

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
- 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

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.^۱ این نرم‌افزار، یکی از محصولات شرکت JetBrains است که امکان استفاده از نسخه Professional آن، برای دانشجویان به طور رایگان فراهم است.

^۱ 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.

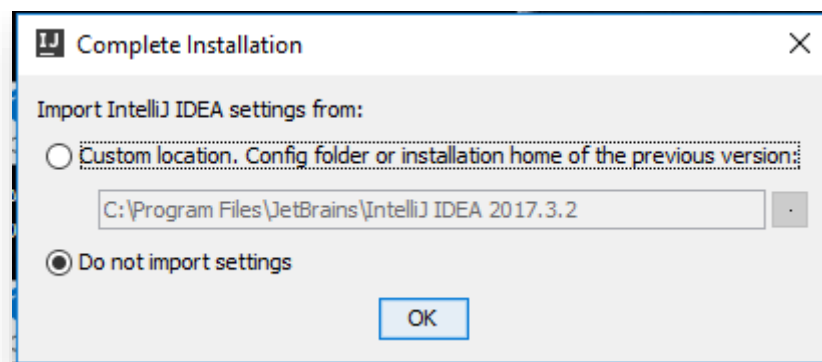
برای ایجاد یک حساب کاربری و راه‌اندازی نسخه Professional این محصول، لازم است به پایگاه اینترنتی <https://www.jetbrains.com/student> مراجعه کرده و مراحل کار را از آنجا دنبال کنید.

پس از ساخت حساب کاربری و فعال‌سازی آن، می‌توانید نسخه Professional این نرم‌افزار را از طریق وب یا سامانه اشتراک‌گذاری فایل دانشکده دریافت نمایید.

برنامه‌های نوشته‌شده به زبان جاوا، برای اجرا، نیازمند Java SDK^۲ هستند. برای دریافت SDK مناسب با سیستم‌عامل خود می‌توانید به آدرس <http://www.oracle.com/technetwork/java/javase/downloads/index.html> یا به سامانه اشتراک‌گذاری فایل دانشکده مراجعه نمایید. در رابطه با وظیفه SDK و نقش آن در فرایند اجرای برنامه‌های جاوا در ادامه این دستور کار بیشتر توضیح خواهیم داد.

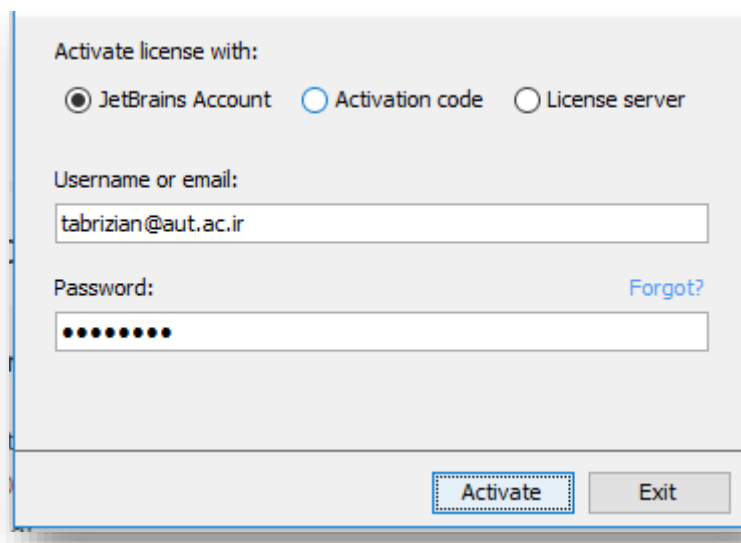
مراحل نصب IntelliJ IDEA

با اجرای فایل نصب نرم‌افزار، پنجره شکل ۱ نمایش داده می‌شود. در صورتی که قبلاً از نسخه دیگری از این نرم‌افزار استفاده می‌کردید، می‌توانید با انتخاب گزینه اول، تمامی تنظیمات اعمال شده 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

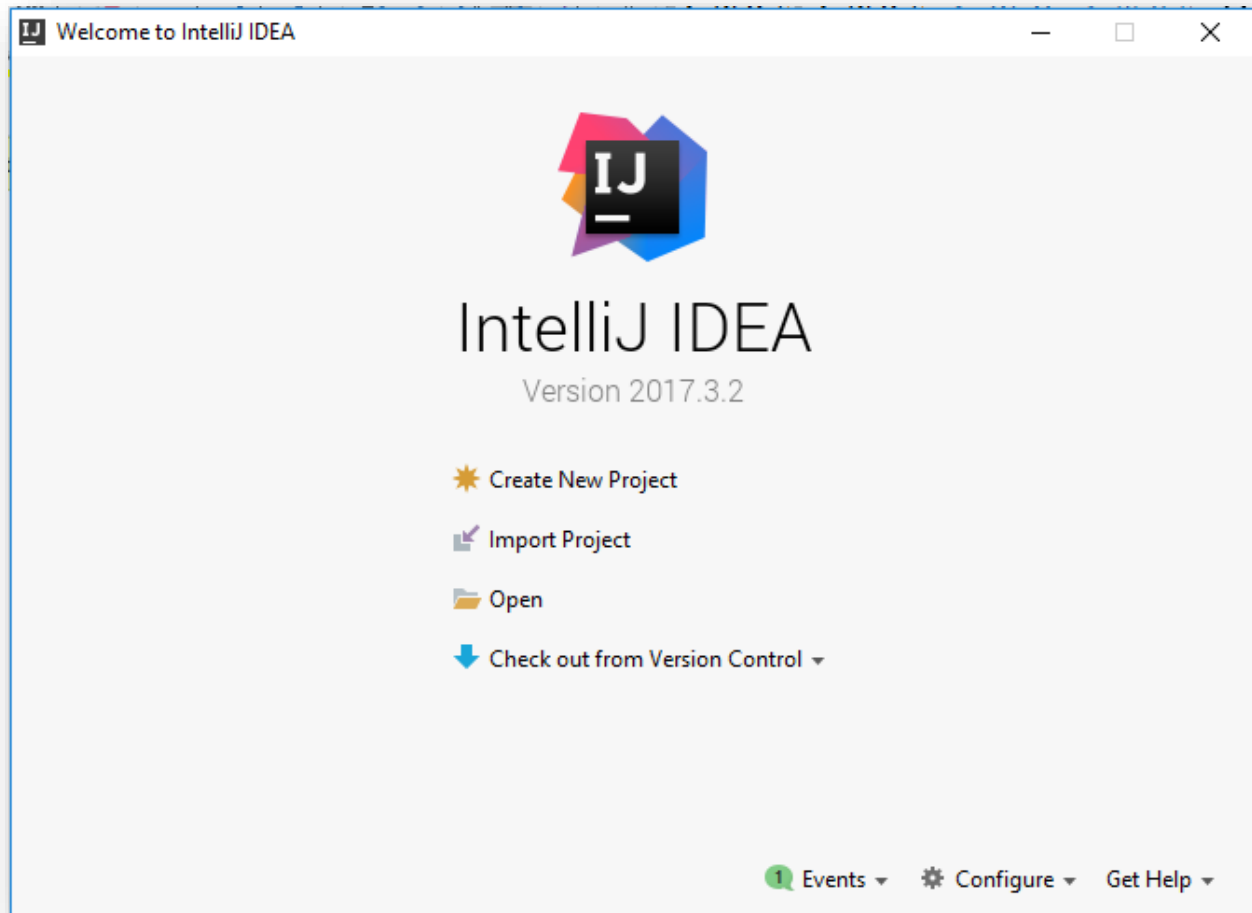
در مرحله بعدی، پنجره‌ای برای فعال‌سازی نرم‌افزار، مطابق شکل ۲، نمایش داده می‌شود. در این پنجره می‌توانید با انتخاب گزینه فعال‌سازی از طریق حساب کاربری و وارد نمودن نام و رمز عبور حساب کاربری خود، نرم‌افزار خود را فعال نمایید.



The image shows a 'License Activation' dialog box. At the top, it says 'Activate license with:'. Below this are three radio buttons: 'JetBrains Account' (which is selected), 'Activation code', and 'License server'. Underneath, there is a 'Username or email:' label followed by a text input field containing 'tabrizian@aut.ac.ir'. Below that is a 'Password:' label followed by a password input field with ten dots. To the right of the password field is a blue link that says 'Forgot?'. At the bottom of the dialog are two buttons: 'Activate' and 'Exit'.

شکل 2- فعال‌سازی نرم‌افزار

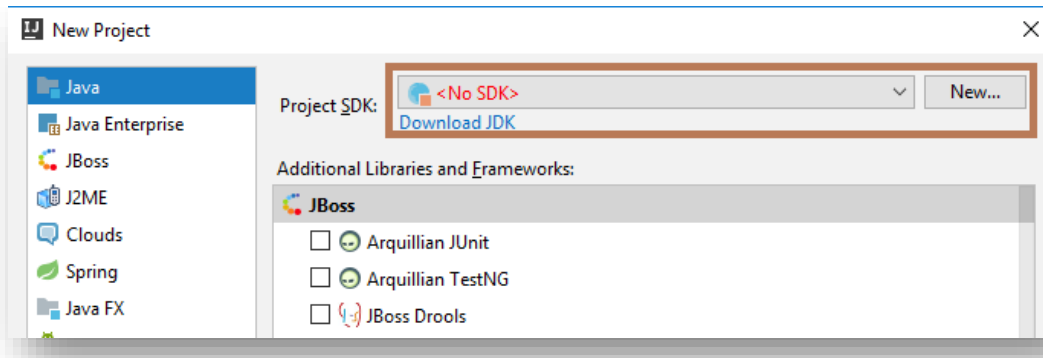
با اتمام فرایند نصب نرم‌افزار، پنجره شکل 3، نمایش داده می‌شود. در این مرحله، فرایند نصب نرم‌افزار تکمیل شده است و از این پس می‌توانید شروع به برنامه‌نویسی نمایید.



شکل 3- صفحه ایجاد پروژه جدید

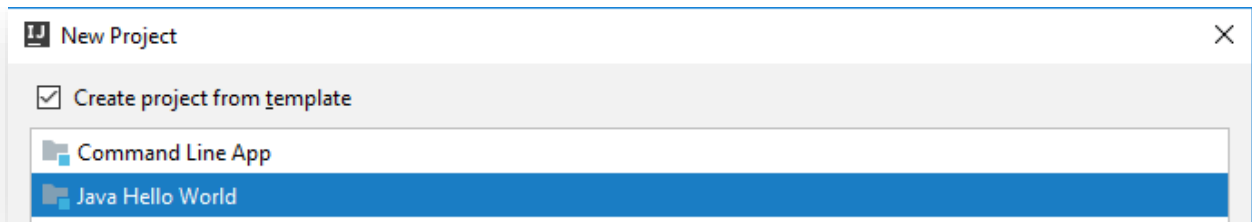
ایجاد پروژه جدید

با انتخاب گزینه **Create New Project** در پنجره شکل 3، می‌توانید یک پروژه جاوای جدید ایجاد نمایید. با انتخاب این گزینه، پنجره‌ای مشابه شکل 4، 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.



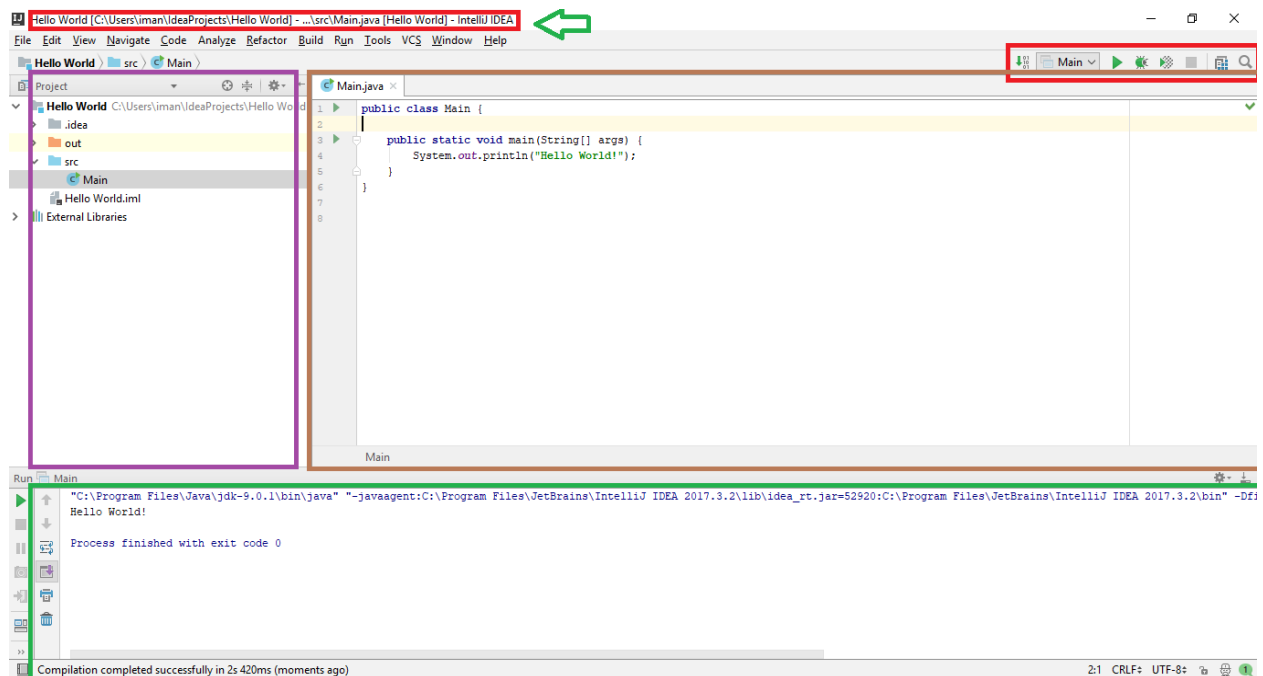
شکل ۴- راه اندازی SDK

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 ۵- 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^۳ نمایش داده می‌شود. میله ابزار در زیر میله عنوان، در بالای صفحه قرار دارد. کادر بنفش رنگ ساختار فایل پروژه، بسته‌ها و کلاس‌های تعریف شده را نمایش می‌دهد. کادر سبز رنگ 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 فولدر 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.

```
public class Main {

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

}
```

شکل ۷- کد برنامه Hello World

۳ Title Bar

تابع `main` نوشته شده در این فایل، اولین نقطه اجرای برنامه است. همان‌طور که در شکل 7 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 در رابطه با این دستور را مشاهده خواهید نمود.

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



شکل ۸ – منوی اجرای پروژه

از آنجایی که شما با مفاهیم آشنایی دارید ما در اینجا مثال‌هایی را آماده کردیم که نیازی به توضیح ندارند و صرفاً با مطالعه آن‌ها شما نحوه کار در جاوا را خواهید آموخت.

انواع داده در جاوا

جدول ۱، انواع داده‌های اولیه^۴ مورد استفاده در جاوا را به همراه اطلاعاتی در رابطه با آن‌ها نمایش می‌دهد.

– جدول انواع داده [جدول

Type	Description	Initial Value	Size	Example Literals
boolean	true or false	false	1 bit	true, false
byte	twos complement integer	0	8 bits	
char	Unicode character	\u0000	16 bits	'a', '\u0041', '\\'
short	twos complement integer	0	16 bits	
int	twos complement integer	0	32 bits	-2, -1, 0, 1, 2
long	twos complement integer	0	64 bits	-2L, -1L, 0L
float	IEEE 754 floating point	0.0	32 bits	1.23e100f, .3f, 3.14F
double	IEEE 754 floating point	0.0	64 bits	1.23456e300d, 1.234e-3d, 1e1d

۴ 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 ۸، نمونه‌ای از تعریف متغیر در جاوا را نمایش می‌دهد.

```
int a, b, c;
a = 1234;
b = 99;
c = a + b;
int[] arr = new int[50];
arr[0] = 0;
```

شکل ۸ – تعریف متغیر و آرایه

ساختارهای کنترلی در جاوا

قواعد نحوی ساختارهای کنترلی IF در جاوا، کاملاً مشابه با قواعد زبان C است. شکل ۹، 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 = 0;
        if (i == 0)
            System.out.println("i is 0");
        else
            System.out.println("i is not 0");
    }
}
```

شکل ۹ – استفاده از IF

شکل ۱۰، 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 = 4;
```



```

switch (day) {
    case 0: System.out.println("Sunday");    break;
    case 1: System.out.println("Monday");    break;
    case 2: System.out.println("Tuesday");   break;
    case 3: System.out.println("Wednesday"); break;
    case 4: System.out.println("Thursday");  break;
    case 5: System.out.println("Friday");    break;
    case 6: System.out.println("Saturday");  break;
    default: System.out.println("invalid day"); break;
}
}
}

```

شکل ۱۰ – 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 11 و 12، 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؟

```

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

        // print out special cases whose ordinal doesn't end in th
        System.out.println("1st Hello");
        System.out.println("2nd Hello");
        System.out.println("3rd Hello");

        // count from i = 4 to 10
        int i = 4;
        while (i <= 10) {
            System.out.println(i + "th Hello");
            i = i + 1;
        }

    }
}

```

شکل ۱۱ – ساختار while

```

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

```

```
// print out special cases whose ordinal doesn't end in th
System.out.println("1st Hello");
System.out.println("2nd Hello");
System.out.println("3rd Hello");

// count from i = 4 to 10
for (int i = 4; i <= 11; i++) {
    System.out.println(i + "th Hello");
}

}
```

شکل ۱۲ – 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 13، نمونه‌ای از استفاده از این ساختار را نمایش می‌دهد. لازم به ذکر است که For-Each تنها بر روی المان‌های قابل پیمایش قابل اجرا می‌باشد. 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);
}
```

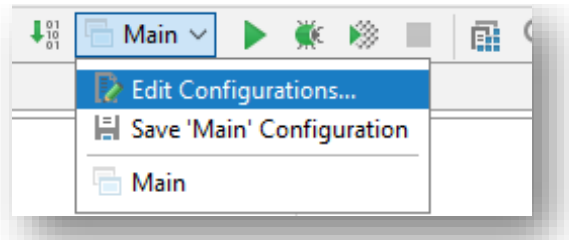
شکل ۱۳ – ساختار for-each

در شکل ۱۳ برای اینکه ما رشته را به نوعی قابل پیمایش تبدیل کنیم بایستی در گام اول آن را به آرایه‌ای از کاراکترها تبدیل کنیم 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

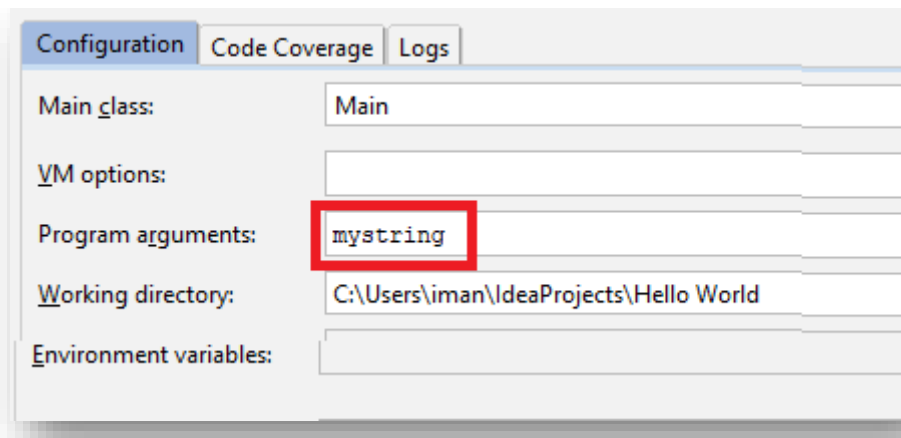
Argument ها متغیرهایی هستند که در زمان اجرا به برنامه پاس داده می‌شوند و رفتار برنامه را در زمان اجرا تغییر می‌دهند. به عنوان مثال فرض کنید به جای آن‌که شما در شکل ۱۳ تک تک حروف یک کلمه‌ی از پیش تعیین شده را چاپ کنید، قصد داشتید تا به عنوان رشته موردنظر را به عنوان یکی از Argument های ورودی دریافت کنید. لازم به ذکر است که این روش با روش‌هایی مانند scanf تفاوت دارد. در برنامه در حین اجرا درخواست ورودی از کاربر می‌کند در صورتی که در روش argument این متغیرها قبل از اجرای برنامه

دریافت شده اند و در آرایه‌ی `String[] args` قرار داده شده اند. حال برای مقداردهی این آرایه در محیط IntelliJ لازم است تا عملیات‌های زیر را انجام دهید.



شکل ۱۴ – تغییر پیکربندی اجرای برنامه

ابتدا مطابق با شکل ۱۴ گزینه `Edit Configurations` را انتخاب می‌کنید تا نحوه اجرای برنامه را تغییر دهید. پس از انجام این کار در پنجره‌ای که مطابق با شکل ۱۵ باز می‌شود عبارت `mystring` را Program arguments وارد کنید. این بخش شامل رشته‌هایی می‌شود که در آرایه‌ی `String[] args` قرار داده خواهد شد. کل عبارتی که در اینجا قرار می‌دهید با `space` جدا سازی می‌شود و به ترتیب در آرایه `args` قرار می‌گیرد. پس از انجام این تغییرات گزینه `Ok` را بزنید و از این صفحه خارج شوید.



شکل ۱۵ – تعیین Argument

حال برنامه شکل ۱۶ 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 ۱۶ – برنامه چاپ عناصر String با استفاده از Argument

تمرین بیشتر

برنامه‌ای بنویسید که دو عدد دریافت کند و بررسی کند آیا این دو عدد نسبت بهم اول هستند یا خیر.

برنامه‌ای بنویسید که یک جدول ۱۰ در ۱۰ را پیاده‌سازی کند.

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 گوشی‌های اندروید، سیستم‌عامل‌های لینوکس و موارد دیگر اجرا شوند.

زبان جاوا به دلیل داشتن یک ماشین مجازی به نام JVM^۵، قادر به ارائه چنین ویژگی مهمی است. از همین رو، آشنایی با ماشین مجازی جاوا و نقش آن در اجرای برنامه‌ها، از اهمیت بالایی برخوردار است.

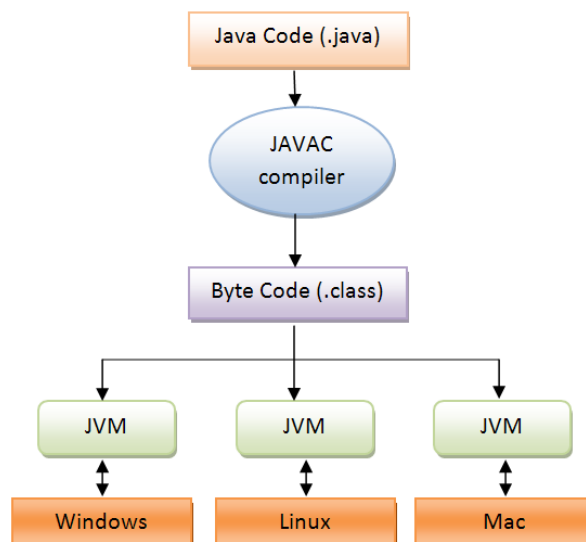
برنامه‌های نوشته شده در زبان جاوا، برخلاف برنامه‌های نوشته شده در زبان‌هایی مانند C، کامپایل و سپس اجرا نمی‌شوند. فرایند کامپایل این برنامه‌ها از دو بخش تشکیل شده است.

- کامپایل کردن متن اصلی برنامه به یک زبان میانی

- اجرای فایل زبان میانی توسط مفسر روی پلتفرم زمان اجرا

به این فرایند دو مرحله‌ای جاوا، Just in Time Compiling گفته می‌شود. 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 که برای مفسرهای جاوا قابل فهم هستند، تولید می‌نماید. این فایل‌های زبان میانی با پسوند class ذخیره می‌شوند. زبان میانی جاوا برای تمام مفسرهای جاوا روی هر پلتفرمی قابل فهم است. کافایت 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 15، این فرایند را به طور کامل نمایش می‌دهد.



شکل ۱۵ – Cross Platform بودن جاوا

JRE و JDK چیست؟

JRE مجموعه ابزارهایی است که برای اجرای اپلیکیشن‌های مبتنی بر جاوا استفاده می‌شود. این ابزار شامل JVM نیز می‌باشد. JRE مخفف Java Runtime Environment است. درواقع JRE تنها امکان اجرای نرم‌افزارهای جاوا را فراهم می‌کند. JDK علاوه بر ابزارهایی که JRE در خود دارد به شما این توانایی را می‌دهد که برنامه‌های جاوا خود را کامپایل و اشکال‌یابی کنید. درواقع JDK ابزارهای موردنیاز برای اشکال‌یابی و کامپایل را نیز در خود دارد.

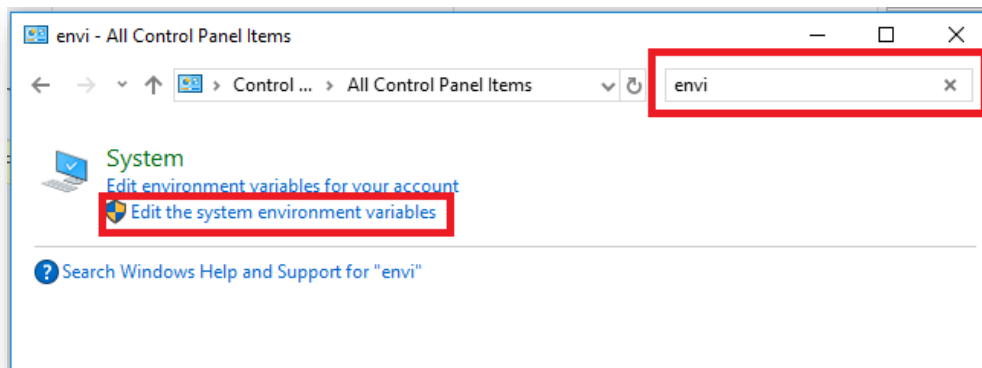
اجرای برنامه‌ها از طریق CLI

در بخش قبل، با نحوه اجرای برنامه در محیط توسعه یکپارچه آشنا شدیم. در این قسمت می‌خواهیم فرایند اجرای برنامه توسط JDK را بررسی نماییم. برای استفاده از JDK، کافی است متغیر محیطی^۶ PATH را به درستی تنظیم کنید تا به آن دسترسی داشته باشید.

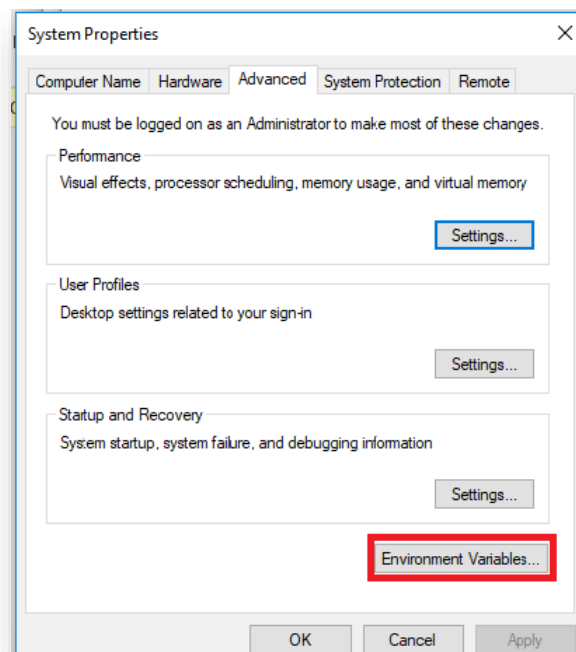
نحوه تنظیم متغیرهای محیطی در Windows 10, 8

۶ Environment Variable

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 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 ۱۶ – جست و جو در Control Panel



شکل ۱۷ – System Properties

اکنون سیستم‌عامل شما برای اجرای برنامه‌های مبتنی بر جاوا پیکربندی شده است. اگر پیکربندی شما صحیح باشد بایستی پس از اجرا کردن cmd.exe و تایپ java -version عبارت زیر را مشاهده کنید.

```
java version "9.0.1"
```

```
Java(TM) SE Runtime Environment (build 9.0.1+11)
```

```
Java HotSpot(TM) 64-Bit Server VM (build 9.0.1+11, mixed mode)
```

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

یک پوشه جدید در آدرس دلخواه خود ایجاد نموده و فایل Main.java مثال شکل ۱۶ 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.

```
mystring java Main
```

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

تمرین

برنامه‌ای بنویسید که با استفاده از ساختار For-Each، تمامی آرگومان‌های ارسال شده 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 می‌باشد که گاهی این مهارت از خود مهارت تولید برنامه مهم‌تر به شمار می‌رود. در این بخش، قصد داریم امکانات اشکال‌یابی IntelliJ را مورد بررسی قرار دهیم.

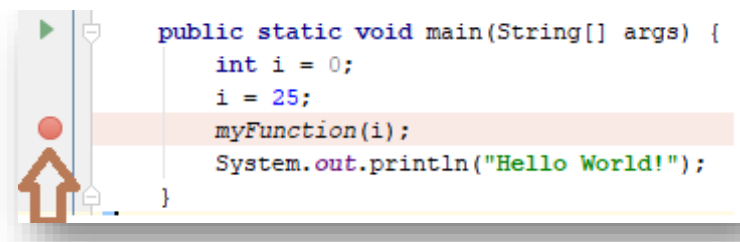
نرم‌افزار IntelliJ به برنامه‌نویسان این امکان را می‌دهد تا بتوانند برنامه خود را به صورت خط به خط، اجرا کرده و مقدار تمامی متغیرهای موجود را در هر لحظه مشاهده کنند. برای استفاده از این امکان، باید یک نقطه توقف^۷ در برنامه، قرار داده شود. پس از قراردادن این نقطه در برنامه، با اجرای برنامه در حالت اشکال‌یابی، 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. ابتدا در خط

```
File: Main.java
01: public class Main {
02:
03:     public static void myFunction(int i) {
04:         i++;
05:         System.out.println("The variable you passed was " + i);
06:     }
07:     public static void main(String[] args) {
08:         int i = 0;
09:         i = 25;
10:         myFunction(i);
11:         System.out.println("Hello World!");
12:     }
13: }
```

شکل ۱۸ – قطعه کد جهت اشکال‌یابی

شما فرض کنید در خط ۱۱ یک نقطه توقف قرار می‌دهید. برای قرار دادن نقطه توقف کافی است تا در ناحیه نشان داده شده را فشار دهید و یک علامت قرمز نشان داده خواهد شد.



شکل ۱۹ – قراردادن نقطه توقف

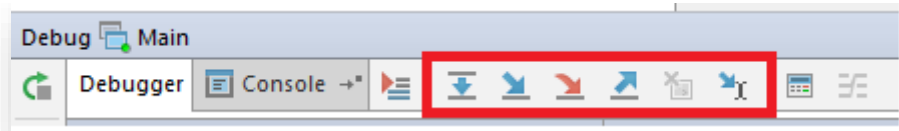
^۷ Break Point

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





شکل ۲۰ = اجرای برنامه در حالت اشکال‌یابی

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



شکل ۲۱ - نوار ابزار اشکال‌یابی

کلید میان‌بر	کارایی	دستور موجود در Tool Box
alt + F10	روی این دکمه کلیک کنید تا خط فعلی اجرای دستورات را نشان دهد.	Show Execution Point
F8	برنامه را تا خط بعدی اجرا کن و توابع بینابینی Skip کن.	Step Over
F7	روی این دکمه کلیک کنید تا به داخل تابعی که در حال حاضر Debugger روی آن قرار دارد برود.	Step Into
Shift + Alt + F7	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.	Force Step Into

Shift + F8	با اجرای این دستور Debugger از تابع فعلی خارج خواهد شد و به خط بعدی نقطه فراخوانی آن خواهد رفت.	Step Out 
Alt + F9	تا اجرای دستور در نقطه‌ی مکان‌نما در Editor پیش خواهد رفت.	 Run to Cursor

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

نکاتی درباره برنامه‌نویسی در جاوا

- جاوا یک زبان برنامه‌نویسی حساس به متن^۸ است.
- طبق قرارداد، اسامی متدها باید با حرف کوچک شروع شود. اگر اسم متد از چند کلمه تشکیل شده است، باید اولین حرف کلمه داخلی بزرگ نوشته شود مانند قطعه کد زیر:

```
public void myMethod()
```

- اسم فایل باید حتماً 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. (با توجه به توصیه‌های بالا و مطالب گفته شده در اول جلسه)

قطعه کد اول:

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

^۸ Case Sensitive

قطعه کد دوم:

```
public class main {  
  
    public void myanothermethod () {  
    }  
  
}
```

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

الف) ;

ب) اولین “

ج) دومین “

د) اولین }

ه) دومین {

و) اولین }

ز) دومین {

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 می‌باشد. مفسری یا کامپایلری؟ توضیح دهید.