت دارد. در scanf برنامه در حین اجرا از کاربر درخواست ورودی می‌کند، در صورتی که در این روش، آرگومان‌های ورودی برنامه قبل از اجرای برنامه دریافت شده‌اند و در آرایه‌ی String[] args قرار گرفته‌اند. حال برای مقداردهی این آرایه در محیط IntelliJ لازم است تا عملیات‌های زیر را انجام دهید.



شکل ۱۵ – تغییر پیکربندی اجرای برنامه

ابتدا مطابق با شکل ۱۵ گزینه Edit Configurations را انتخاب می‌کنید تا نحوه اجرای برنامه را تغییر دهید. پس از انجام این کار در پنجره‌ای که مطابق با شکل ۱۶ باز می‌شود، عبارت mystring را Program arguments وارد کنید. این بخش شامل رشته‌هایی می‌شود که در آرایه‌ی String[] args قرار داده می‌شوند. عباراتی که در اینجا قرار می‌دهید با فاصله (space) از یکدیگر جدا می‌شود و به ترتیب در آرایه args قرار می‌گیرد که در ورودی تابع main قابل دسترسی است.

پس از انجام این تغییرات گزینه Ok را بزنید و از این صفحه خارج شوید.



شکل ۱۶ – تعیین Argument

حال برنامه شکل ۱۷ را اجرا کنید و خروجی آن را تفسیر کنید.

```
public class Main {

    public static void main(String[] args) {
        String myStr = args[0];
        for (char s : myStr.toCharArray()) {
            System.out.println(s);
        }
    }
}
```

شکل ۱۷ – برنامه چاپ عناصر String با استفاده از Argument

انجام دهید

۱) برنامه‌ای بنویسید که دو عدد دریافت کند و بررسی کند آیا این دو عدد نسبت به هم اول هستند یا خیر.

۲) برنامه‌ای بنویسید که یک جدول ضرب ۱۰ در ۱۰ را حساب کند و در کنسول نمایش دهد.

نحوه اجرای برنامه در جاوا

JVM چیست و چه کاری انجام می‌دهد؟

یکی از ویژگی‌های برجسته زبان جاوا، Cross Platform بودن آن است؛ به این معنی که برنامه‌های نوشته شده در زبان جاوا می‌توانند روی پلتفرم‌ها و سیستم‌عامل‌های مختلفی مانند ماشین‌های ویندوزی، سیستم‌عامل‌های لینوکس و موارد دیگر اجرا شوند.

زبان جاوا به دلیل داشتن یک ماشین مجازی به نام JVM ، قادر به ارائه چنین ویژگی مُهمی است. از همین رو، آشنایی با ماشین مجازی جاوا و نقش آن در اجرای برنامه‌ها از اهمیت بالایی برخوردار است.

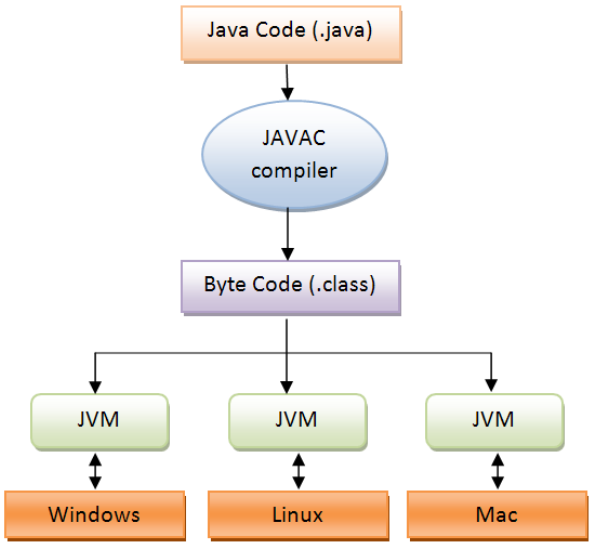
برنامه‌های نوشته شده در زبان جاوا، برخلاف برنامه‌های نوشته شده در زبان‌هایی مانند C، کامپایل و سپس اجرا نمی‌شوند. فرایند کامپایل این برنامه‌ها از دو بخش تشکیل شده است:

- کامپایل‌کردن کد منبع برنامه به یک زبان میانی (bytecode)
- اجرای فایل برنامه تبدیل‌شده به زبان میانی توسط کامپایلر/مفسر در محیط زمان اجرای جاوا (JRE[۷])

طی این فرایند، فایل‌های با پسوند .java که فایل‌های متنی ساده‌ای هستند و فقط متن اصلی برنامه را ذخیره می‌نمایند، به کامپایلر جاوا داده می‌شوند. این کامپایلر با دریافت این فایل‌ها، فایل‌هایی به یک زبان میانی تولید می‌نماید که برای مفسرها/کامپایلرهای محیط زمان اجرای جاوا قابل فهم هستند. این فایل‌های زبان میانی با پسوند .class ذخیره می‌شوند. زبان میانی جاوا برای تمام مفسرها/کامپایلرهای جاوا روی هر پلتفرمی قابل فهم است. کافیست این فایل‌های میانی، به مفسر مربوطه روی یک پلتفرم داده شود تا مفسر بتواند آن را اجرا نماید. به این ترتیب، نیازی به دریافت کامپایلر جاوا برای تمام پلتفرم‌ها و کامپایل کردن کد اصلی برای تمام پلتفرم‌ها به طور جداگانه وجود ندارد. شکل ۱۸، این فرایند را به طور کامل نمایش می‌دهد.

۶ Java Virtual Machine

۷ Java Runtime Environment



شکل ۱۸ – Cross Platform بودن جاوا

JRE و JDK چیست؟

JRE مجموعه ابزارهایی است که برای اجرای برنامه‌های مبتنی بر جاوا استفاده می‌شود. بخش اصلی این ابزار JVM است. JRE مخفف Java Runtime Environment است. درواقع JRE تنها امکان اجرای نرم‌افزارهای جاوا را فراهم می‌کند. JDK (Java Development Kit) علاوه بر ابزارهایی که JRE در خود دارد، ابزارهای مورد نیاز برای کامپایل و اشکال‌یابی برنامه‌های جاوا را نیز داراست. پس برای توسعه برنامه‌های به زبان جاوا نیازمند نصب JDK هستیم که پیشتر انجام شد.

اجرای برنامه‌ها از طریق CLI (Command-Line Interface)

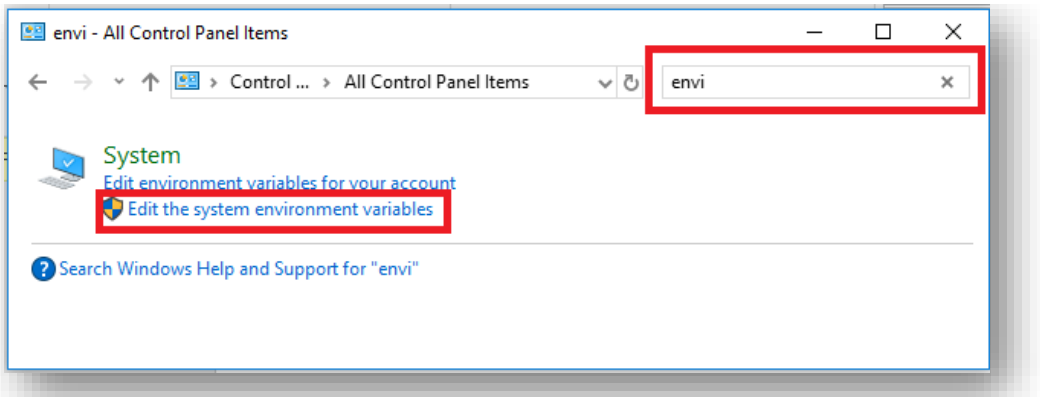
در بخش قبل، با نحوه اجرای برنامه در محیط توسعه یکپارچه آشنا شدیم. در این قسمت می‌خواهیم فرایند اجرای برنامه در محیط ترمینال یا دستوری را بررسی نماییم. پس از نصب JDK و برای استفاده از آن در سیستم‌عامل ویندوز کافی است متغیر محیطی PATH را به درستی تنظیم کنید[۸] تا به آن دسترسی داشته باشید.

نحوه تنظیم متغیرهای محیطی در Windows ۱۰

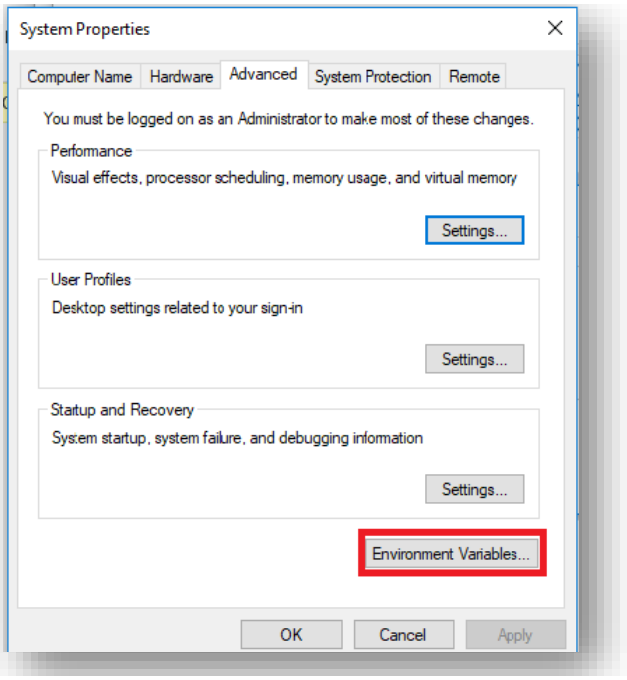
برای این کار کافی است تا از طریق Control Panel و در قسمت Search، عبارت environment را تایپ کنید. سپس گزینه Edit the System Environment variables را انتخاب کرده و از پنجره باز شده عبارت Environments را انتخاب کنید. سپس از پنجره باز شده روی PATH کلیک کرده و گزینه

۸ Environment Variable

Edit را فشار دهید. از داخل پنجره جدید گزینه New را انتخاب کرده و آدرس پوشه bin موجود در JDK را به انتهای مقادیر موجود اضافه نمایید.



شکل ۱۹ – جست و جو در Control Panel



شکل ۲۰ – System Properties

اکنون سیستم‌عامل ویندوز شما برای اجرای برنامه‌های مبتنی بر جاوا پیکربندی شده است (برای محیط‌های لینوکسی و Mac تنظیمات به نحو دیگری است). اگر پیکربندی شما به درستی انجام شده باشد، بایستی پس از اجرا کردن cmd.exe و اجرای دستور java –version، نسخه نصب‌شده جاوا در ترمینال نمایش داده شود.

حال قصد داریم تا با استفاده از این محیط پیکربندی شده برنامه‌ی چاپ حروف یک رشته را در محیط ترمینال اجرا کنیم.

یک پوشه جدید در آدرس دلخواه خود ایجاد نموده و فایل Main.java مثال شکل ۱۷ را در آن کپی کنید. سپس با استفاده از دستور cd در cmd به پوشه ایجاد شده بروید. در cmd دستور زیر را اجرا نمایید.

```
javac Main.java
```

دستور فوق، فایل Main.java را با استفاده از کامپایلر javac به زبان میانی کامپایل کرده و خروجی را در همان پوشه ایجاد می‌نماید. بررسی کنید در پوشه جاری، فایلی به نام Main.class ایجاد شده باشد. پس از انجام این کار دستور زیر را برای اجرای برنامه وارد نمایید.

```
java Main mystring
```

با این کار، ماشین مجازی جاوا، bytecode ایجاد شده در فایل Main.class را خوانده و آن را اجرا می‌نماید. به این ترتیب، برنامه نوشته‌شده توسط شما اجرا خواهد شد. لازم به ذکر است که IDE IntelliJ نیز در هنگام اجرای برنامه‌های شما، از همین دستورات برای کامپایل و اجرای آن‌ها استفاده می‌کند.

انجام دهید

برنامه‌ای بنویسید که با استفاده از ساختار For-Each، تمامی آرگومان‌های ارسال‌شده به تابع main را چاپ نماید (مثال دستور کار در بخش For-Each). این برنامه را از طریق CLI کامپایل و اجرا نمایید.

اشکال‌یابی

یکی از مهم‌ترین مهارت‌های برنامه‌نویسی، مهارت اشکال‌یابی و اشکال‌زدایی است که خود به عنوان یکی از مهارت‌های اصلی در زمینه تولید و توسعه نرم‌افزار محسوب می‌شود. در این بخش، قصد داریم امکانات اشکال‌یابی IntelliJ را مورد بررسی قرار دهیم.

نرم‌افزار IntelliJ به برنامه‌نویسان این امکان را می‌دهد تا بتوانند برنامه خود را به صورت خط به خط اجرا کرده و مقدار تمامی متغیرهای موجود را در هر لحظه مشاهده کنند. برای استفاده از این امکان، باید یک نقطه وقفه در برنامه، قرار داده شود. پس از قراردادن این نقطه در برنامه، با اجرای

۹ Break Point

برنامه در حالت اشکال‌یابی، می‌توان از امکان اجرای خط به خط برنامه و مشاهده مقادیر متغیرها استفاده نمود.

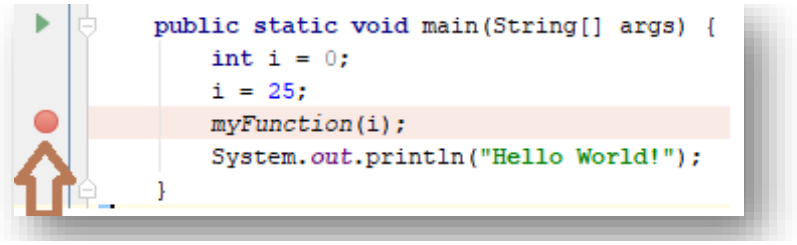
اکنون برنامه زیر را در نظر بگیرید. این برنامه این کاربرد را دارد که یک تابع در آن تعریف شده است که مقدار ورودی را یک واحد افزایش می‌دهد و سپس مقدار جدید را به همراه یک رشته کاراکتر چاپ می‌کند.

```
public class Main {

    public static void myFunction(int i) {
        i++;
        System.out.println("The variable you passed was " + i);
    }
    public static void main(String[] args) {
        int i = ۰;
        i = ۲۵;
        myFunction(i);
        System.out.println("Hello World!");
    }
}
```

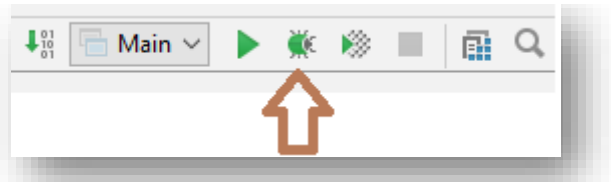
شکل ۲۱ – قطعه کد جهت اشکال‌یابی

شما فرض کنید در خط ۱۱ یک نقطه وقفه قرار می‌دهید. برای قرار دادن نقطه وقفه کافی است تا ناحیه مشخص‌شده در شکل ۲۲ را فشار دهید تا یک علامت دایره قرمز نشان داده شود.



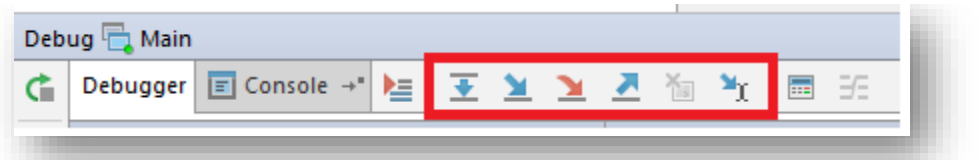
شکل ۲۲ – قراردادن نقطه توقف

حال برای اجرای برنامه در حالت اشکال‌یابی کافی است بر روی شکل زیر در قسمت بالا و سمت راست صفحه کلیک کنید.



شکل ۲۳ - اجرای برنامه در حالت اشکال‌یابی

با کلیک کردن بر روی این دکمه برنامه در حالت debugging اجرا می‌شود و بر روی خط دارای breakpoint (نقطه وقفه) توقف خواهد کرد و صفحه Editor مقدار تمامی متغیرهای موجود در صفحه را نشان خواهد داد.



شکل ۲۴ – نوار ابزار اشکال‌یابی

دستور موجود در نوار ابزار	کارکرد	کلید میان‌بر
Show Execution Point	روی این دکمه کلیک کنید تا خط فعلی اجرای دستورات را نشان دهد.	alt + F۱۰
Step Over	برنامه را تا خط بعدی اجرا کن و توابع بینابینی Skip کن.	F۸
Step Into	روی این دکمه کلیک کنید تا به داخل تابعی که در حال حاضر Debugger روی آن قرار دارد برود.	F۷
Force Step Into	دستور Step Into بعضی اوقات دستورات مربوط به خود SDK را وارد نمی‌شود با این کار شما Debugger را مجبور به ورود به این دستورات خواهید کرد.	Shift + Alt + F۷
Step Out	با اجرای این دستور Debugger از تابع فعلی خارج خواهد شد و به خط بعدی نقطه فراخوانی آن خواهد رفت.	Shift + F۸
Run to Cursor	تا اجرای دستور در نقطه‌ی مکان‌نما در Editor پیش خواهد رفت.	Alt + F۹

انجام دهید

اکنون با توضیحات داده شده طوری دستورات نوار ابزار را اجرا کنید که به داخل تابع رفته و تنها دستور اول آن را اجرا کند و سپس به دستور بعد از نقطه وقفه برود.

سوال جنبی: آیا می‌دانید چرا به اشکال‌های نرم‌افزاری اصطلاحا bug گفته می‌شود؟

نکاتی درباره برنامه‌نویسی در جاوا

- جاوا یک زبان برنامه‌نویسی حساس به حروف[۱۰] است.
- طبق قرارداد، اسامی متدها (تابع‌ها) باید با حرف کوچک شروع شود. اگر اسم متد از چند کلمه تشکیل شده است، باید اولین حرف کلمه داخلی بزرگ نوشته شود (اصطلاحاً CamelCase). مانند قطعه کد زیر:

```
public void myMethod()
```

- اسم فایل باید حتما با اسم کلاس مطابقت داشته باشد.
- برنامه‌های جاوا از متد main با شکل زیر آغاز می‌شوند.

```
public static void main(String args[])
```

اشکال‌زدایی

۱. در ادامه شما چند قطعه کد مشاهده خواهید کرد و وظیفه شما آن است که اشکالات این قطعه کدها را پیدا کنید.

قطعه کد اول:

```
    public void my_method () {
    }
```

قطعه کد دوم:

```
public class main {

    public void myanothermethod () {
    }
}
```

۱۰ Case Sensitive

۲. توضیح دهید که اگر ما بخش‌های زیر را از برنامه Hello World برداریم، چه خطاهایی رخ خواهد داد.

الف) ;

ب)یکی از ”ها

ج) یکی از آکلادها

۳. توضیح دهید چرا قطعه کد زیر اجرا نمی‌شود؟

```
public class Hello {
   public static void main() {
      System.out.println("Doesn't execute");
   }
}
```

۴. جاوا چه نوع زبان برنامه‌نویسی است: مفسری یا کامپایلری؟ توضیح دهید.

انجام دهید

با استفاده از مستند گیت که در مودل قرار داده شده است و با راهنمایی استاد کارگاه، ابتدا نرم‌افزار گیت را بر روی سیستم خود نصب کنید. سپس، با مراجعه به سایت گیت دانشکده، برای خود یک اکانت ایجاد کرده و به صورت تمرینی یک پروژه در آن ایجاد نمائید.