مدیریت استثناها (Exception-Handling)

یادآوری: انواع خطاها در یک برنامه ←
① خطای نحوی: در زمان کامپایل مشخص می‌شود (قبل از اجرا) کار متوقف شود و برنامه اجرا نمی‌شود.
② خطای معنایی: که برنامه اجرا می‌شود ولی خروجی مورد نظر بدست نمی‌آید
③ خطای زمان اجرا: که برنامه به درستی کامپایل می‌شود ولی در اساس سمکدیو در حین اجرا برنامه، برنامه پایان ناپذیرد.

در ادامه بحث بر آن است تا بروی استثناها که حاصل از خطای زمان اجرای برنامه است و استریج و جلوی خروج ناگهانی از برنامه را بگیریم.

قالب برای مدیریت استثناها:

```
try {
  // code block
}
catch (ExceptionType name) {
  // code block
}
finally {
  // code block }
```


2019 شهریور ۹۸ Sep 6

۱۴۴۱ جمعه ۶ محرم

بلاک try : کدهایی که ممکن است حاوی خطا باشد در این بلاک قرار می‌گیرد.

بلاک catch : این قسمت اجرا می‌شود اگر خطایی در بلاک try اتفاق بیفتد. لازم به ذکر است که این بلاک اجرا می‌شود اگر خطای اتفاق افتاده در try و یا از نوع ExceptionType معرفی شده در بلاک catch باشد.

بلاک finally : در صورت وجود / عدم وجود خطا (در هر صورت) بلاک finally اجرا می‌شود.

برخی از انواع استثناهای رایج:

① ArithmeticException : این کلاس برای هندل کردن استثناهای مربوط به محاسبات ریاضی در نظر گرفته شده است، مثل تقسیم بر صفر.

② NullPointerException : این کلاس برای هندل کردن استثناهای مربوط به مقدار Null و متدهای روی آن، در نظر گرفته شده است، مثل برگرداندن طول رشته Null.

③ NumberFormatException : این کلاس برای هندل کردن استثناهای مربوط به فرمت نادرست رشته‌ای اعداد، در نظر گرفته شده است، مثل تبدیل یک رشته حاوی کاراکترهای غیر عددی به عدد صحیح.

۱۴) ArrayIndexOutOfBoundsException : این کلاس برای مدیریت کردن استثناهایی مربوط به دسترسی به عناصر آرایه مورد استفاده قرار می‌گیرد، مثل دسترسی به عنصری از آرایه که خارج از محدوده آرایه است.

نکته ۱: در صورتی که بخواهیم هر استثنایی را مدیریت کنیم می‌توانیم به جای ExceptionType از کلاس Exception استفاده کنیم.

نکته ۲: می‌توانیم چندین catch برای یک try بنویسیم که استثناهای نوع خاص مقدم‌تر را پوشش دهیم.

```
try {
  // code block
}
catch (ExceptionType1 e1)
{
  // code block
}
:
catch (ExceptionTypen en)
{
  // code block
}
finally {
// code block
}
```

۱۸ | ۱۴۴۱ دوشنبه ۹ محرم | 2019 شهریور ۹۸ Sep 9

مثال:

```
try {
    int num1=24;
    int num2= 0;
    int result = num1/num2;
    System.out.println(result);
}
catch ( Exception e) {
    System.out.println("Error :" + e.getMessage());
}
finally {
    system.out.println("The End ...");
}
```

تغییر کد بالا و بررسی تک تک استثناها: ① بررسی multi catch

② بررسی NullPointerException

③ بررسی NumberFormatException

④ بررسی ArrayIndexOutOfBoundException

کار با فایل های متنی در جاوا:

فایل ها به عنوان یک بخش مهم از هر برنامه کاربردی برای ذخیره و بازیابی داده ها شناخته می شوند.

در ادامه قصد داریم چندین متود برای کار با فایل ها تعریف: ایجاد کردن، خواندن، بروزرسانی و حذف را بررسی کنیم.

برای کار با فایل، می بایست ابتدا کلاس File را import کنیم:

```
import java.io.File;
```

برای تعریف یک آبجکت از کلاس File می بایست از دستور زیر استفاده کنیم:

```
File fileObjectName = new File("filePath");
```

جایگزین fileName نام آبجکت برای کار با فایل و filePath رشته تعیین کننده مسیر فایل / پوشه می باشد.

برخی از متودهای پرکاربرد کار با فایل ها:

1. ایجاد یک پوشه (یا دایرکتوری):

```
fileObjectName.mkdir();
```

مثال:

```
String PathDirectory = "/home/majtaba/TestFolder";

File MyFile = new File(PathDirectory);

if (myFile.mkdir()) {
    System.out.println("Folder created successfully.");
}
else {
    System.out.println("Folder could not be created");
}
```

ایجاد یک فایل خالی:

```
FileObjectName.createNewFile();
```

نکته: دستور ایجاد فایل لازم است حتماً در یک بلاک try...catch تعریف شود و در غیر اینصورت با خطا مواجه می‌شود.

مثال:

```
String PathFile = "/home/majtaba/TestFolder/TestFile.txt";

File myFile = new File(PathFile);

try {
    if (myFile.createNewFile()) {
        System.out.println("File created successfully");
    } else {
        System.out.println("File could not be created."); }
}
```