

معالجة الاستثناءات Exception Handling



جامعة طرابلس - كلية تقنية المعلومات

د. عبد الحميد الواعر

معالجة الاستثناءات

Exception Handling

الاستثناءات (Exceptions)

الاستثناء هو عبارة عن مشكلة تحدث خلال تنفيذ البرنامج ويتسبب بمقاطعة تسلسل تنفيذ البرنامج وهو يمكن يحدث لأسباب مختلفة منها:

- المستخدم قام بأدخال بيانات غير صالحة.
- لا يمكن العثور على الملف المراد فتحه لقراءة بيانات منه.
- قطع الاتصال بالشبكة خلال عملية الاتصال.
- عند تشغيل الجافا يحدث نفاذ للذاكرة (JVM has run out of memory)

البرنامج التالي يقوم بإدخال عدد صحيح عن طريق لوحة المفاتيح ثم إيجاد مربعه وطباعته.

```

package Lecture4.ExceptionHandling;

import java.util.Scanner;

public class SequareExample {
    public static void main (String args []){

        Scanner input = new Scanner(System.in);
        int x = input.nextInt();
        System.out.println("The swquare of "+x+" is "+x*x);

    }
}

```

في حالة ادخال قيمة غير عدد صحيح مثل ادخال حروف سينتج عن عملية الادخال هذه استثناء كما هو موضح أدناه:

```
run:
```

```
abc
```

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

```
at java.util.Scanner.throwFor(Scanner.java:909)
```

```
at java.util.Scanner.next(Scanner.java:1530)
```

```
at java.util.Scanner.nextInt(Scanner.java:2160)
```

```
at java.util.Scanner.nextInt(Scanner.java:2119)
```

```
at Lecture4.ExceptionHandling.SquareExample.main(SquareExample.java:15)
```

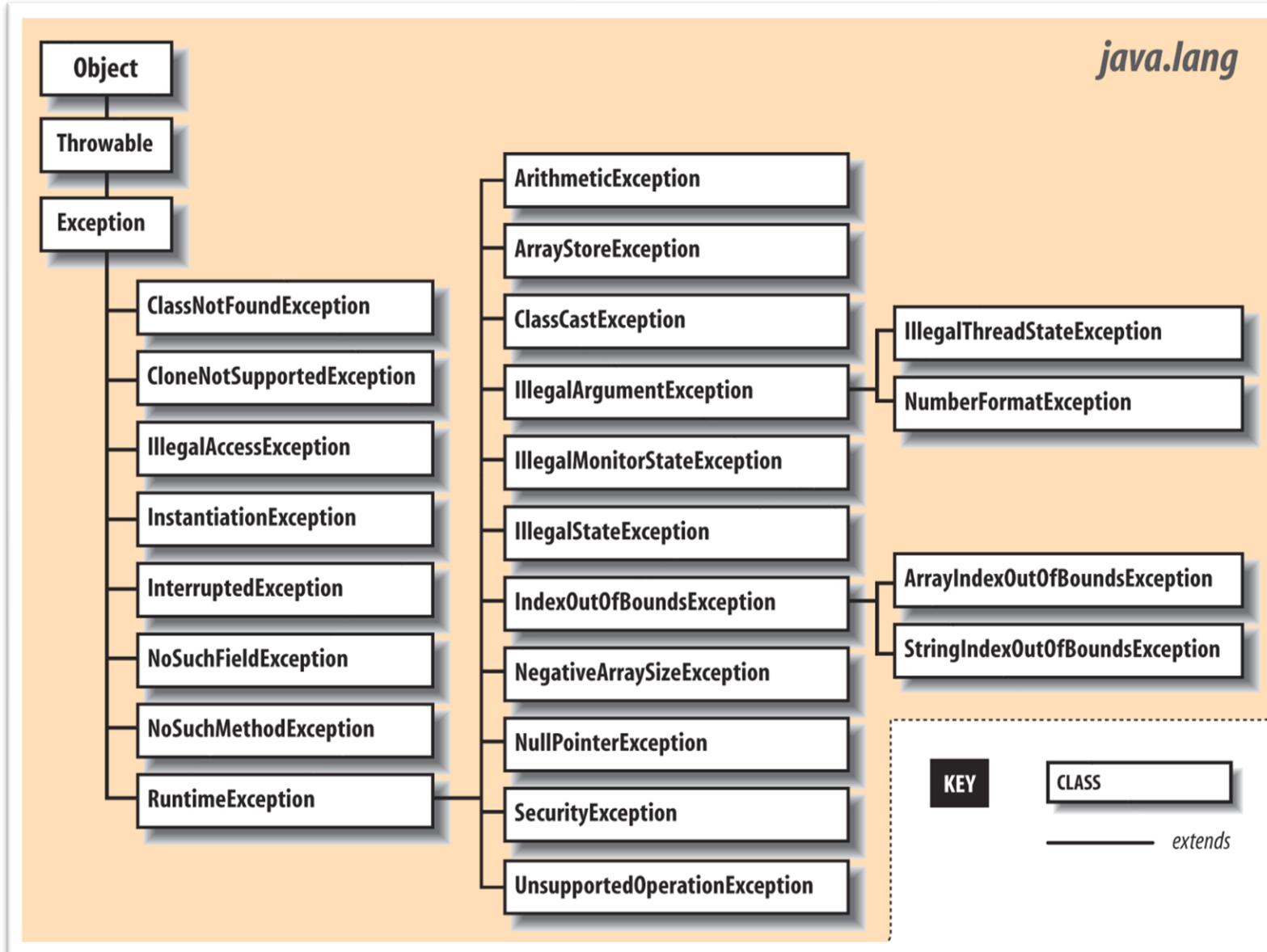
```
Java Result: 1
```

InputMismatchException

Exception Object



- عند حدوث خطأ في البرنامج تقوم method التي حدث بها الخطأ بإنشاء exception object يحتوي على معلومات عن الخطأ مثل نوعه وكذلك معلومات عن حالة البرنامج عند حدوث الخطأ.
- لغة جافا توفر مجموعة من exception classes وكذلك تسمح بكتابة أخرى جديدة.



يوجد هناك نوعان من Exceptions :

- Checked exception

- Unchecked exception

Checked exception



وهي التي تحصل أثناء : compile time

- ClassNotFoundException
- IllegalAccessException
- NoSuchFieldException
- IOException

Unchecked exception



وهي التي تحصل أثناء :run time

- ArithmeticException
- ArrayIndexOutOfBoundsException
- NullPointerException
- IllegalArgumentException

معالجة الاستثناءات في لغة جافا تتم في خطوتين:

- وضع جزء البرنامج الذي ممكن يسبب في أستثناء (Exception) داخل `try{} Block`.
- وضع جزء البرنامج الذي يتعامل مع الاستثناء داخل `catch{} Block`.

عند حدوث استثناء يتم تنفيذ الجمل الموجودة داخل `catch{} Block` ثم يستمر البرنامج في التنفيذ ولا يتوقف.

في المثال السابق الدالة `nextInt()` تقوم بأرسال أستثناء عند أذخال أي قيمة لاتكون من النوع الصحيح، وبم أنه لم تتم إلتقاط هذا الاستثناء ومعالجته ، جافا تقوم بإنهاء تنفيذ البرنامج وطباعة

```
package Lecture4.ExceptionHandling;

import java.util.Scanner;

public class SequareExample {
    public static void main (String args []){

        Scanner input = new Scanner(System.in);
        int x = input.nextInt();
        System.out.println("The swquare of "+x+" is "+x*x);

    }

}
```

مايفيد بالخطأ الذي حدث.

لغة جافا تستخدم Exception Class في التعامل مع الاستثناءات التي تحدث عند تنفيذ البرامج، وباستخدامها يمكن كتابة برامج يتم فيها تفادي هذه الاستثناءات والاستمرار في تنفيذ البرنامج.

للتعامل مع الاستثناء الذي حدث في البرنامج السابق سيتم تعديله بالشكل التالي:

الامر الذي ممكن أن
يسبب في حدوث أستثناء

الاوامر المطلوب تنفيذها
عند حدوث أستثناء

```

package Lecture4.ExceptionHandling;

import java.util.InputMismatchException;
import java.util.Scanner;

public class SequareExample {

    public static void main(String args[]) {

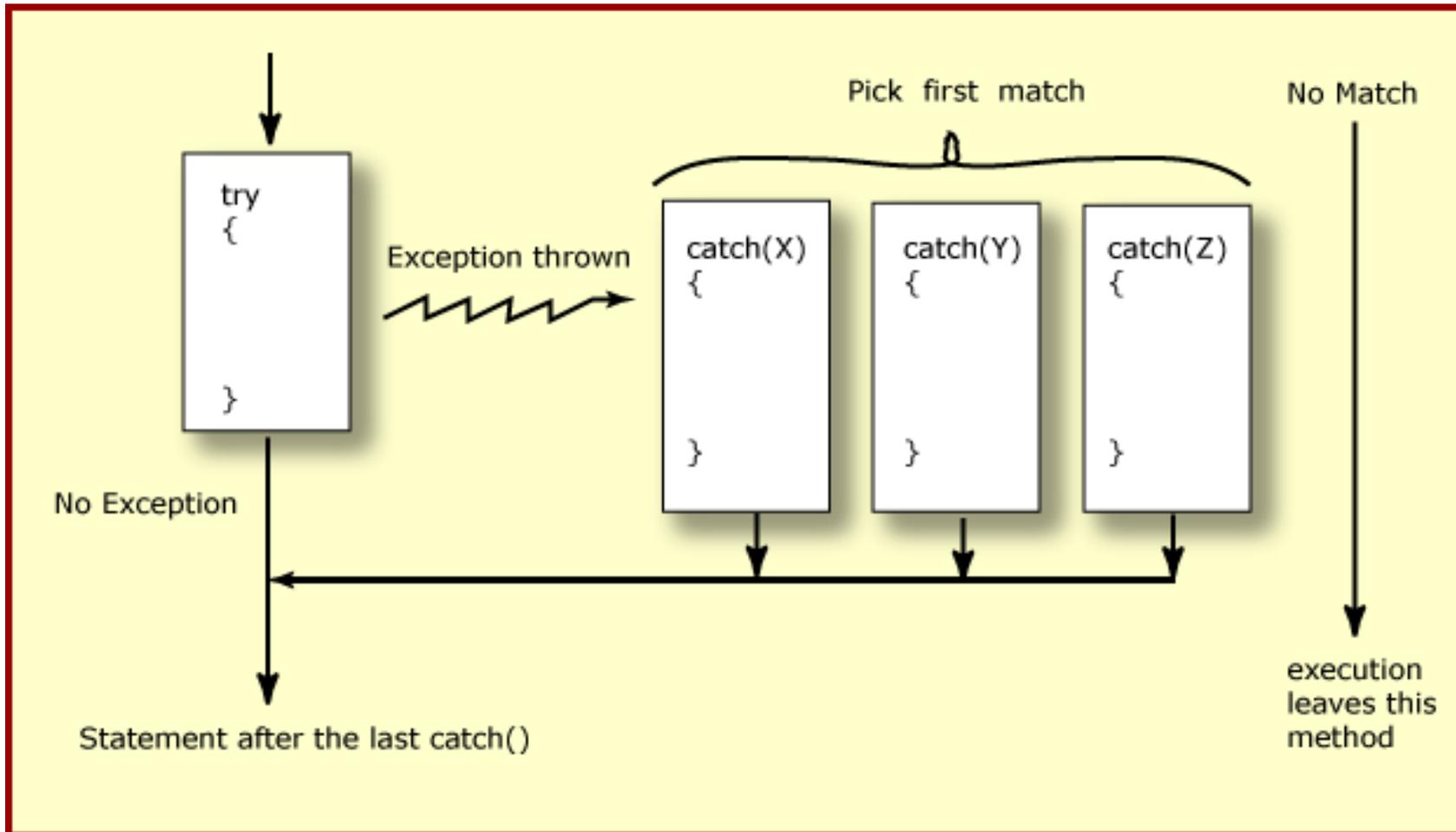
        Scanner input = new Scanner(System.in);
        try {
            int x = input.nextInt();
            System.out.println("The swquare of " + x + " is " + x * x);

        } catch (InputMismatchException ex) {
            System.out.println("You entered bad data.");
            System.out.println("Run the program again.");
        }
        System.out.println("Good-by" );
    }
}

```

```
Try
{
    الاوامر المسببة للاستثناءات
}
catch ( SomeExceptionType ex ) {
    الاوامر المعالجة لهذا النوع من الاستثناءات
}
catch ( AnotherExceptionType ex ) {
    الاوامر المعالجة لهذا النوع من الاستثناءات
}
catch ( YetAnotherExceptionType ex ) {
    الاوامر المعالجة لهذا النوع من الاستثناءات
}
باقي أوامر البرنامج
```

الرسم التالي يوضح كيفية عمل try and catch



البرنامج التالي يقوم بأدخال قيمة عددين صحيحين عن طريق لوحة المفاتيح ثم إجراء عملية القسمة بينهما. ويتم فيه معالجة استثنائين الأول ناتج عن إدخال قيمة غير مقبولة والآخر عند القسمة على صفر.

```

package Lecture4.ExceptionHandling;

import java.util.InputMismatchException;
import java.util.Scanner;

public class DivisionExample {

    public static void main(String[] a) {
        Scanner scan = new Scanner(System.in);
        int num = 0, div = 0;

        try {
            System.out.print("Enter the numerator: ");
            num = scan.nextInt();
            System.out.print("Enter the divisor : ");
            div = scan.nextInt();
            System.out.println(num + " / " + div + " is " + (num / div) );
        } catch (InputMismatchException ex) {
            System.out.println("You entered bad data.");
            System.out.println("Run the program again.");
        } catch (ArithmeticException ex) {
            System.out.println("You can't divide " + num + " by " + div);
        }
    }
}

```

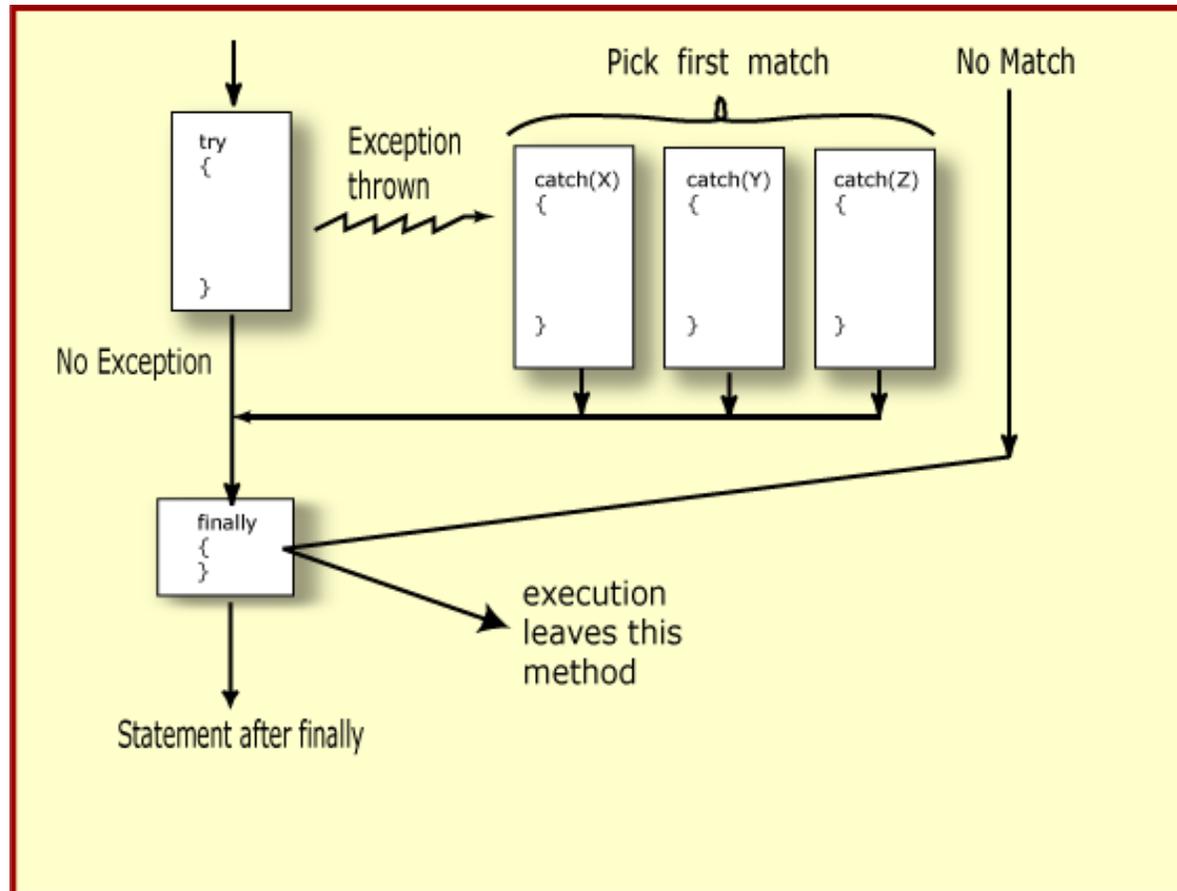
معالجة الاستثناء الناتج عن
أدخال غير مقبول

معالجة الاستثناء الناتج عن
القسمة على صفر

The Finally {} Block



يستخدم `finally {}` block للتأكيد تنفيذ مجموعة من الاوامر بغض النظر عن تنفيذ جمل `try` أو `catch` الرسم التالي يبين طريقة تنفيذها:



البرنامج التالي يوضح طريقة استخدام Finally block.

```

package Lecture4.ExceptionHandling;

public class FinallyExample {
    public static void main(String args[]){
        int a[] = new int[2];
        try{
            System.out.println("Access element three :" + a[3]);
        }catch(ArrayIndexOutOfBoundsException e){
            System.out.println("Exception thrown  :" + e);
        }
        finally{
            a[0] = 6;
            System.out.println("First element value: " +a[0]);
            System.out.println("The finally statement is executed");
        }
    }
}

```

Throws Keyword



تستخدم عند الاعلان عن method ما لتوضح أنها قد تسبب في حصول Exception وعلى مستخدم هذه method أن يقوم بالتعامل مع هذا Exception.

```
class Demo
{
    static void throwMethod() throws NullPointerException
    {
        System.out.println ("Inside throwMethod");
        NullPointerException ex = new NullPointerException ("Demo");
        throw ex;
    }
    public static void main(String args[])
    {
        try
        {
            throwMethod();
        }
        catch (NullPointerException exp)
        {
            System.out.println ("The exception get caught" +exp);
        }
    }
}
```

Throw Keyword



عند حدوث خطأ في البرنامج في لغة جافا يتم إرسال Exception يبين أن هناك خطأ قد حدث ويتم ذلك عن طريق استخدام Throw Keyword.

```
/* In this program we are checking the student age
 * if the student age<12 and weight <40 then our program
 * should return that the student is not eligible for registration.
 */
public class ThrowExample {
    static void checkEligibility(int stuage, int stuweight){
        if(stuage<12 && stuweight<40) {
            ArithmeticException ex = new ArithmeticException("Student is not eligible for registration");
            throw new ArithmeticException("Student is not eligible for registration");
        }
        else {
            System.out.println("Entries Valid!!");
        }
    }

    public static void main(String args[]){
        System.out.println("Welcome to the Registration process!!");
        checkEligibility(10, 39);
        System.out.println("Have a nice day..");
    }
}
```

User Defined Exceptions



في بعض الاحيان يحتاج مطوري البرامج إلى إنشاء exception object الخاص بهم ويتم ذلك بعمل

. Exception Class Jextend

```
class MyException extends Exception{
    String str1;
    MyException(String str2) {
        str1=str2;
    }
    public String toString(){
        return ("Output String = "+str1) ;
    }
}
class CustomException{
    public static void main(String args[]){
        try{
            throw new MyException("Custom");
            // I'm throwing user defined custom exception above
        }
        catch(MyException exp){
            System.out.println("Hi this is my catch block") ;
            System.out.println(exp) ;
        }
    }
}
```

الاختلاف بين كل من Throws وThrow



- Throws clause يستخدم للإعلان عن Exception بينما Throw تستخدم في إظهار Exception.
- Throws تأتي متبوعة ب Exception class name بينما Throw تأتي متبوعة ب instance variable.

```
public void sample() throws ArithmeticException{
    //Statements

    .....

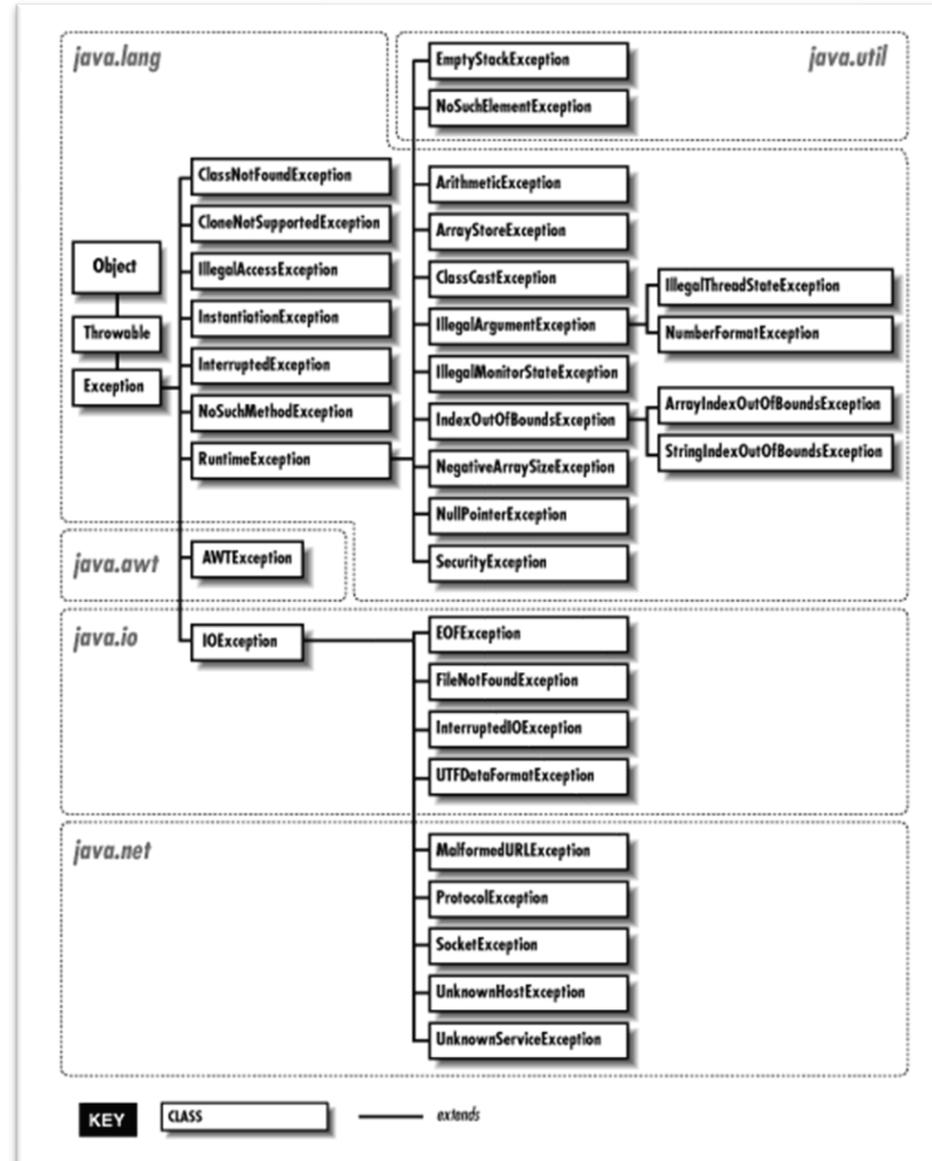
    //if (Condition : There is an error)
    ArithmeticException exp = new ArithmeticException();
    throw exp;
    ...
}
```

- Throws تسمح بالاعلان عن أكثر من Exception بينما Throw تسمح فقط بأظهار Exception واحد.

```
throws IOException, ArithmeticException, NullPointerException.
```

```
throw new IOException("Connection failed!!")
```

الشكل التالي يوضح الاستثناءات المنبثقة عن Exception Class.



شكراً