الاستثناءات Exceptions

مفهوم الاستثناءات في جافا

استثناءات: تعني Exceptions في البرمجة، والاستثناء عبارة عن خطأ يحدث أثناء تشغيل البرنامج يؤدي إلى إيقافه بشكل غير طبيعي. وهذا أمر سيئ جداً لأنه يؤدي إلى نفور عدد كبير من المستخدمين وعدم رغبتهم في العودة إلى استخدام هذا البرنامج مجدداً.

كيف يمكن ان تتجنب حدوث أخطاء في البرامج التي تكتبها، و فعلياً ستتعلم كيف تلتقط هذه الأخطاء في حال حدثت لجعل البرنامج شغال دائماً في نظر المستخدم و لا يظهر له أي أخطاء.

بعض الأسباب التي تسبب حدوث الاخطاء

- في حال إدخال قيمة لا تتطابق مع نوع المتغير الذي ستخزن فيه.
- في حال إدخال رقم **index** غير موجود في مصفوفة أو في متغير نوعه String.
- في حال كان البرنامج يتصل بالشبكة وفجأةً انقطع الاتصال.
- في حال كان البرنامج يحاول قراءة معلومات من ملف نصي, و كان هذا الملف غير موجود.

فئات الاستثناءات

الاستثناءات قد تحدث بسبب المستخدم (**User**), أو المبرمج (**Programmer**), أو بسبب الأجهزة المستخدمة (**Physical Resources**). بناءً على هذا, تم تقسيم الاستثناءات إلى ثلاث فئات أساسية:

- **Checked Exception**: تعني خطأ برمجي يحدث أثناء ترجمة البرنامج (أي قبل تشغيل الكود).
- **Unchecked Exception**: تعني خطأ منطقي يحدث أثناء تشغيل البرنامج.
- **Error**: تعني خطأ يحدث بسبب الجهاز الذي نحاول تشغيل البرنامج عليه.

Checked Exception: تعني استثناء يحدث أثناء ترجمة الكود, و هنا يكون الخطأ من المبرمج لأن الكود الذي كتبه يوجد فيه مشكلة ظاهرة من الأساس. **مثال**: الآن في حال قمنا بتعريف متغير نوعه int, ثم قمنا

بإعطائه نص كقيمة، سنلاحظ أنه سيظهر لنا خطأ قبل تشغيل البرنامج، و في حال حاولنا تشغيله, سيظهر الخطأ عندما يحاول ترجمة الكود.

```
Public class Main {
    Public static void main (String[] args) {
        int a;
        a = "this is incompatible type, 'a' should be String";
    }
}
```

سيظهر الخطأ التالي إذا قمنا بتشغيل البرنامج.

```
Exception in thread "main" java.lang.RuntimeException:
Uncompilable source code - incompatible types:
java.lang.String cannot be converted to int
```

هذا الخطأ يعني أن هناك مشكلة في النوع المستخدم لتخزين البيانات. ثم أخبرنا بسبب المشكلة وهي أنه لا يمكن تحويل الرقم الى نص, أي لا يمكن تخزين قيمة نوعها String في متغير نوعه int .

Unchecked Exception : تعني استثناء يحدث أثناء تشغيل البرنامج و يسمى أيضاً (Runtime Exception), و هو يتضمن الـ (Programming Bugs) و التي تعني أخطاء منطقية (Logical Errors) أو أخطاء سببها عدم استخدام الأشياء المعرفة في لغة البرمجة بالشكل الصحيح.(APIs errors)

الآن في حال قمنا بتعريف مصفوفة نوعها int تتألف من 5 عناصر، ثم قمنا بطباعة قيمة عنصر غير موجود فيها (مثل العنصر رقم 10). سنلاحظ أنه سيظهر الخطأ أثناء تشغيل البرنامج و ليس أثناء ترجمة الكود، و السبب أنه سيكتشف عدم وجود عنصر يحمل الـ index رقم 10 بعد أن يتم إنشاء هذه المصفوفة في الذاكرة (أي بعد ترجمة الكود و تنفيذه).

إذاً حدوث خطأ أثناء التشغيل يعني أن الخطأ لا يكتشف إلا أثناء محاولة تنفيذ الأوامر.

```
public class Main {
   public static void main(String[] args) {
      int[] a = { 1, 2, 3, 4, 5 };
      System.out.println( a[10] );
   }
}
```

سيظهر الخطأ التالي إذا قمنا بتشغيل البرنامج.

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 10

- هذا الخطأ يعني أن العنصر ليس موجود في المصفوفة ثم أخبرنا بسبب المشكلة و هي أنه لم يجد عنصر يحمل الـ index رقم 10.

Error: تعني خطأ يحدث بسبب الجهاز الذي نحاول تشغيل البرنامج عليه, لا علاقة أبداً للبرنامج بهذا الخطأ. فمثلاً إذا امتلأت ذاكرة الجهاز الذي يعمل عليه البرنامج سيحدث خطأ، وهو أن نظام التشغيل لا يقدر أن يشغل هذا البرنامج لأن ذاكرة الجهاز ممتلئة. وعندها سيظهر الرسالة التالية لتوضيح الخطأ JVM is out of Memory. لذلك تجد بعض البرامج تحفظ الأشياء التي يفعلها المستخدم كل مدة معينة.

بناء الاستثناءات في جافا

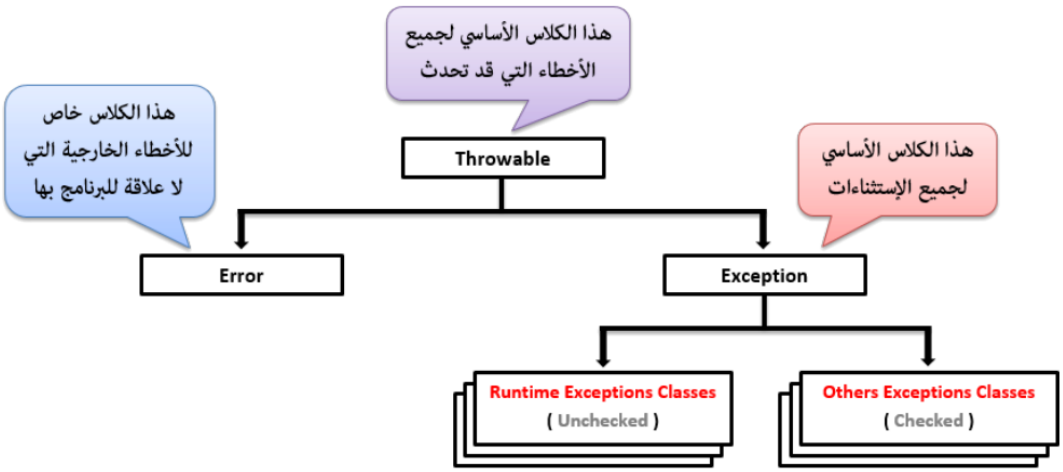
تم تقسيم الاستثناءات في جافا إلى عدة أنواع وكل نوع تم تمثيله في كلاس منعزل.

جميع هذه الكلاسات ترث من كلاس أساسي اسمه Exception .

الكلاس Exception بدوره يرث من كلاس اسمه Throwable .

أي كلاس يرث من الكلاس Exception هو كلاس يمثل استثناء معين.

الأخطاء التي سببها الأجهزة والتي تسمى Errors, تم إنشاء كلاس خاص لهم اسمه Error وهو يرث مباشرةً من الكلاس Throwable.



التقاط الاستثناء في جافا

التقاط الاستثناء يسمى ,Catching Exception وهو عبارة عن طريقة تسمح لك بحماية البرنامج من أي كود تشك بأنه قد يسبب أي خطأ باستخدام الجملتين try وcatch

أي كود مشكوك فيه يجب وضعه بداخل حدود الجملة try والجملة catchعبارة عن دالة يمكنك من خلالها معرفة كل شيء عن الخطأ الذي حدث وطريقة تعريف الجمل try وcatch.

```
try {
   // Protected Code
   هنا نكتب الأوامر التي قد تولد استثناء //
}
Catch (ExceptionType e) {
   // Error Handling Code
   هنا نكتب أوامر تحدد للبرنامج ماذا يفعل إذا قامت الـ try برمي استثناء //
}
```

الكود الذي نضعه بداخل الجملة try يسمى Protected Code وهذا يعني أن البرنامج محمي من أي خطأ قد يحدث بسبب هذا الكود.

الكود الذي نضعه بداخل الجملة catch يسمى Error Handling Code ويقصد منها الكود الذي سيعالج الاستثناء الذي قد يتم التقاطه.

الاستثناء الذي تقوم الجملة try برميه عبارة عن كائن من إحدى الكلاسات التي ترث من الكلاس Exception.

ملاحظة: عندما تستخدم الجملة try حتى لو لم تضع بداخلها أي كود، فأنت مجبر على وضع الجملة catch أو finally بعدها. كما أنه بإمكانك وضع العدد الذي تريده من الجملة catch وستتعرف على الجملة finally

```
public class Main {
   public static void main(String[] args) {
      try {                    // هنا قمنا بتجربة الكود الموجود بداخل الجملة try
         int[] a = new int [5];       // هنا قمنا بإنشاء مصفوفة تتألف من 5 عناصر
         System.out.println( a[10] );  // catch هنا حاولنا عرض قيمة عنصر غير موجود في المصفوفة, لذلك سيحدث خطأ, مما سيؤدي رمي إستثناء إلى الدالة
      }
      catch( Exception e ) {                    // هنا سيتم إلتقاط الإستثناء, ثم تخزينه في الكائن e
         System.out.println( "Exception thrown: " + e );
// هنا قمنا بعرض محتوى الكائن e لنعرف طبيعة الخطأ الذي حدث //
      }
      // بعد الإنتهاء من تجربة الكود سيتم تنفيذ باقي الأوامر الموجودة في البرنامج
      System.out.println( "The program still work properly" );
   }
}
```

```
Result of code
Exception thrown: java.lang.ArrayIndexOutOfBoundsException:
10
The program still work properly
```

إذا وضعنا الكلاس Exception كباراميتر للدالة ,catch فهذا يعني أن أي استثناء سترميه الدالة try ستقوم الدالة catch بالتقاطه.

في المثال السابق كان بإمكانك وضع الكلاس ArrayIndexOutOfBoundsException بدل الكلاس Exception, لأننا بعد تجربة الكود عرفنا أن نوع الخطأ الذي قد يرمى بسبب هذا الكود نوعه ArrayIndexOutOfBoundsException.

إذا كنت لا تعرف طبيعة الخطأ الذي قد يحدث ضع الكلاس Exception في الدالة catch لأن الكلاس Exception يعتبر الكلاس الأساسي والافتراضي لجميع الأخطاء التي قد تحدث.

وضع عدة جمل catch في جافا أحياناً الكود الواحد قد يسبب عدة أخطاء، ففي هذه الحالة ستضطر إلى معالجة كل خطأ قد يحدث على حدة ليستمر البرنامج في العمل بشكل طبيعي. إذاً يمكنك وضع العدد الذي تريده من الجملة catch بعد الجملةtry .

طريقة وضع أكثر من catch

```
try { // Protected Code
   هنا نكتب الأوامر التي قد تولد إستثناء //
}
catch(ExceptionType1 e1) {   // Error Handling Code For ExceptionType1
   }
catch(ExceptionType2 e2) {  // Error Handling Code For ExceptionType2
}
catch(ExceptionType3 e3) {   // Error Handling Code For ExceptionType3
 }
```

طريقة تعامل البرنامج مع الجملcatch

إذا قامت الجملة try برمي استثناء, سيتم إرساله إلى جملة الـ catch الأولى, بعدها سيتم مقارنة نوع الاستثناء المرمي مع نوع الاستثناء الموضوع كباراميتر في الدالة, إذا كان نوع الاستثناء المرمي هو نفسه نوع الاستثناء الموضوع كباراميتر, سيتم معالجة الخطأ الذي حدث بداخل هذه الـcatch .

إذا كان نوع الاستثناء المرمي غير النوع الموضوع كبارامیتر، سيتم الانتقال إلى جملة الـ catch التالية وتنفيذ نفس الخطوات السابقة.

في حال تم إيجاد استثناء يطابق الاستثناء المرمي، سيتم تنفيذ جميع الأوامر الموجودة في جملة الـ catch وبعدها سيتخطى البرنامج جميع جمل الـ catch الموجودة بعدها ويكمل باقي الأوامر الموجودة في البرنامج.

وفي حال لم يتم إيجاد أي استثناء يطابق الاستثناء المرمي، سيتوقف البرنامج ويظهر الخطأ الذي حدث.

```
public class Main {
  public static void main(String[] args) {
    String s = "12345678910111213141516171819202122223";
    int a;
    try {
      System.out.println( "s.charAt(28): " + s.charAt(28) );
      a = Integer.parseInt(s);
    }
    catch( StringIndexOutOfBoundsException e1 ) {          // e1 بعدها تقوم بتخزينه في الكائن
StringIndexOutOfBoundsException هذه الـ catch تلتقط الإستثناء الذي نوعه
      System.out.println( "Index is not exist in the string!" );
    }
    catch( NumberFormatException e2 ) {
  // e2 هذه الـ catch تلتقط الإستثناء الذي نوعه NumberFormatException بعدها تقوم بتخزينه في الكائن
      System.out.println( "Can't convert 's' to a number because is to long!" );
    }
    catch( Exception e3 ) {
      System.out.println( "Exception thrown: " + e3 );
    }
    System.out.println( "The program still work properly" );
  }
```

The result is :

s.charAt(28): 9

Can't convert 's' to a number because is to long!

The program still work properly

يمكنك أيضاً وضع أكثر من استثناء بداخل نفس الجملة catch باستخدام العامل | والذي يعني " أو"

عندها إذا رمت الجملة try أي استثناء من الاستثناءات الموجودة في الجملة catch سيتم معالجة الاستثناء فيها.

```
public class Main {
   public static void main(String[] args) {
      String s = "123456789101112131415161718192021222 3";
      int a;
      try {
         System.out.println( "s.charAt(28): " + s.charAt(28);
         a = Integer.parseInt(s);
      }
      catch( StringIndexOutOfBoundsException|NumberFormatException e1 ) {
System.out.println( "The string 's' throw: " +e1 );
      }
      catch( Exception e2 ) {
         System.out.println( "Exception thrown: " + e2 );
      }
      System.out.println( "The program still work properly" );
  }}
```

```
The result is s.charAt(28): 9
The string 's' throw: java.lang.NumberFormatException: For
input string: "123456789101112131415161718192021222 3"
The program still work properly
```

الجملة finally في جافا :الجملة finally تأتي بعد الجملتين try و .catchالكود الموضوع في الجملة finally يتنفذ دائماً، أي في حال حدث استثناء أو لم يحدث فإنه سيتنفذ .

مكان وضع الجملة finally

```
try {
   // Protected Code
}
catch(ExceptionType1 e1) {
   // Error Handling Code For ExceptionType1
}
catch(ExceptionType2 e2) {
   // Error Handling Code For ExceptionType2
}
catch(ExceptionType3 e3) {
   // Error Handling Code For ExceptionType3
}
finally {
   // Optional Cleanup Code
   // هنا نقوم بكتابة أوامر للتخلي عن أي شيء لم يعد البرنامج بحاجة له
}
```

```
public class Main {
   public static void main(String[] args) {
      String s = "123456789101112131415161718192021222 3";
      int a;
      try {
         System.out.println( "s.charAt(28): " + s.charAt(28) );
         a = Integer.parseInt(s);
      }
      catch( StringIndexOutOfBoundsException e1 ) -
         System.out.println( "Index is not exist in the string!" );
      }
      catch( NumberFormatException e2 ) {
         System.out.println( "Can't convert 's' to a number because is to long!" );
      }
      catch( Exception e3 ) {
         System.out.println( "Exception thrown: " + e3 );
      }
      finally {
         System.out.println( "finally codes always execute" );
      }
      System.out.println( "The program still work properly" );
   }
}
```

```
The result is : s.charAt(28): 9
Can't convert 's' to a number because is to long!
finally codes always execute
The program still work properly
```

ستعرف فائدة الجملة finally في دروس لاحقة عندما تتعامل مع الملفات، الشبكات, قواعد البيانات إلخ..

إذا كان البرنامج يقرأ محتوى ملف نصي، سيكون عليك التأكد من إغلاق الملف بعد الإنتهاء من قرائته.

إذا كان البرنامج يقرأ بيانات من قاعدة بيانات، سيكون عليك التأكد من إغلاق قاعدة البيانات بعد الإنتهاء من التعامل معها.

إذا كان البرنامج يتصل بالشبكة، سيكون عليك التأكد من إغلاق الإتصال بعد الإنتهاء من التعامل معها.

و التي يمكن استخدامها لمعرفة الأخطاء Exceptionهنا وضعنا بعض الدوال الموجودة في الكلاس التي حدثت في البرنامج.

الدالة مع تعريفها

- public String getMessage : ترجع رسالة بسيطة تظهر لنا الخطأ الذي حدث.
- public String toString() : ترجع اسم كلاس الاستثناء الذي حدث مضافاً إليه نتيجة الخطأ التي ترجعها الدالة getMessage.
- public void printStackTrace(): يستخدمها المبرمج لمعرفة الأخطاء التي قد تحدث أثناء تشغيل البرنامج, و هي مفيدة جداً لأنها تظهر الخطأ الذي حدث كما هو بكامل تفاصيله. إذاً هذه الدالة تستخدم للـ Debugging.

المثال الأول

•في هذا المثال سنستخدم الدالة getMessage لمعرفة سبب الخطأ الذي حدث فقط.

```
public class Main {
  public static void main(String[] args) {
     String s = "abcd 12345";
     int a;
     try {  a = Integer.parseInt(s);              // int هنا حاولنا تحويل قيمة s لقيمة}
     catch( Exception e ){
        System.out.println( e.getMessage() );   // getMessage() هنا قمنا لمعرفة الخطأ الذي حدث باستخدام الدالة
     }}}
```

The result is : `For input string: "abcd 12345"`

المثال الثاني

في هذا المثال سنستخدم الدالة toString لمعرفة سبب الخطأ الذي حدث و إسم الكلاس الذي يمثل الإستثناء الذي حدث.

```
public class Main {

   public static void main(String[] args) {

      String s = "abcd 12345";

      int a;

      try {

         a = Integer.parseInt(s);            // int هنا حاولنا تحويل قيمة s لقيمة

      }

      catch( Exception e ) {                 // e تلتقط أي إستثناء قد يحدث, بعدها تقوم بتخزينه في
هذه الـ catch الكائن

         System.out.println( e.toString() );  // toString() هنا قمنا لمعرفة الخطأ الذي حدث
باستخدام الدالة

      }

   }

}
```

The result is : `java.lang.NumberFormatException: For input string: "abcd 12345"`

المثال الثالث

•في هذا المثال سنستخدم الدالة printStackTrace لمعرفة تفاصيل الخطأ الذي حدث بتفصيل.

```
public class Main {
   public static void main(String[] args) {
      String s = "abcd 12345";
      int a;
      try {
         a = Integer.parseInt(s);    // int هنا حاولنا تحويل قيمة s لقيمة
      }
      catch( Exception e ) {         // e تلتقط أي إستثناء قد يحدث, بعدها تقوم بتخزينه في الكائن
catch هذه الـ
         e.printStackTrace();        // هنا قمنا printStackTrace() لمعرفة الخطأ الذي حدث
باستخدام الدالة
      }
   }
}
```

The result is :

```
java.lang.NumberFormatException: For input
string: "abcd 12345"
at
java.lang.NumberFormatException.forInputString(NumberFo
rmatException.java:65)
at java.lang.Integer.parseInt(Integer.java:580)
at java.lang.Integer.parseInt(Integer.java:615)
at testexceptions.Main.main
```

الكلمتين throws و throw في جافا

إذا قمت بتعريف دالة وأردت لهذه الدالة أن ترمي استثناء إذا حدث شيء معين فعليك وضع الكلمة throws بعد أقواس البارامترات ثم تحديد نوع الاستثناء الذي قد ترميه الدالة، وإذا قمت مسبقاً بتعريف استثناء يمكنك جعل الدالة تقوم برميه.

Syntax:

throw *Instance*

Example:

throw new ArithmeticException("/ by zero"); \\subclass of Throwable.

الكلمة throw تستخدم لتحديد نوع الاستثناء الذي سيرسل إلى الجملة catch يجب ان يكون كلاس فرعي من **Exception** . اما الكلمة throws تستخدم لتحديد نوع الاستثناء الذي قد تقوم الدالة برميه في حال كنت تريد تجربة الكود ومعالجته في مكان استدعاء الدالة.

المثال الأول: في هذا المثال قمنا ببناء دالة ترمي استثناء وتعالجه بداخلها في حال قمنا بإعطائها عمر أكبر من 63.

```
public class Main {
   public static void main(String[] args) {
      checkAge(70);
   }
   public static void checkAge (int age) {
      try {
         if(age > 63)
            throw new Exception("you are too old!");
      }
      catch( Exception e ) {
         System.out.println( e.getMessage() );
      } }
}
```

Th result is: you are too old!

Syntax:

type method_name(parameters) throws exception_list

`exception_list is a comma separated list of all the exceptions which a method might throw.`

إذاً لا نحتاج إلى استخدام الكلمة throws إلا إذا أردنا وضع الدالة بداخل الجملة try عند استدعائها.

المثال الثاني: في هذا المثال قمنا ببناء دالة ترمي استثناء يعالج في مكان الاستدعاء في حال قمنا بإعطائها عمر أكبر من 63.

```
public class Main {
   public static void main(String[] args) {
      try {
         checkAge(70);
      }
      catch( Exception e ) {
         System.out.println( e.getMessage() );
      }
   }
   public static void checkAge (int age) throws Exception{
      if(age > 63) {
         throw new Exception("you are too old!");
      }
   }
}
```

The result is: you are too old!

المثال الثالث :في هذا المثال قمنا ببناء دالة قد ترمي استثناءين يتم معالجتهما في مكان الاستدعاء.

```
public class Main {
   public static void main(String[] args) {
      try {
         checkAge(0);
      }
      catch( ArithmeticException e1 ) {
         System.out.println( e1.getMessage() );
      }
      catch( Exception e2 ) {
         System.out.println( e3.getMessage() );
      }
   }
   public static void checkAge (int age) throws ArithmeticException, Exception{
      if(age <= 0) {
         throw new ArithmeticException("This is ArithmeticException");
      }
      else{
         System.out.println( 100/age );
      }
   }
}
```

The result is : This is ArithmeticException

انشاء Exception جديد واستخدامه في جافا

خطوات إنشاء استثناء جديد

- يجب إنشاء كلاس جديد يرث من الكلاس.Exception
- يجب استدعاء الكونستركتور الـ Superclass أي الكونستركتور الخاص بالكلاس Exception في كونستركتور الكلاس الجديد الذي نقوم بإنشائه.
- يجب تمرير الرسالة التي سيرميها الاستثناء في الدالة super.

مثال: هنا قمنا بتعريف كلاس جديد يمثل Exception و قمنا بتسميته MyException.

```
public class MyException extends Exception {
   public MyException(String msg){
      super(msg);
   }
}
```

```
public class Main {
   public static void main(String[] args) {
      checkAge()
      try {
         checkAge(20);
         checkAge(18);
         checkAge(10);
      }
      catch( MyException e ) {
         System.out.println( e.getMessage() );
      }
   }
   public static void checkAge(int age) throws MyException {
      if(age < 13) {
         throw new MyException("you can't watch horror movies");
      }
      else {
         System.out.println( "you can watch the movie" );
      }
   }
}
```

The result is: you can watch the movie

you can watch the movie

you can't watch horror movies