2.مثال: التعامل مع and inputMismatchExceptions Ari thmeticExcepti ons

- يستخدم التطبيق الوارد في الشكل 11.2معالجة الاستثناءات لـ معالجة أي مدخلات وإدخال من نوع Ari thmeti cExcepti الذي يظهر .

- إذا ارتكب المستخدم خطأً ، فإن البرنامج يمسك ويتعامل (أي يتعامل مع) الاستثناء -في هذه الحالة ، مما يسمح للمستخدم بمحاولة إدخال المدخلات مرة أخرى.

```
1 // 11.2:DivideByZeroWithExceptionHandling.java الشكل
2 // HandlingArithmeticExceptionsandInputMismatchExceptions.
3 importjava.util.InputMismatchException؛ <---------- -
4 importjava.util.Scanner ؛
5
6 publicclassDivideByZeroWithExceptionHandling
7 {
8 //مظاهرة رامية واستثناء عند تقسيم على صفر يحدث
publicstaticintquotient (intnumerator ، intdenominator)
9 10 throwsArithmeticException <---------- -
12 {
11 / returnnumeratorالقاسم ؛ // ممكن تقسيمها
13 // {مقياس النهاية
14
15 ثابت عام { رئيسي (سلسلة [] args) فارغ
16
17 Scannerscanner = newScannerfSystem.in ) ؛ // scanner for
18 المدخلات المنطقية = continueLoopصحيح ؛ // ifmoreinputisneededددحي
19
```

نوع الاستثناء الذي طرحه العديد

طرق فئة الماسح الضوئي

قم برمي استثناء حسابي

قد يشير إلى أن هذه الطريقة

الشكل HandlingArithmeticExposionsandInputMismatchException | 11.2الاستثناءات.

(جزء 4.)

```
فعل
{
حاول // قراءة رقمين وحساب حاصل القسمة ---------------------------------------------------- *
{
intSystem.out.printf( "الرجاء إدخال عدد صحيح بسط: ")؛ البسط ؛ ()
System.out.print( "الرجاء إدخال عدد صحيح:")؛
int = scanner.nextlntf) ؛ مقام

نتيجة = intحاصل البسط ، المقام) ؛
System.out.printff "\ n النتيجة: %d / %d = %d \ n" ، البسط ،
المقام ، النتيجة) ؛
ContinueLoop = خطأ ؛ // الإدخال ناجح ؛ حلقة النهاية
} // إنهاء المحاولة
catch (InputMismatchException inputMismatchException) <
{
System.err.printff "\ n استثناء: % s \ n"،
inputMismatchException) ؛
scanner.nextLine () ، // تجاهل الإدخال حتى يتمكن المستخدم من المحاولة مرة أخرى
System.out.pri ntl n (
"يجب إدخال أعداد صحيحة. يرجى المحاولة مرة أخرى. \ n" ) ؛
} // end catch
```

يبدأ رمز القفل في أي حالة قد يحدث استثناء: يحتوي أيضًا على كتلة تحتوي على كود لا ينبغي تنفيذه في حالة حدوث استثناء

المصيد والعمليات InputMismatch استثناءات

الشكل | I1.2 التعامل مع الاستثناءات الحسابية و InputMismatchException.

(الجزء 2 من 4.)

```
catch (ArithmeticExceptionarithmeticException)
{
System.err.printf (System.out.println (
"Zeroisaninvaliddenominator.Pleasetryagain. \ n");
} // endcatch
} while (continueLoop); // enddo ... while
} // endmain
} // endclassDivideByZeroWithExceptionHandling
```

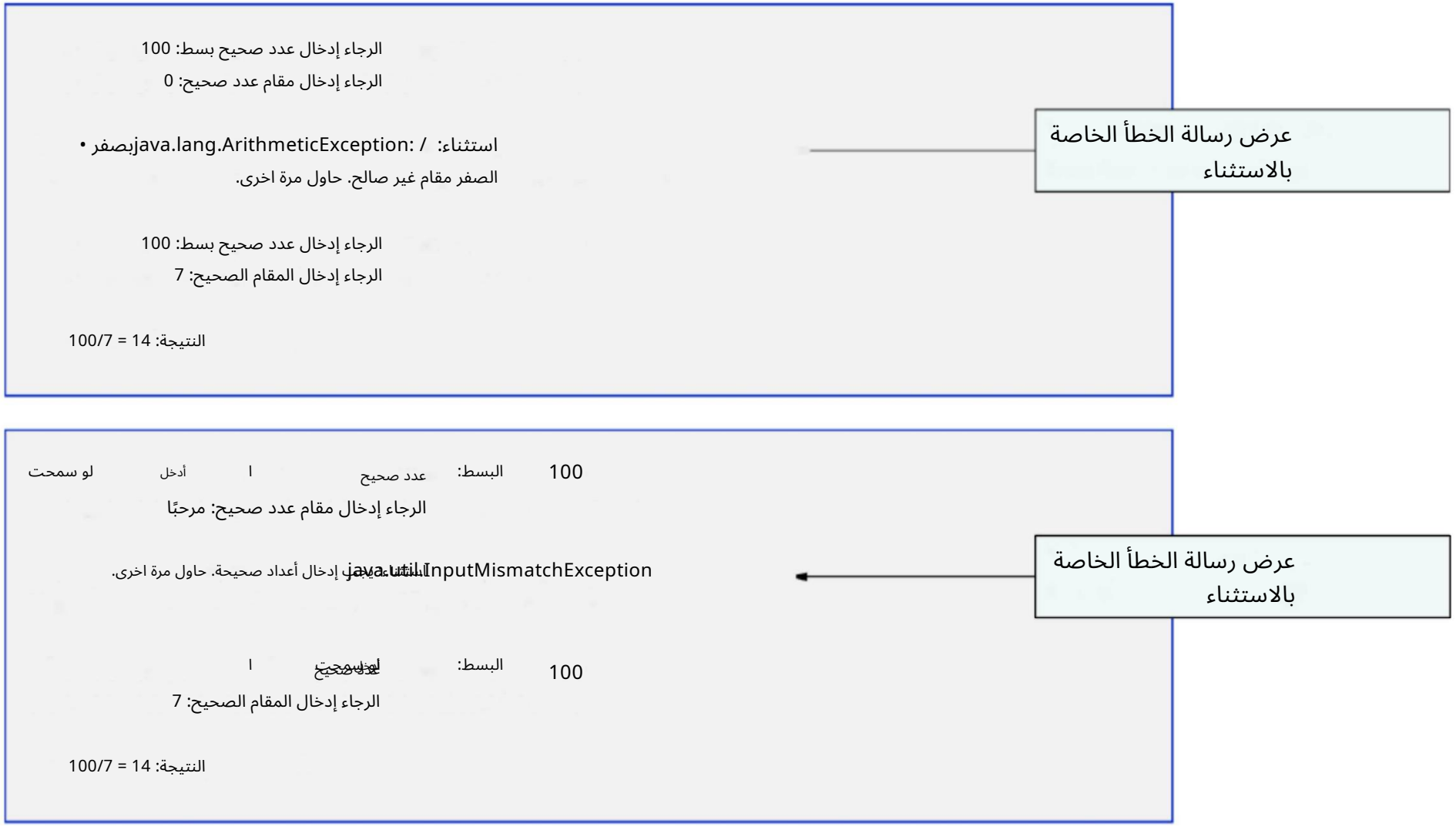
استثناء حسابي);

المصيد والعمليات
علم الحساب
استثناءات

```
الرجاء إدخال عدد صحيح بسط: 100
الرجاء إدخال المقام الصحيح: 7

النتيجة: 14 = 100/7
```

الشكل HandlingArithmeticExposionsandInputMismatchException | 11.2الاستثناءات.

(Part3of4.)



الشكل | 11.2التعامل مع الاستثناءات الحسابية و .InputMismatchException

(الجزء 4من 4.)

ملحوظة:

- استثناء غير معلوم -استثناء لا توجد له كتل صيد مطابقة.

- تذكر أن الاستثناءات السابقة غير المعلومة تسببت في التطبيق للإنهاء مبكرًا.
- لا يحدث هذا دائمًا كنتيجة لاستثناءات غير معلومة.

- تستخدم Java نموذجًا متعدد مؤشرات الترابط لتنفيذ البرنامج.
- كل خيط هو نشاط مواز.
- يمكن أن يحتوي برنامج واحد على العديد من الخيوط.
- إذا كان البرنامج يحتوي على مؤشر ترابط واحد فقط ، فسيحدث استثناء غير معلوم البرنامج المراد إنهاءه.
- إذا كان البرنامج يحتوي على مؤشرات ترابط متعددة ، فسيؤدي استثناء غير معلوم إلى إنهاء مؤشر الترابط الذي حدث فيه الاستثناء فقط.

- في حالة حدوث استثناء في كتلة try ، يتم إنهاء كتلة try على الفور وينتقل التحكم في البرنامج إلى أول كتلة catch مطابقة.

- بعد معالجة الاستثناء ، يُستأنف التحكم بعد آخر كتلة catch .

- يُعرف بنموذج الإنهاء الخاص بمعالجة الاستثناءات.
- بعض اللغات تستخدم نموذج استئناف الاستثناء
المعالجة ، حيث يتم التحكم فيها بعد معالجة الاستثناء
يستأنف بعد نقطة الرمي مباشرة.

- في حالة عدم وجود استثناءات في كتلة try ، يتم تخطي كتل catch ويستمر التحكم مع العبارة الأولى بعد كتل catch

- تشكل كتلة try وblock catch و / أو fl nal 1 y المقابل عبارة try .

دنع إنهاء كتلة ، tryتخرج المتغيرات المحلية المعلنة في الكتلة عن النطاق.

المتغيرات المحلية لمجموعة tryلا يمكن الوصول إليها في كتل الصيد المقابلة.

عندما تنتهي كتلة ، catchفإن المتغيرات المحلية المعلنة داخل كتلة catch(بما في ذلك معلمة الاستثناء) تخرج أيضًا عن النطاق.

يتم تجاهل أي كتل catchمتبقية في تعليمة ، tryويستأنف التنفيذ في السطر الأول من التعليمات البرمجية بعد تسلسل .try ... catch

كتلة ، y 1 nal 1 fإذا كان أحدها موجودًا.

عبارة - throws تحدد طريقة ما في الاستثناءات رميات.

- يظهر بعد قائمة معلمات الأسلوب وقبل نص الأسلوب.

- يحتوي على قائمة مفصولة بفواصل للاستثناءات التي ستطرحها الطريقة في حالة حدوث مشكلات مختلفة.
  - قد يتم طرحها بواسطة عبارات في جسم الطريقة أو بواسطة طرق تسمى من الجسم.

- يمكن للطريقة طرح استثناءات من الفئات المدرجة في رميات بند أو من الفئات الفرعية الخاصة بهم.

- يتم إبلاغ عملاء طريقة ما مع شرط رميات أن الطريقة قد تؤدي إلى استثناءات.

- عندما تطرح عملية استثناءً ، تنتهي الطريقة ولا تُرجع قيمة ، وتخرج متغيراتها المحلية عن النطاق.

- إذا كانت المتغيرات المحلية عبارة عن إشارات إلى كائنات ولم تكن هناك مراجع أخرى لهذه الكائنات ، فستكون الكائنات متاحة لجمع البيانات المهملة.

متى يتم استخدام معالجة الاستثناءات

تم تصميم معالجة الاستثناءات لمعالجة الأخطاء المتزامنة ، والتي تحدث عند تنفيذ العبارة.

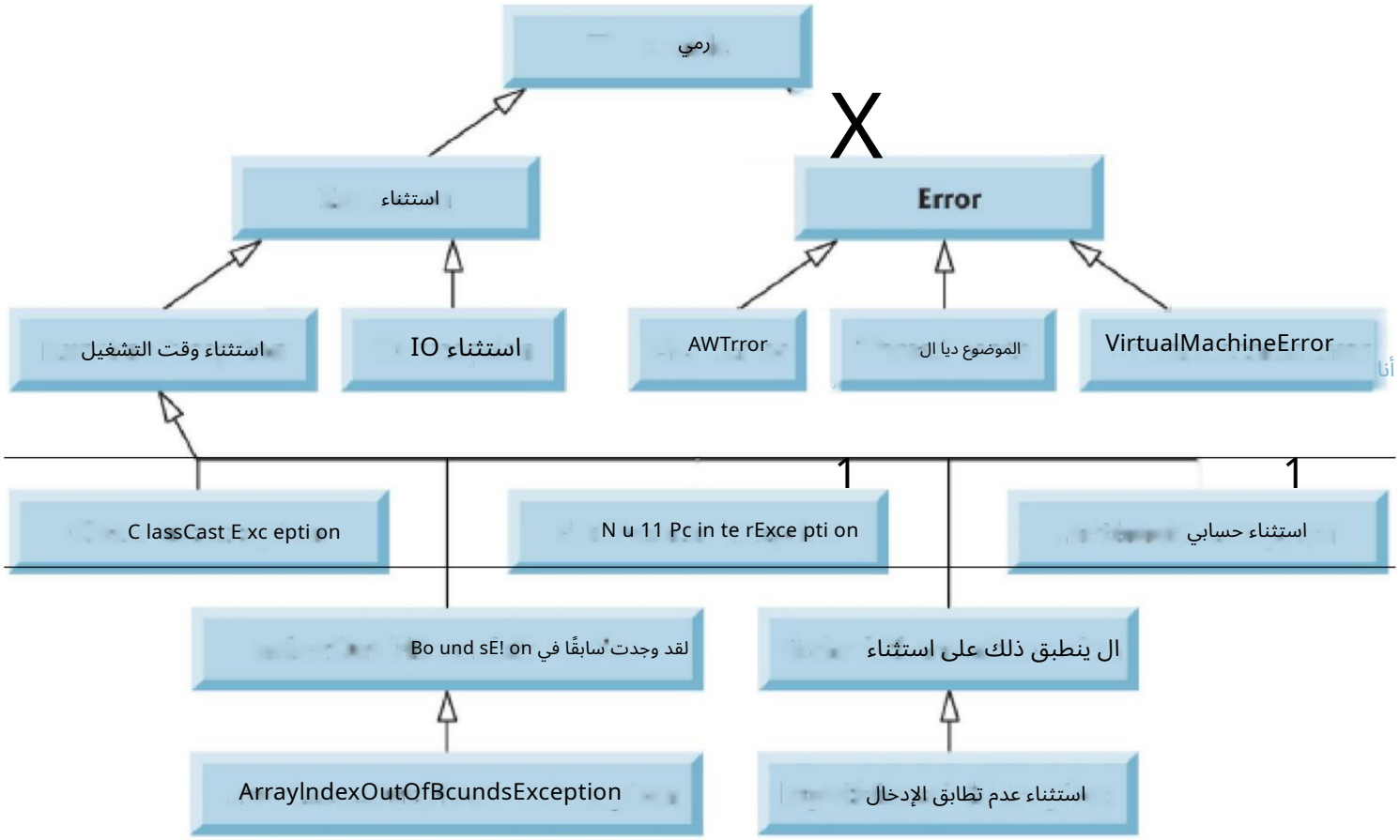
أمثلة شائعة:

- مؤشرات مصفوفة خارج النطاق
- تجاوز حسابي
- قسمة على صفر
- معلمات طريقة غير صالحة
- مقاطعة الخيط
- تخصيص ذاكرة غير ناجح

لم يتم تصميم معالجة الاستثناءات لمعالجة المشكلات المرتبطة بالأحداث غير المتزامنة

- إتمام إدخال / إخراج القرص
- وصول رسائل الشبكة
- نقرات الماوس وضغطات المفاتيح

التسلسل الهرمي لاستثناء Java(تابع)



الشكل IJ3 | جزء من التسلسل الهرمي للميراث من فئة Throwable.

التسلسل الهرمي لاستثناء Java(تابع)

- الاستثناءات التي تم التحقق منها مقابل الاستثناءات التي لم يتم التحقق منها.
- يفرض المترجم مطلب التقاط أو إعلان للاستثناءات التي تم التحقق منها .

- يحدد نوع الاستثناء ما إذا كان محددًا أو غير محدد.

- فئات فرعية مباشرة أو غير مباشرة من فئة RuntimeException (الحزمة زأفا. 1آنغ) استثناءات. دون رادع
- يحدث عادةً بسبب عيوب في رمز البرنامج (على سبيل المثال ، Ar r ay I ndexoutof Bounds exceptions).

- فئات فرعية من استثناء! على ولكن ليس Runti meExcepti على استثناءات. التحقق
- بسبب الظروف التي لا تخضع لتحكم البرنامج -على سبيل المثال ، في معالجة الملفات ، لا يمكن للبرنامج فتح ملف لأن الملف غير موجود.

التسلسل الهرمي لاستثناء Java(تابع)

تعتبر الفئات التي ترث من فئة خطأ

دون رادع.

المفحوصات المترجم كل طريقة استدعاء وطريقة

إعلان لتحديد ما إذا كان الأسلوب رمى

فحص الاستثناءات. إذا كان الأمر كذلك ، يتحقق المترجم من أن الاستثناء المحدد قد تم اكتشافه أو

أنه تم اكتشافه

أعلن في بند الصفوف .

تحدد عبارة throwsالاستثناءات التي يرميها الأسلوب .

عادة لا يتم القبض على مثل هذه الاستثناءات في جسم الطريقة.

التسلسل الهرمي لاستثناء Java(تابع)

تعتبر الفئات التي ترث من فئة خطأ

دون رادع.

المترجم الفحوصات كل طريقة استدعاء وطريقة

إعلان لتحديد ما إذا كان الأسلوب رمى

فحص الاستثناءات. إذا كان الأمر كذلك ، يتحقق المترجم من أن الاستثناء المحدد قد تم اكتشافه أو

أنه تم اكتشافه

أعلن في بند الصفوف.

تحدد عبارة throws الاستثناءات التي يرميها الأسلوب.

عادة لا يتم القبض على مثل هذه الاستثناءات في جسم الطريقة.

التسلسل الهرمي لاستثناء Java(تابع)

تعتبر الفئات التي ترث من خطأ فئة

دون رادع.

الفيروسات الممرجم

كل طريقة استدعاء وطريقة

إعلان لتحديد ما إذا كان الأسلوب رمى

فحص الاستثناءات. إذا كان الأمر كذلك ، يتحقق المترجم من أن الاستثناء المحدد قد تم اكتشافه أو أنه تم اكتشافه

أعلن في شرط رميات.

تحدد جملة الصفوف lth الاستثناءات التي يرميها الأسلوب.

عادة لا يتم القبض على مثل هذه الاستثناءات في جسم الطريقة.

التسلسل الهرمي لاستثناء Java(تابع)

شرط الالتقاط أو الإعلان الكود الذي يُنشئ الاستثناء ، يجب أن يتم لقه في try

كتلة ويجب أن توفر معالج التقاط لنوع الاستثناء المحدد (أو أحد الفئات الفائقة).

شرط الالتقاط أو الإعلان يعلن

اذا لم يتم استيفاء شرط الإمساك أو الإعلان ، فإن

سيصدر المترجم رسالة خطأ تشير إلى وجوب اكتشاف الاستثناء أو إعلانه.

إذا كان صراحة فتم إغراء التجميع الاستثناء الذي تم فحصه (أو استدعاء طريقة أخرى من شأنها أن تم التحقق من الاستثناء) وهذا الاستثناء

لم يتم إدراجها في بند طريقة العرض.

إذا كان كطريقة صنف فرعي عادة عن طريقة الطبقة الفائقة، فهذا خطأ، فهو خطأ في طريقة لغته الفرعية لطريقة اللغة الفرعية، ولكن

هناك استثناءات أخرى ترميها بطريقة مشابهة لطريقة الطبقة الفائقة التي تم تجاوزها.

إذا كانت الاستثناءات المعلنة في طريقة ما ترمي بشكل صريح استثناءات محددة ، فإن

أو يجب أن يتم القبض

يتم الإعلان عن استثناء ذو مغزى في طريقة ما ، يجب أن

التسلسل الهرمي لاستثناء Java(تابع)

- لا يتحقق المترجم من الكود لتحديد ما إذا كان قد تم اكتشاف استثناء لم يتم التحقق منه أو إعلانه.
- يمكن عادةً منع حدوث ذلك عن طريق الترميز المناسب.
- على سبيل المثال ، يمكن تجنب تشغيل Ari thmeti cExcepti إذا كانت إحدى الطرق تضمن أن المقام ليس صفراً قبل محاولة إجراء القسمة.

- لا يشترط إدراج الاستثناءات التي لم يتم تحديدها في شرط رميات الأسلوب.
- حتى لو كان الأمر كذلك ، فليس مطلوبًا أن يتم تسجيل مثل هذه الاستثناءات من خلال تطبيق ما.

sei nt، par

التسلسل الهرمي لاستثناء Java(تابع)

يمكن أيضًا التقاط معلمة صيد من نوع الطبقة الفائقة

كل أنواع الفئات الفرعية لنوع الاستثناء هذا.

- يُمكِّن المصيد من التعامل مع الأخطاء الإلكترونية ذات الصلة بإيجاز

الرموز

- يسمح بمعالجة متعددة الأشكال للاستثناءات ذات الصلة
  - اصطياد الاستثناءات ذات الصلة في كتلة catchواحدة يكون منطقيًا فقط إذا كان سلوك المعالجة هو نفسه لجميع الفئات الفرعية.
- You can also catch each subclass type individually if

تتطلب هذه الاستثناءات معالجة مختلفة.

التسلسل الهرمي لاستثناء Java(تابع)

- إذا كانت هناك كتل صيد متعددة تطابق ملف نوع الاستثناء ، أول كتلة catch مطابقة فقط ينفذ.
- من الخطأ التجميعي الإمساك بالنوع نفسه بالضبط كتلتان مختلفتان للقبض مرتبطان بـ كتلة خاصة ر.

يخضع اصطياد الفئة الفرعية بشكل فردي إذا كانت الأنواع تخطئ في فئة الأنواع الفرعية

السابق

تتسع مع الفئات الفرعية أو ربما أكثر ... يضمن اصطياد الطبقة الفائقة أن الكائنات

كاليفورنيا

يضمن تحديد موقع الطبقة الفائقة بعد كل

يتم اصطياد جميع الأنواع الفرعية للفئات الفرعية من تلك الفئة الفائقة أن

نكون استثناءات فئة فرعية

يكتب كتلة catch لاستثناء فئة superclass be وضع أ الصدارة

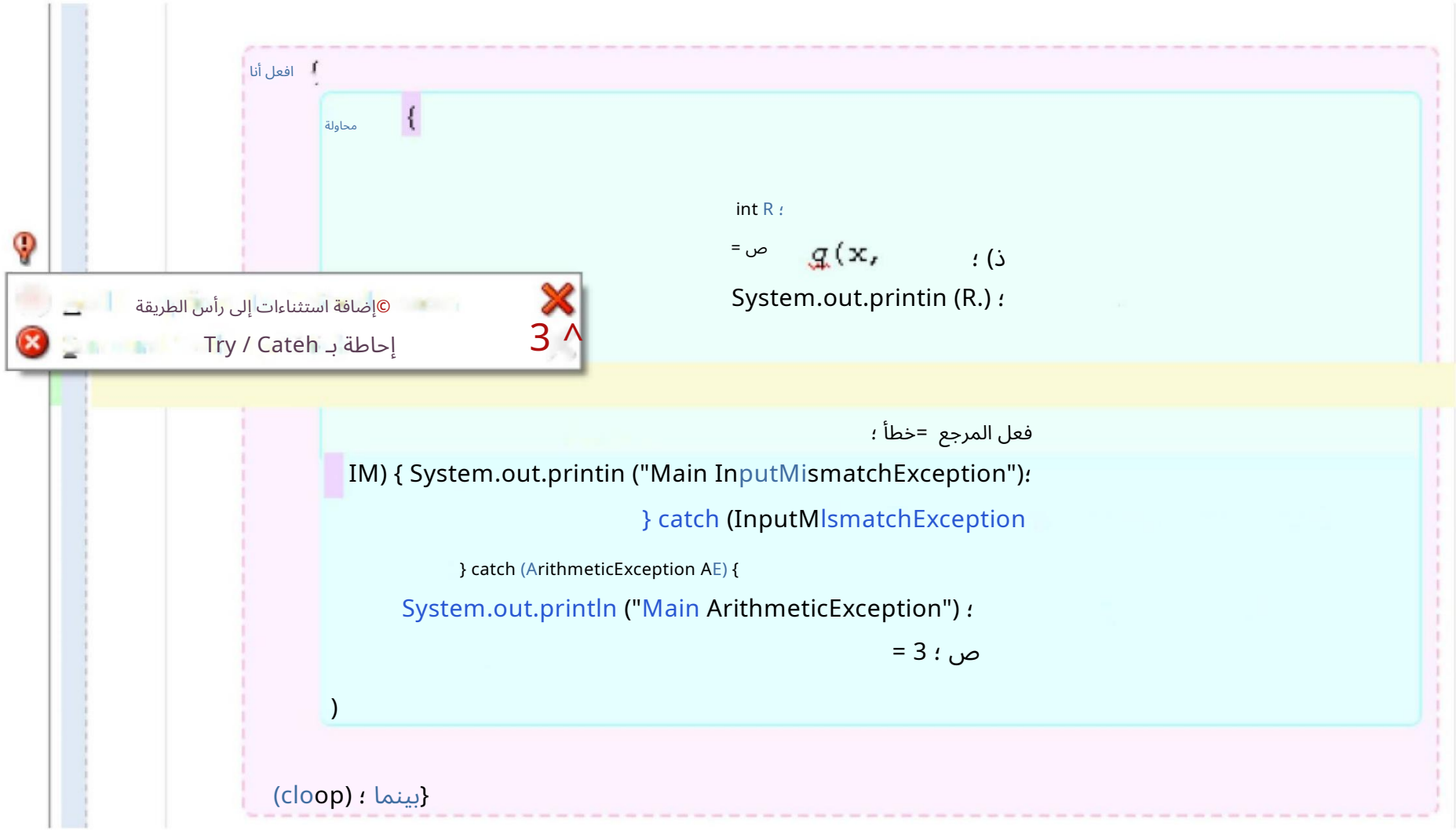
الكتل من التي تلتقط استثناءات فئة فرعية خطأ في التجميع تلك كاليفورنيا أخرى

لذلك أ

```
فئة عامة أ {
يطرح public static int q (int n، int d) استثناء {
عودة ن / د ؛

}
العامة الثابتة الفراغ الرئيسي (سلسلة [] args) {
ضرب منطقي =صحيح ؛ كثافة العمليات س ؛ int y = 0 ؛ الماسح= 15 ؛ الماسح = sالماسح
الجديد ؛ (System.in)
فعل {

محاولة {
int R ؛
؛ R = q (x ، y) ؛ System.out.println (R) ؛ خرق R =خطأ ؛

IM) {System.out.printlnf'Main InputMismatchException "):
} catch (InputMismatchException
AE) {System.out.println ("Main ArithmeticException"):
= 3 ؛ ص} catch (ArithmeticException

}
{بينما {} ؛ (cloop)
```

```
فئة عامة أ {
static int q(int n ، int d) يطرح استثناء {
عودة ن / د ؛
}
main(سلسلة [] args) الفراغ العام
cloop = true؛ الفراغ العام [] args)
int y = 0 ؛ {إنتكس ؛ 15 =الماسح = الماسح الجديد
(System.in) ؛
فعل {
محاولة {
int R ؛
R = q (x ، y) ؛
```


```
System.out. println (R) ؛ خرق =خطأ ؛

IM) {System.out.printlnf'Main InputMismatchException ");
} catch (InputMismatchException
ption AE) {System.out.println ("Main ArithmeticException");
} catch (ArithmeticExce
ص ؛ 3 =
}
(cloop) ؛ {} بينما
```



فئة عامة أ {

static int q(int n ، int d) يطرح استثناء {

العودة الحاصة : n / d ؛ 824987

}

يطرح public static void main (String [] args) استثناء {

ضرب منطقي =صحيح ؛ كثافة العمليات س ؛ int y = 0 ؛ 15 =الماسح = الماسح الجديد ؛ (System.in)

فعل {

محاولة {

int R ؛

R = q (x ، y) ؛ System.out.println (R) ؛ خرق =خطأ ؛

} catch (InputMismatchException IM) {

printin ("Main InputMismatchException") ؛ النظام ، خارج.

} catch (ArithmeticException AE) {

out.println ("Main ArithmeticException") ؛ ص ؛ 3 = نظام.

}

}

{بينما } ؛ (cloop)

فئة عامة؛ a { public return n / d

static int q(int n ، int d) { يطرح استثناء

public static void main (String [] args) { boolean cloop = true؛ {كثافة العمليات س ؛ y = 0

int ؛ 15 =الماسح= الماسح الجديد ؛ (System.in)افعل ؛ (x ، y) R = q ؛ int R {try {النظام ، خارج.

println (R) ؛ خرق =خطأ ؛ { ؛ (" Main ArithmeticException ")؛ {System.out.println (AE

IM) { System.out.printlnf'Main InputMismatchException ");} catch (ArithmeticException

catch (InputMismatchException ص Jcatch ؛ 3 =(استثناء هـ) {

أو

} } while (cloop)؛ }}

أخيرًا بلوك.

f 1 nal 1 y ستنفذ الكتلة سواء أكان ملف
تم طرح استثناء في كتلة المحاولة المقابلة.

سيتم تنفيذ الكتلة أخيرًا إذا خرجت كتلة t ry بحلول
باستخدام عبارة return أو break أو conti nue أو
ببساطة عن طريق الوصول إلى قوس الإغلاق الأيمن.

تنفيذ إذا تم إنهاء التطبيق على الفور ليس. أخيرًا منع الإرادة
عن طريق استدعاء الأسلوب
System.exit.

```
1  //الشكل :11.5استخدام الاستثناءات ، حاول جافا ... .fi nal * ly ،. .catchآلية
2  J!                                                معالجة الاستثناءات.
3
5  فئة عامة باستخدام الاستثناءات {
4
6  سلسلة mainf العامة الثابتة الفارغة ![] ) 1 args (
8
7  حاول {
9
10 throwExceptionQ ؛ // call method throwException } // end try catch(استثناء
11 throwException { طرحه بواسطة تم استثناء // (استثناء
17
13
15 end catch // { ؛ ( "معالجة الاستثناء الرئيسي" )System.err.println
14
16
17 mai n النهاية doesNotThrowExceptionO ؛ } //
18
19
```

تبدأ سلسلة استدعاء يكون فيها ملف
سيتم طرح استثناء

تبدأ سلسلة مكالمات لا يوجد فيها
تحدث استثناءات

11.5 تين.

حاول ... التقاط ، آلية معالجة الاستثناءات في التجمع (الجزء الأول من 4.)

```
20  // demonstrate try...catch...finally
21  public static void throwException() throws Exception
22  {
27     try // throw an exception and immediately catch it
26     {
25        System.out.println( "Method throwException" );
24        throw new Exception(); // generate exception
23     } // end try
28                                  { (استثناء استثناء)catch        //التقاط الاستثناء في المحاولة
32
31                                  نظام .println .rre( "استثناء تمت معالجته في طريقة throwException" ); // رمي الاستثناء
30                                  إعادة رمي لمزيد من المعالجة
29
                                    //لن يتم الوصول إلى الكود هنا ؛ من شأنه أن يسبب أخطاء في الترجمة

                                    catch { بغض النظر عما يحدث في} // end catch finally // executes
34 35 3637                          try ...
38
39                                  System.err.println( "تم التنفيذ أخيرًا في throwException" ); } // انتهى أخيرًا
40
43
42                                  //لن يتم الوصول إلى الكود هنا ؛ من شأنه أن يسبب أخطاء في الترجمة
41
44                                  // }طريقة النهاية throwException
```

This method might throw an `Exception` (this is a *checked* type)

Throws a new `Exception` that is caught at line 28 and thrown again at line 32

إعادة طرح الاستثناء يعني أنه لم يتم اعتباره كذلك

التعامل معها

يتم تنفيذ هذه الكتلة

على الرغم من أن السطر 32في

معالج الالتقاط

ألقى استثناء. ثم الطريقة

ينتهي

الشكل | 11.5حاول ... ..fi nal ly ،catchآلية معالجة الاستثناءات (الجزء 2من 4.)

```
47
46      // demonstrate finally when no exception occurs
45      public static void doesNotThrowException()
50      {
49         try // try block does not throw an exception
48         {
52            System.out.println( "Method doesNotThrowException" );
51         } // end try
63         catch ( Exception exception ) // does not execute
62         {
61            System.err.println( exception );
60         } // end catch
59         finally // executes regardless of what occurs in try...catch
58         {
57            System.err.pri ntln( "تم تنفيذه أخيرًا في doesNotThrowException" ) ؛ // } النهاية في
56
55
54
53
64
```

This method does not throw any exceptions

This try block will execute all of its statements correctly

This catch handler will be skipped; no exceptions occur

This finally block still executes

يستمر التحكم في البرنامج هنا

doesNotThrowException") ؛ } // نهاية الأسلوب "System.out.println

doesNotThrowException 65 } // end class Usi rrgExceptions طريقة النهاية

تلر11.5حاول ... التقاط آلية معالجة الاستثناء النهائية. (الجزء 3من 4)

```
طريقة throwException
تم التعامل مع الاستثناء في طريقة throwException
تم تنفيذه أخيرًا في throwException
تم التعامل مع الاستثناء بشكل رئيسي
الأسلوب لا نوثرووإكسسيبشن
تم تنفيذه أخيرًا في doNotThrowException
نهاية الأسلوب doesNotThrowException
```

4 من 4 | 11.5 حاول ... اصطياد ... آلية معالجة الاستثناءات. (جزء

كومة الاسترخاء

- فك المكدس -عند طرح استثناء ولكن لم يتم اكتشافه في نطاق معين ، فإن مكدس استدعاء الأسلوب هو " Ainwound

جرت محاولة للقبض على الاستثناء في الكتلة الخارجية التالية.

جميع المتغيرات المحلية في طريقة الإلغاء تخرج عن النطاق والعودة إلى العبارة التي تم استدعاؤها في الأصل تلك الطريقة.

إذا تضمنت كتلة t ry هذا البيان ، فسيتم إجراء محاولة للقبض على الاستثناء.

إذا لم تقم كتلة t ry بإحاطة هذه العبارة أو إذا لم يتم اكتشاف الاستثناء ، فسيحدث فك المكدس مرة أخرى.

getStackTrace و pri ntStackTrace و getMessage

- طريقة الطباعة printStackTrace القابلة للرمي ينتج عنه ملف كومة تتبع لتيار الخطأ القياسي.
- مفيد في الاختبار والتصحيح.

Throwabl e طريقة getStackTrace تسترجع معلومات تتبع المكدس.

Throwabl e method getMessage تُرجع السلسلة الوصفية المخزنة في استثناء.

- لإخراج معلومات تتبع المكدس إلى تدفقات غير تدفق الخطأ القياسي:
  - استخدم المعلومات التي يتم إرجاعها من سباق getStackT وإخراجها إلى تيار آخر
  - استخدم أحد الإصدارات المحملة بشكل زائد من طريقة السباق pri ntStackT

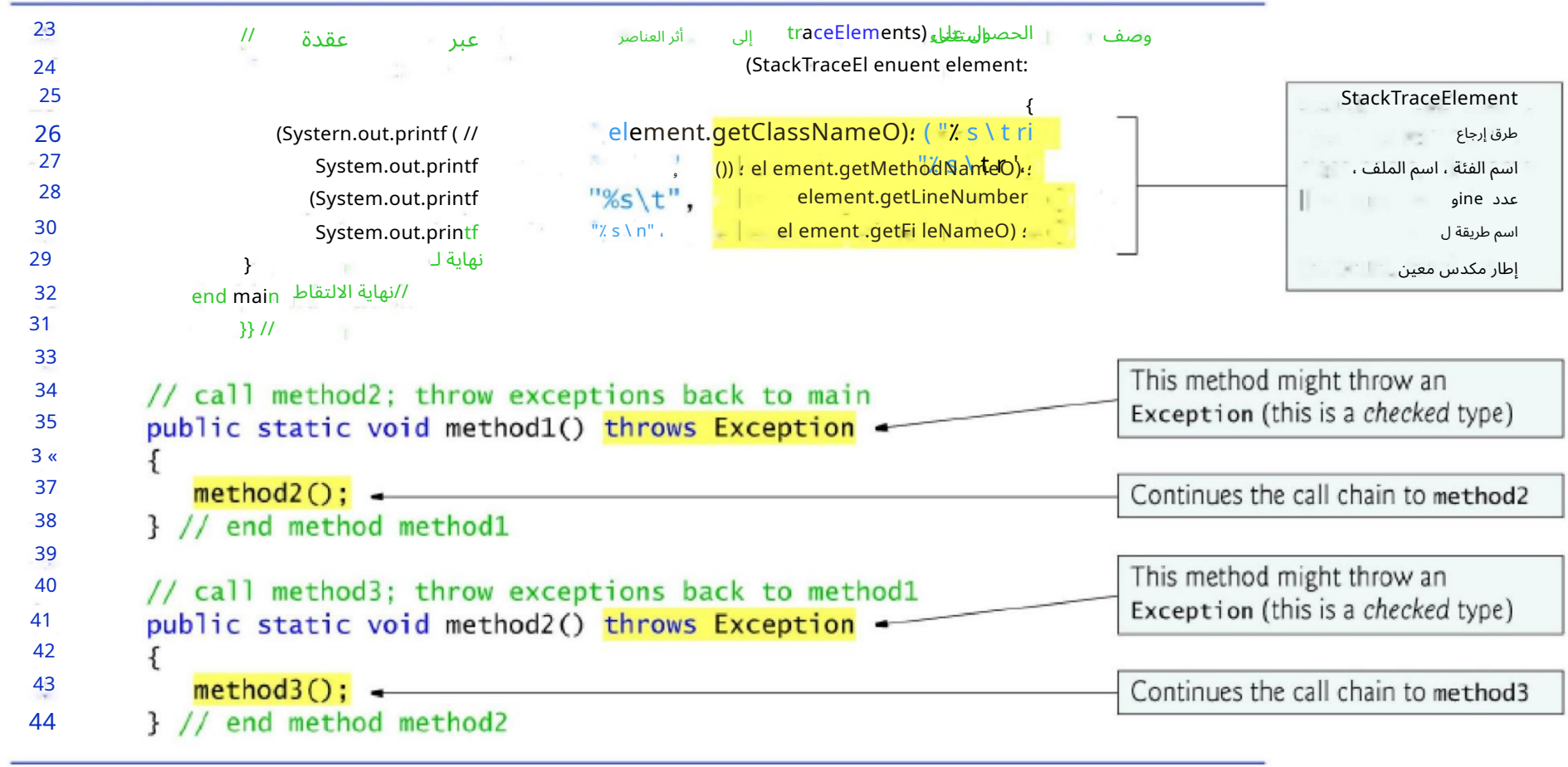
```
11.7: UsingExceptions.java الشكل//
printStackTrace. وgetMessageT getStackTrace طرق قابلة للرمي//

الطبقة العامة باستخدام الاستثناءات
{
ثابتة        فارغ        Stringf) args) رئيسي)        عام }

محاولة
{
طريقة // طريقة الاتصال
// إنهاء المحاولة}
{ (استثناء استثناء)catch        إمساك استثناء تم إلقاؤه في الطريقة//

.استثناء ا :(OegasseHteg.استثناءSystem.err.printf ( ، '٪ s \ n \ n ،،
printStackTraceO ؛ // طباعة تتبع مكدس الاستثناء

//الحصول على معلومات تتبع المكدس
():؛ ecarTkcatSteg.استثناءStackTraceEl ement [] traceEl ements =

getStackTrace:" )؛ تتبع المكدس منSystem.out، println ( "\ n
System.out. pri ntln ( ، 1Class \ t \ t File \ t \ t \ tLi ne \ tMethocT ) ؛
```

تبدأ سلسلة الاستدعاء التي ستؤدي إلى ملف
استثناء في هذا البرنامج

لا أحد من الآخر
طرق التقاط
استثناء: هكذا المكدس
غير ملفوف و
تم اكتشاف الاستثناء
هنا

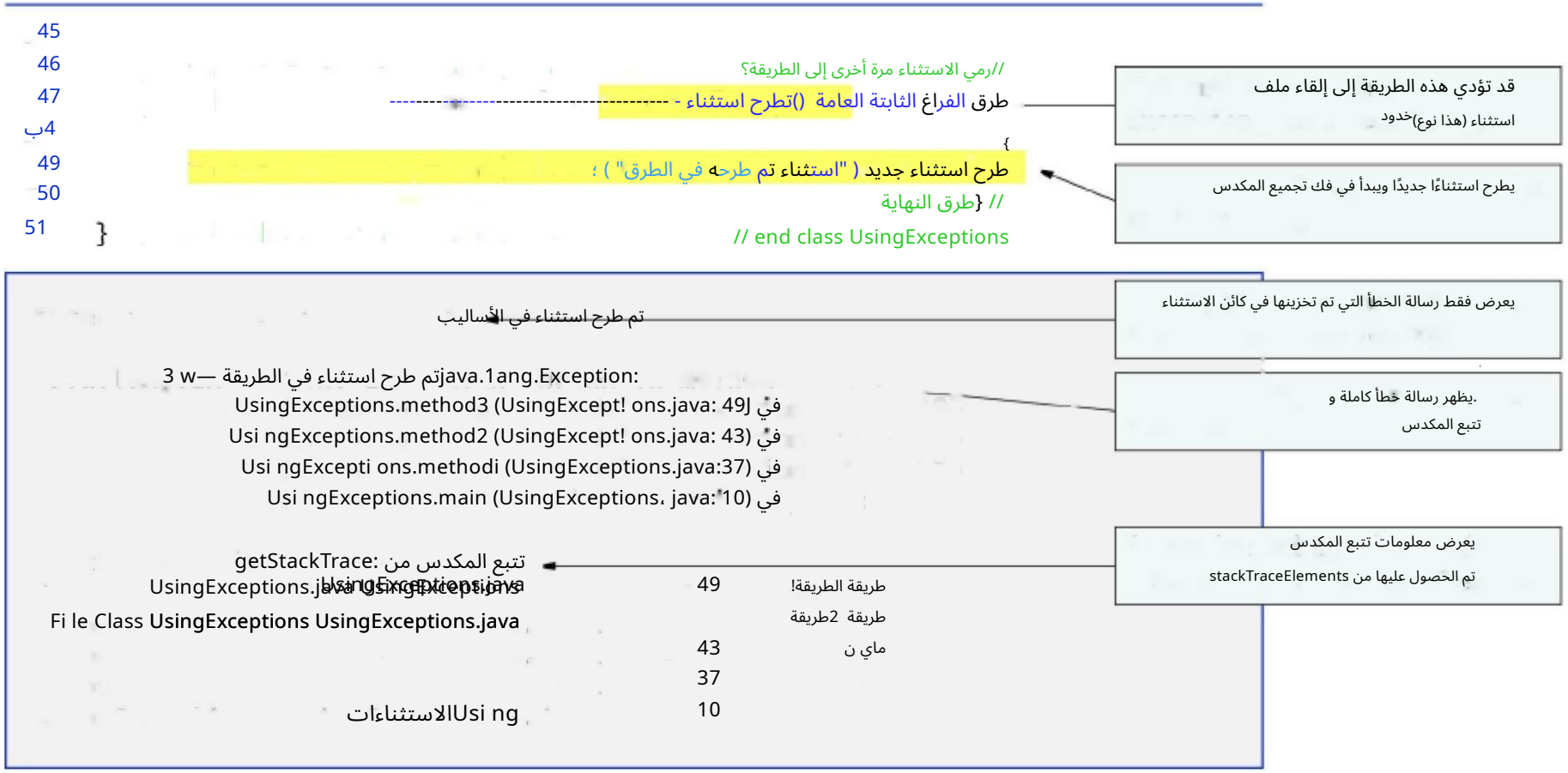
يحصل على مجموعة من
StackTraceElements

الشكل 11.7طرق الرمي getMessage وgetStackTrace و
طباعة تتبع المكدس. (الجزء الأول من 3.)



الشكل- H.7 | طرق الرمي getMessage وgetStackTrace

طباعة تتبع المكدس. (الجزء 2من 3.)



الشكل 11.7طرق الرمي وgetMessage وgetStackTrace وprintStackTrace(الجزء 3من 3.)

استثناءات بالسلاسل

- في بعض الأحيان تستجيب الطريقة لاستثناء من خلال طرح نوع استثناء مختلف خاص بالتطبيق الحالي .

- إذا طرحت كتلة catch استثناءً جديدًا ، فستفقد معلومات الاستثناء الأصلي وتتبع المكدس.

- لم توفر إصدارات Java السابقة أي آلية لالتفاف معلومات الاستثناء الأصلية بمعلومات الاستثناء الجديد.

- جعل هذا تصحيح مثل هذه المشاكل صعبًا بشكل خاص.
- تسمح الاستثناءات المتسلسلة لكائن استثناء بالحفاظ على معلومات تتبع المكدس الكاملة من الاستثناء الأصلي.

```
1   //التين ، :11.8استخدام السلسلة. java
2   //الاستثناءات بالسلاسل.
3
    UsirigChainedException فئة عامة4
5   {
7   سلسلة mainf ثابتة عامة (args []
6   {
8   محاولة
9   {
10  منهجي. // طريقة الاتصال
11  // {إنهاء المحاولة
12  catch(استثناء استثناء) // الاستثناءات التي تم إلقاؤها من methodi
13  {
14  استثناء. pri ntStackT السباق J O
15  } // end catch
10  } // end main
17
```

يمسك الاستثناء المتسلسل ويعرض تتبع المجموعة

3.) استثناءات متسلسلة . ILS(الجزء الأول من

```
18  //طريقة الاتصال ؛ 2طرح الاستثناءات مرة أخرى إلى أسلوب الفراغ الثابت
23  العام ) يطرح استثناء {
22
21  حاول {
20
19  طريقة C try catch } // end method2 ؛ // call method2 استثناء استثناء} // استثناء من الطريقة؟ {
27
26
25
24  طرح استثناء جديد ( "استثناء تم طرحه في طريقة" ، استثناء) ؛ } // end method methodi // { catch
40  end
39
38
37  //طريقة الاتصال ؛ 3طرح الاستثناءات مرة أخرى إلى () methodi public static void method2 تلقي استثناء {
36
35
34  حاول {
33
32  3 { استثناء من الطريقة // استثناءً) metlhod3O. // Tall method3 } // end try catch C
31
30
29
28  طرح استثناء جديد "تم طرح استثناء في الطريقة "، 2استثناء) ؛ } // end method2 // { catch end
42
41
```

يخلق ملف
استثناء مع رسالة مخصصة ؛
سلاسل الاستثناء الذي تم
طرحه بواسطة method2

يخلق ملف
استثناء مع رسالة مخصصة ؛
سلاسل الاستثناء الذي تم
طرحه بواسطة الطريقة 3

تلميح 11.8 الاستثناءات بالسلاسل. (الجزء 2 من 3 J.

```
43
44                                  //رمي استثناء عام باطل ثابت      العودة إلى الطريقة 2
45                                                                  طريقة 30رميات استثناء
46                                                                  {
47                                                                  طرح ExceptionCالجديد "استثناء تم طرحه في الطريقة ؛ ( "3
48                                                                  //طرق النهاية}
49 } // إنهاء الفئة UsingChainedExceptions
```

الاستثناء الأصلي

```
java.1ang.Exception: استثناء تم طرحه في الطريقة
في (UsingChainedException $ .java: 10)في Usi ngChairedExceptions.methodl (Usi ngChai nedExceptions.java:27)
Usi ngChainedExceptions.main
* ---------- بسبب: java.lang.Exception: تم طرح استثناء في الطرق
في Usi ngChainedExceptions.method2 (Usi ngChai nedExceptions.java:40)
في Usi ngChai redExceptions.methodl (Usi ngChai nedExceptions.java:23)
1 ... أكثر السبب: java. لانج. استثناء: تم طرح استثناء في الطرق -في (Usi ngChai nedExceptions.java:47)
Usi ngChai nedExceptions.method ^ (Usi ngChai nedExceptions.java:36)في UsingChainedExceptions.method3

2أكثر ، ..
```

لاحظ أن الاستثناءات المتسلسلة تظهر في معلومات تتبع الكومة

الشكل الأول | 1.8الاستثناءات بالسلاسل. (الجزء 3من 3.)

التأكيدات

- عند تنفيذ فصل دراسي وتصحيحه ، يكون الأمر كذلك
من المفيد أحيانًا ذكر الشروط التي يجب أن تكون صحيحة في نقطة معينة في طريقة ما.

تساعد التأكيدات في ضمان صلاحية البرنامج عن طريق اكتشاف الأخطاء المحتملة وتحديد الأخطاء المنطقية المحتملة أثناء التطوير.

- الشروط المسبقة والشروط اللاحقة نوعان من التأكيدات.

تأكيدات (تابع)

تتضمن Javaنسختين من جملة التأكيد للتحقق من صحة التأكيدات برمجيًا.

- يؤكد تقييم التعبير الأبولي ، وإذا كان ، fal se
يطرح AssertionError(فئة فرعية من الخطأ).

التعبير- الجزء

التعبير هو التعبير !: ، fa Ise

- يطرح خطأ AssertionErrorإذا

تأكيد •يقيّم

ويطرح خطأ AssertionErrorمع التعبير 1التعبير!

كرسالة خطأ إذا كان التعبير 1هو .f al se

يمكن استخدامها لتنفيذ الشروط المسبقة برمجيًا
والشروط اللاحقة أو للتحقق من أي حالات وسيطة أخرى
التي تساعدك على التأكد من أن التعليمات البرمجية الخاصة بك تعمل بشكل صحيح.

بيان التأكيد في Java

في ، Javaيبدأ بيان التأكيد بالكلمة الأساسية "الأصول" متبوعة بتعبير منطقي.

يمكن كتابة جملة التأكيد في Javaبطريقتين:

تأكيد التعبير

تأكيد التعبير: تعبير ؛ 2

في كلتا الطريقتين ، تكون التعبيرات المستخدمة مع الكلمة الأساسية Assertهي التعبيرات المنطقية.

```
فئة رئيسية {
main static void main (String args []) {
سلسلة []عطلات نهاية الأسبوع ={"الجمعة" ، "السبت" ، "الأحد"} ؛ التأكيد على طول عطلة نهاية الأسبوع ؛ 2 ==

System.out.println("لدينا " + weekends.length + "أيام نهاية الأسبوع في الأسبوع") ؛
}
}
```

انتاج |

```
استثناء في موضوع java.lanq.AssertionError"الرئيسي"
```

```
13.9: AssertTest.java الشكل1 //
2 // يوضح التأكيد statement
3 استيراد java.util.Scanner ؛ 4
5 اختبار AssertTest للفئة العامة 6
{
الرئيسية العامة الثابتة الفراغ (سلاسل سلسلة [])
{
إدخال الماسح =الماسح الجديد (System.in) ؛

System.out.print( "أدخل رقمًا بين 0 و 10:" ) ؛ int = input .next I nt () ؛

// assert أن القيمة المطلقة <= 0
assert ( رقم <= 10) رقم && 0 =< رقم
System.out.printf( "لقد أدخلت٪ / d \ n" رقم )؛
} // end main
} // نهاية فئة AssertTest
```

```
أدخل رقمًا بين 0و 10: 5
لقد دخلت 5
```

```
أدخل رقمًا بين 0و 10: 50
استثناء في الموضوع "الرئيسي" java.lang.Assertion خطأ: رقم غير صالح: 50
في AssertTest.main (AssertTest.java:15)
```