**به نام خدا**

## آموزش پنجاه و ششم

اهداف آموزشی این قسمت عبارتند از:

۱. معرفی انواع Exception ها در زبان جاوا
۲. آشنایی با مفاهیم Compile-time و Runtime
۳. آشنایی با NullPointerException
۴. آشنایی با `NumberFormatException`
۵. آشنایی با `ArrayIndexOutOfBoundsException`

به طور کلی Exception ها را می توان به دو گروه اصلی Checked و Unchecked دسته بندی کرد. در واقع منظور از Checked Exception ها Exception هایی هستند که در حین Compile-time رخ می دهند که از آن جمله می توان به IOException و SQLException اشاره کرد. در واقع این دست از Error ها مشکلاتی هستند که به Syntax برنامه مرتبط هستند. مثلاً اگر به جای نوشتن int عبارت INT را بنویسیم، از آنجا که یک مشکل Syntax یی داریم بایستی انتظار مشکلی مرتبط با Compile time داشته باشیم. به طور کلی این دسته از Exception ها "حتماً" می بایست اصطلاحاً Handle شده یا مد نظر قرار داده شوند. منظور از Unchecked Exception ها نوعی از Exception ها است که در حین Runtime رخ می دهندکه از آن جمله می توان به ArithmeticException و NullPointerException اشاره نمود. در واقع در این دست از مشکلات برنامه بدون هیچ مشکلی Compile می شود اما این در حالی است که در حین اجرای برنامه کاربر نتایج غیر قابل انتظاری مشاهده خواهد کرد و حتی ممکن است برنامه Crash هم بکند.

در حقیقت منظور از Compile-time این است که زمانیکه ما یک برنامه در زبان برنامه نویسی جاوا می نویسیم، کدهای ما می بایست به زبان ماشین تبدیل شوند و زمانیکه کدها به زبان ماشین تبدیل می شوند Compile-time نامیده می شود. حال پس از آنکه کدها به زمان ماشین تبدیل

شدند و اصطلاحاً برنامه Compile شد، کاربر می تواند برنامه را اجرا کند و به زمانیکه کاربر یک برنامه را اجرا می کند Runtime گفته می شود.

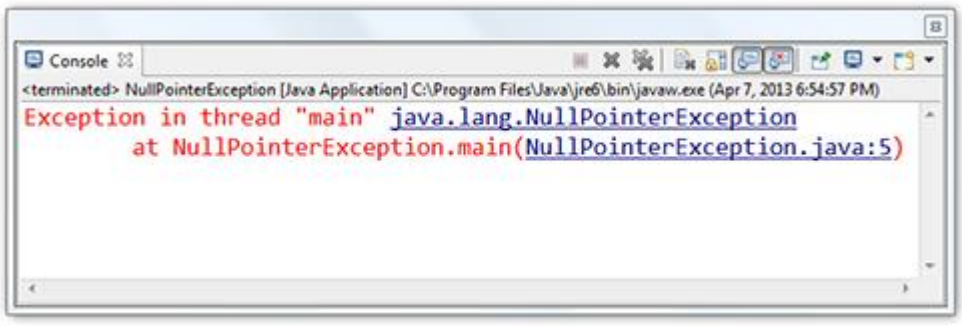
پس از آنکه با مفاهیم Compile-time و Run-time در زبان برنامه نویسی جاوا آشنا شدیم در این بخش از آموزش به منظور بررسی دیگر انواع Exception ها کلاس های دیگری به پروژه خود اضافه می کنیم:

```
public class NullPointerException {

	public static void main(String[] args) {
		String s = null;
		System.out.println(s.length());

	}
}
```

همانطور که در کد فوق می بینیم کلاسی تحت عنوان NullPointerException ایجاد کرده ایم. سپس در متد main این کلاس یک شیئ از روی کلاس String تحت عنوان s ساخته ایم و مقدار اولیه آن را برابر با null قرار داده ایم. حال در دستور System.out.println با استفاده از متد length قصد داریم تعداد کاراکتر های شیئ ساخته شده از روی کلاس String را شمارش کنیم. برای همین منظور داخل پرانتز مرتبط با متد println نام شیئ ساخته شده از روی کلاس String را نوشته سپس یک نقطه قرار داده و متد length را به آن ضمیمه می کنیم. اکنون می توانیم برنامه خود را اجرا کنیم:




همانطور که در تصویر فوق می بینیم پس از اجرای برنامه با یک NullPointerException مواجه می شویم. علت مواجهه با چنین Exception یی این است که برای Object های ایجاد شده از روی کلاس String می بایست مقداری همچون یک عبارت یا یک کلمه در نظر گرفت

که در این صورت اگر از متد length هم استفاده کنیم این متد تعداد کاراکترهای مرتبط با شیئ ساخته شده از روی کلاس String را خواهد شمارد اما از آنجا که در مثال فوق مقدار اولیه این کلاس را برابر با null قرار داده ایم حال اگر بخواهیم متد length را به شیئی ضمیمه کنیم که مقدار اولیه آن null است، برنامه ما با Exception یی از جنس NullPointerException رو به رو خواهد شد. چنانچه بخواهیم عبارت NullPointerException را به صورت تحت الفظی ترجمه کنیم می توانیم معادل "مشکلی که به خاطر اشاره به چیزی که تهی است ایجاد شده است" را در نظر بگیریم.

حال کلاس دیگری تحت عنوان NumberFormatException ایجاد می کنیم که از طریق آن Exception یی با نام NumberFormatException را مورد بررسی قرار خواهیم داد:

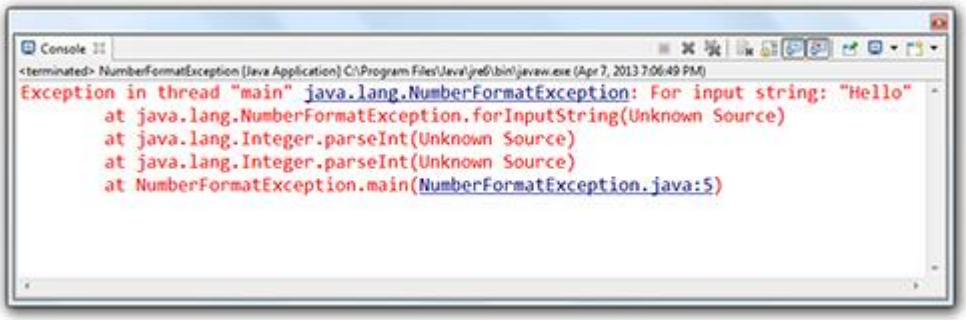
```
public class NumberFormatException {

	public static void main(String[] args) {
		String s = "Hello";
		int i = Integer.parseInt(s);

	}
}
```

همانطور که در کد فوق مشخص است شیئی تحت عنوان s از روی کلاس String ایجاد کرده و مقدار اولیه آن را برابر با Hello قرار می دهیم. سپس یک متغیر از جنس int تحت عنوان i ایجاد می کنیم و مقدار آن را برابر با متدی تحت عنوان parseInt که به کلاسی تحت عنوان Integer ضمیمه شده است قرار داده و شیئ ساخته شده از روی کلاس String را به عنوان پارامتر متد parseInt در نظر می گیریم.

همانطور که در آموزش های گذشته توضیح داده شد، به منظور تبدیل متغیرها به یکدیگر می توان از کلاس ها و متدهای مرتبط با آنها استفاده کرد. حال برنامه را اجرا می کنیم:




در واقع علت بروز چنین Exception یی این است که به هیچ وجه نمی توانیم یک شیئ از جنس کلاس String را به متغیری از جنس int تبدیل کنیم.

Exception دیگری که می خواهیم مورد بررسی قرار دهیم ArrayIndexOutOfBoundsException نام دارد. برای این منظور کلاسی با همین نام در پروژه خود ایجاد می کنیم و آن را به صورت زیر تکمیل می کنیم:

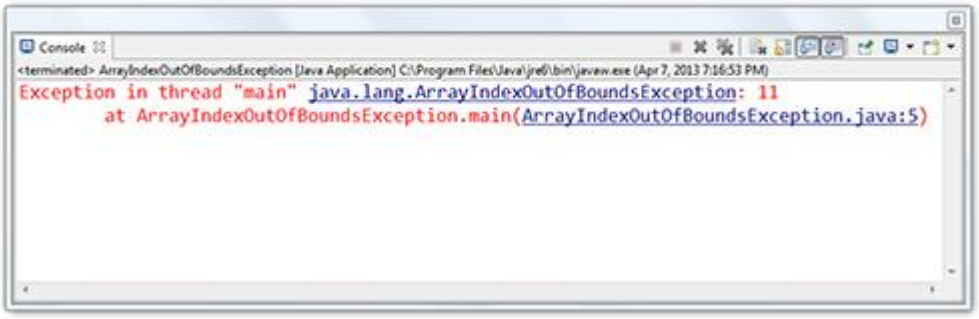
```
public class ArrayIndexOutOfBoundsException {

	public static void main(String[] args) {
		int[] numbers = new int[10];
		numbers[11] = 100;

	}

}
```

همانطور که در کد فوق می بینیم یک Array از جنس int ایجاد کرده ایم که numbers نام دارد. همانطور که مشخص است این Array قرار است ۱۰ گزینه را در خود جای دهد. پس از تعریف این Array در خط دوم می بینیم که Array یی با شماره ۱۱ را مد نظر قرار داده ایم و مقدار آن را برابر با عدد ۱۰۰ در نظر گرفته ایم. حال برنامه را اجرا می کنیم:



همانطور که در تصویر فوق می بینیم برنامه ما با یک Exception از جنس ArrayIndexOutOfBoundsException رو به رو می شود و علت هم آن است که ما در Array خود فقط ده گزینه داریم اما در ادامه برنامه گزینه شماره یازده را هدف قرار داده ایم و از آنجا که این گزینه خارج از محدوده تعریف شده برای این Array است با چنین Exception یی رو به رو خواهیم شد.

پس از مطالعه این آموزش انتظار می رود بتوانیم به سؤالات زیر پاسخ بدهیم:

۱. چند نوع Exception در زبان جاوا وجود دارد؟
۲. برای هر یک از انواع Exception ها چند مثال بزنید؟
۳. منظور از Compile-time چیست؟
۴. تفاوت Compile-time با Runtime چیست؟