

Exceptions in Java

الاستثناءات في الجافا

Dr. REEMA AL-KAMHA

مقدمة

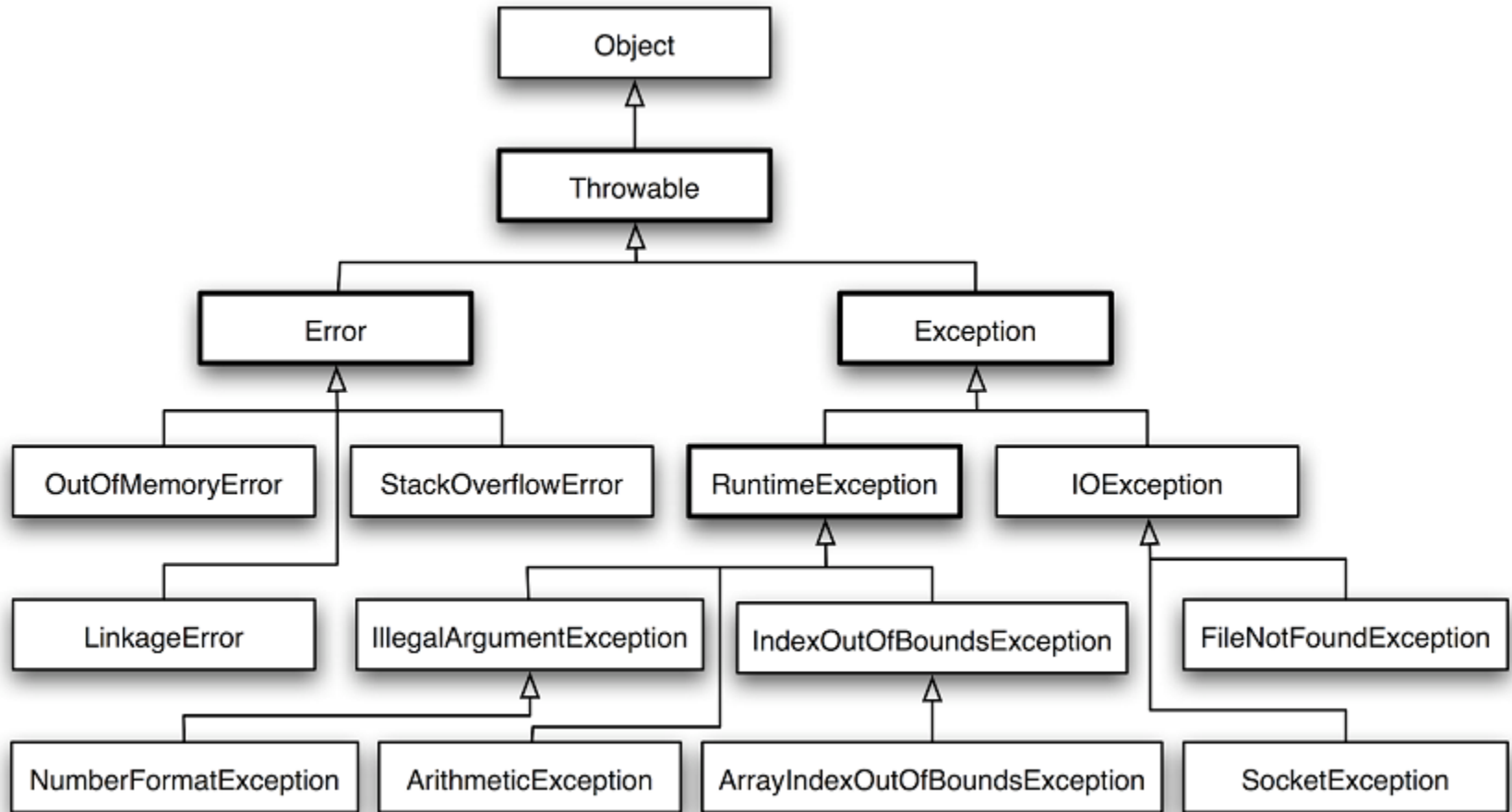
- يمكن للبرنامج في بعض الحالات أن ينفذ تعليمات غير مسموحة، و بالتالي يقع في حالة استثنائية يمكن أن تؤدي إلى توقف عمله.
- **الاستثناء هو حدث غير مرغوب به أو غير متوقع يحدث أثناء تنفيذ البرنامج فيؤدي إلى إيقافه.**
- **أمثلة على حالات تحدث استثناء:**
 - القسمة على صفر هي إحدى الحالات التي تحدث استثناء يؤدي إلى توقف البرنامج.
 - محاولة التعامل (فتح أو الكتابة) مع ملف غير موجود.
- من المهم جدا للمبرمج أن ينتبه إلى إمكانية حصول الاستثناءات و معالجتها إذا أراد لبرنامجها أن يكون فعالا.
- **تسمح معالجة الاستثناءات بكتابة تطبيقات تستمر في عملها بعد حدوث الاستثناءات مما يجعل هذه التطبيقات أكثر مرونة.**

مقدمة (تتمة)

- توفر لغة الجافا أداة لالتقاط الاستثناءات عند حصولها و تحويلها إلى المقطع البرمجي المناسب لمعالجتها.

- بفضل هذه الأداة يستطيع المبرمج وضع مقاطع البرنامج التي يمكن أن تولد استثناء تحت المراقبة و توجيه التحكم إلى مقاطع برمجية لمعالجة الاستثناءات إذا حدثت، و بالتالي يتم رفع من الوثوقية في البرنامج من خلال منعه من التوقف عن العمل.

Exception Hierarchy هيكلية الاستثناء



الصف Exception هو صف الاستثناء الأساسي، و الذي بدوره يرث من الصف Throwable. كل صف يرث من الصف Exception يمثل صف استثناء من نوع معين.

بعض الاستثناءات الشهيرة التي توفرها لغة الجافا

❖ **ArithmeticException** يولد عندما يقوم مقطع برمجي بعملية حسابية غير مسموحة مثل القسمة على صفر.

❖ **NullPointerException** يولد عندما يطلب الحصول على معلومات من غرض غير موجود.

❖ **ArrayIndexOutOfBoundsException** يولد عندما يتم تجاوز عدد عناصر متجه.

❖ **FileNotFoundException** يولد عندما يحاول مقطع برمجي التعامل مع ملف غير موجود.

❖ **ClassCastException** يولد عندما يحاول مقطع برمجي التحويل بين الأنواع casting بشكل غير صحيح.

❖ **IllegalArgumentException** يولد عندما تتلقى طريقة معينة قيمة (أو قيم) غير مناسبة لوسطائها.

❖ **InputMismatchException** يولد عندما يدخل المستخدم قيمة من نوع مخالف للنوع المطلوب.

مثال محاولة القسمة على صفر تولد استثناء

```
public class Ex13 {  
    public static void main(String[] args) {  
        int num = 100/0; // Arithmetic Exception  
        System.out.println(num);  
    }  
}
```

النتيجة:

```
Exception in thread "main"  
java.lang.ArithmeticException: / by zero  
    at ex.pkg13.Ex13.main(Ex13.java:19)
```

مثال محاولة تجاوز عدد عناصر متجه تولد استثناء

```
public class Ex16 {  
    public static void main(String[] args) {  
        int []s=new int[2];  
        s[0]=1;  
        s[1]=3;  
        for(int i=0; i<=s.length;i++)  
            System.out.println(s[i]); //ArrayIndexOutOfBoundsException  
    }  
}
```

النتيجة:

1

3

Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 2
 at ex.pkg16.Ex16.main

مثال عند إدخال المستخدم قيمة غير صحيحة يتولد استثناء

```
public class Ex18 {  
    public class Main {  
        public static void main(String[] args) {  
            Scanner sc=new Scanner(System.in);  
            int i=sc.nextInt();  
        }  
    }  
}
```

النتيجة:

عندما يدخل المستخدم القيمة 2.3
يتولد الاستثناء التالي:

Exception in thread "main" java.util.**InputMismatchException**

مثال عندما يطلب الحصول على معلومات من غرض غير موجود
يتولد استثناء

```
public class Ex20{  
    public class Main {  
        public static void main(String[] args) {  
            String s=null;  
            System.out.println(s.length());  
        }  
    }  
}
```

النتيجة:

يتولد الاستثناء التالي:

Exception in thread "main" java.lang.**NullPointerException**

معالجة الاستثناء

Exception Handling

التعليمة try-catch

□ الصيغة العامة لاستخدام هذه التعليمة:

```
try{  
    //statements  
}  
catch(ExceptionType name){  
    تعليمات لمعالجة الاستثناء//  
}
```

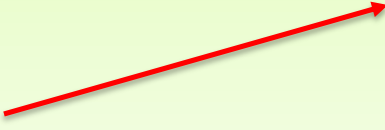
□ عند وضع مجموعة من التعليمات (statements) **ضمن العبارة try** فإن البرنامج يقوم بمحاولة تنفيذ هذه التعليمات. في حال فشل تنفيذ تعليمة معينة يحصل نوع من الاستثناءات (ExceptionType) و ينتقل التحكم مباشرة إلى داخل العبارة **catch** لمعالجة الاستثناء. يستطيع المبرمج أن يكتب ضمن العبارة **catch** أي تعليمات لمعالجة الاستثناء.

□ إضافة إلى ذلك، يمكن للمبرمج أن يعرف أي عدد من الاستثناءات التي يريد معالجتها و ذلك عن طريق تكرار العبارة **catch**.

مثال على معالجة استثناء القسمة على صفر

```
public class Ex17 {  
    public static void main(String[] args){  
        try{  
            System.out.println(100/0);  
        }  
        catch(Exception e){  
            System.out.println(e);  
        }  
        System.out.println("The End");  
    }  
}
```

يتم طباعة اسم الاستثناء ونوعه



النتيجة:

```
java.lang.ArithmeticException: / by zero  
The End
```

مثال على معالجة استثناء القسمة على صفر

```
public class Ex17 {  
    public static void main(String[] args){  
        try{  
            System.out.println(100/0);  
        }  
        catch(Exception e){  
            System.out.println(e.getMessage());  
        }  
        System.out.println("The End");  
    }  
}
```

الطريقة getMessage() تعيد نوع الاستثناء



النتيجة:

/ by zero
The End

مثال على معالجة استثناء عند محاولة تجاوز عدد عناصر متجه

```
public class Ex17 {  
    public static void main(String[] args){  
        int []s=new int[2];  
        s[0]=1;  
        s[1]=3;  
        try{  
            for(int i=0; i<=s.length;i++){  
                System.out.println(s[i]);  
            }  
        } catch(Exception e){  
            System.out.println(e);  
        }  
        System.out.println("The End");  
    }  
}
```

في الصف أعلاه نجد أن هناك تجاوزاً لحدود المتجه بسبب شرط التوقف في الحلقة for. (يجب أن يكون شرط التوقف $i < s.length$ و **بالتالي** بغياب التعليمة try-catch فإن البرنامج سيتوقف كلياً. أما مع وجود التعليمة try-catch فإن البرنامج لا يتوقف عند حدوث تجاوز حدود المتجه بل ينتقل التحكم إلى العبارة catch و التي تقوم بمعالجة الخطأ و استمرار عمل البرنامج.

النتيجة:

```
1  
3  
java.lang.ArrayIndexOutOfBoundsException: 2  
The End
```

مثال على معالجة استثناء عند إدخال المستخدم قيمة غير صحيحة

```
public class Ex17 {  
    Import java.util.*;  
    public static void main(String[] args){  
        System.out.print("Enter an Integer number= ");  
        Scanner sc=new Scanner(System.in);  
        try{  
            int i=sc.nextInt();  
        }  
        catch(Exception e ){  
            System.out.println(e);  
            System.out.println("You should enter an Integer number");  
        }  
        System.out.println("the end");  
    }  
}
```

النتيجة:

```
Enter an Integer number= 3.7  
java.util.InputMismatchException  
You should enter an Integer number  
the end
```

الشكل العام لتعليمة try-catch

```
try{  
    //statements  
}  
catch(NumberFormatException e){  
    تعليمات لمعالجة الاستثناء//  
}  
catch(IOException e){  
}  
catch(Exception e){  
}  
finally{  
}
```

عند حدوث استثناء ضمن التعليمة try تتم المعالجة من خلال عبارة واحدة أو أكثر عدة عبارات catch كل منها تحوي استثناء مختلف عن الآخر، وقد يتم وضع عبارة finally واحدة في الأخير.

هذه الاستثناءات

- NumberFormatException أي الرقم مكتوب بشكل خاطئ.
- IOException أي حصل خطأ في التعامل مع وحدات الدخل أو الخرج (مثلا محاولة فتح ملف غير موجود).
- Exception هو صف عام يمكن استخدامه مع أي استثناء.
- العبارة finally هي اختيارية و تنفذ دائما حتى و إن لم يطابق الاستثناء الحاصل أيا من الاستثناءات في catch .

مثال على تكرار العبارة catch

```
public class EX30 {
    public static void main(String[] args){
        try{
            System.out.println("Hello There");
            String s=null;
            System.out.println(s.length());
        }
        catch(ArithmeticException e){
            System.out.println(e);
        }
        catch(NullPointerException e ){
            System.out.println(e);
        }
        catch(Exception e) {
            System.out.println(e);
        }
        System.out.println("The End");
    }
}
```

هنا يحدث استثناء

يتم معالجة الاستثناء ضمن المقطع catch المناسب له، و هو المقطع الثاني

يوضح هذا المثال عملية حدوث استثناء و معالجته بعدة عبارات catch ، كل منها تحوي استثناء مختلف عن الآخر.

هذه الاستثناءات

➤ ArithmeticException أي القيام بعملية

حسابية غير مسموحة مثل القسمة على صفر.

➤ NullPointerException أي طلب

الحصول على معلومات من غرض غير

موجود.

➤ Exception هو صف عام يمكن استخدامه

مع أي استثناء.

النتيجة:

Hello There
java.lang.NullPointerException
The End

مثال على تكرار العبارة catch

عندما يحوي المقطع try على أكثر من استثناء، يتم معالجة الاستثناء الذي يحدث أولاً، وذلك ضمن المقطع catch المناسب لذلك الاستثناء.

```
public class EX24 {  
    public static void main(String[] args) {  
        try {  
            System.out.println("Hello There");  
            String s=null;  
            System.out.println(s.length());  
            System.out.println(100/0);  
        }  
        catch (ArithmeticException e) {  
            System.out.println(e);  
        }  
        catch (NullPointerException e ) {  
            System.out.println(e);  
        }  
        System.out.println("The End");  
    }  
}
```

الاستثناء الأول

الاستثناء الثاني

يتم هنا معالجة الاستثناء الأول

النتيجة:

عندما يحدث الاستثناء الأول ينتقل التنفيذ مباشرة لمعالجته ضمن المقطع catch الثاني المناسب له

Hello There

java.lang.NullPointerException

The End

مثال على تكرار العبارة catch

```
public class EX26 {  
    public static void main(String[] args){  
        try{  
            System.out.println("Hello There");  
            System.out.println(100/0);  
            String s=null;  
            System.out.println(s.length());  
        }  
        catch(ArithmeticException e){  
            System.out.println(e);  
        }  
        catch(NullPointerException e ){  
            System.out.println(e);  
        }  
        catch(Exception e ){  
            System.out.println(e);  
        }  
        System.out.println("The End");  
    }  
}
```

النتيجة:

```
Hello There  
java.lang.ArithmeticException: / by zero  
The End
```

مثال على تكرار العبارة catch

عند تكرار العبارة catch في معالجة الاستثناء، يجب المحافظة على ترتيب الاستثناءات من الخاص إلى الأعم، و إلا لن يترجم البرنامج

```
public class EX28 {
    public static void main(String[] args){
        try{
            System.out.println("Hello There");
            System.out.println(100/0);
            String s=null;
            System.out.println(s.length());
        }
        catch(Exception e) {
            System.out.println(e);
        }
        catch(ArithmeticException e){
            System.out.println(e);
        }
        catch(NullPointerException e ){
            System.out.println(e);
        }

        System.out.println("The End");
    }
}
```

الاستثناء العام Exception يجب أن يكون في مقطع catch الأخير

Compile-time error

مثال على العبارة finally

```
public class EX30 {  
    public static void main(String[] args){  
        try{  
            System.out.println("Hello There");  
            String s=null;  
            System.out.println(s.length());  
        }  
        catch(ArithmeticException e){  
            System.out.println(e);  
        }  
        catch(NullPointerException e ){  
            System.out.println(e);  
        }  
        finally{  
            System.out.println("finally block is always executed");  
        }  
        System.out.println("The End");  
    }  
}
```

Hello There

java.lang.NullPointerException

finally block is always executed

The End

النتيجة:

توليد الاستثناءات في طريقة

- يقدم المثال التالي توضيحا لكيفية توليد استثناء في طريقة معينة.

```
public class HandlingExceptionviaMethod {  
    public void a(int n)throws Exception{  
        if(n==0){  
            throw new Exception("dividing by Zero");  
        }  
        else  
            System.out.println(100/n);  
    }  
}
```

كما تلاحظ من الطريقة أعلاه، و بشكل خاص الجزء المكتوب باللون الأحمر، فإن إحدى طرق توليد استثناء هي:

➤ التصريح عند نهاية ترويسة الطريقة بأن هذه الطريقة تولد استثناء (العبارة `throws`)، ثم تحديد نوع الاستثناء الذي تولده الطريقة. في مثالنا: `Exception`

➤ ثم نضع العبارة التي تولد ذلك الاستثناء في المكان الذي سوف يقع فيه الاستثناء (إن وقع):

```
Throw new Exception("dividing by Zero");
```

استدعاء الطريقة a

```
public class MethodException {  
    public static void main(String[] args) {  
        HandlingExceptionviaMethod hm=new HandlingExceptionviaMethod();  
        try{  
            hm.a(0);  
        }  
        catch(Exception e){  
            System.out.println(e);  
        }  
    }  
}
```

النتيجة:

java.lang.Exception: dividing by Zero