جَامعة المَنارة
MANARA UNIVERSITY

كلية الهندسة المعلوماتية

برمجة 3

Java Programming

ا. د. علي عمران سليمان

محاضرات الأسبوع السادس

Exceptions Handling and IOFile

الفصل الثاني 2024-2025

content

- Exceptions Handling .
- Exception Classes
- Handling Multiple Exceptions.
- The finally Clause.
- Throwing Exceptions .
- Input Output File,
- ArrayList. .Using an ArrayList. .Create an ArrayList.
- Methods ArrayList .
- Using Objects with Graphics.
- Displaying Text and Images Using Labels.

References

- Deitel & Deitel, Java How to Program, Pearson; 10th Ed(2015)
- د.علي سليمان، بنى معطيات بلغة JAVA، جامعة تشرين 2013-2014

Exceptions Handling 1

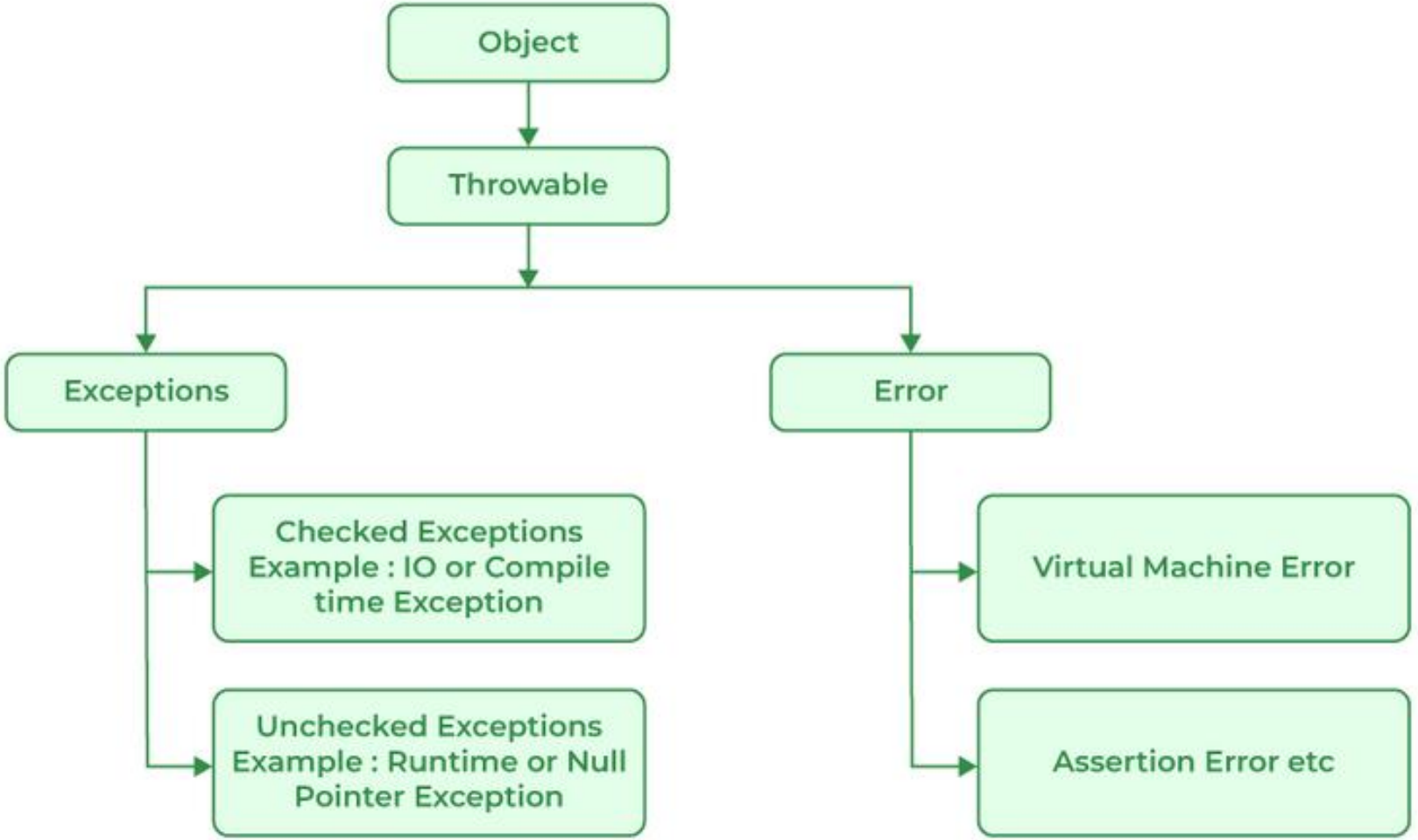
- الاستثناء هو كائن يتم إنشاؤه ورميه (إلقائه) نتيجة لخطأ أو حدث غير متوقع خلال التنفيذ.
- مصدرالإستثناءات قد تحدث بسبب المستخدم (User), أو المبرمج (Programmer), أو بسبب الأجهزة المستخدمة (Physical Resources).
- تقسيم الإستثناءات إلى ثلاث أصناف أساسية:
  1- Checked Exception تعني خطأ برمجي يحدث أثناء ترجمة البرنامج (أي قبل تشغيل الكود).
  2- Unchecked Exception تعني خطأ منطقي يحدث أثناء تشغيل البرنامج.
  3- Error تعني خطأ يحدث بسبب الجهاز الذي نحاول تشغيل البرنامج عليه.

الاول: تعريف متغير من نمط واسناد قيمة من نمط غير مطابق ولايمكن قسرها للنمط المسند له.

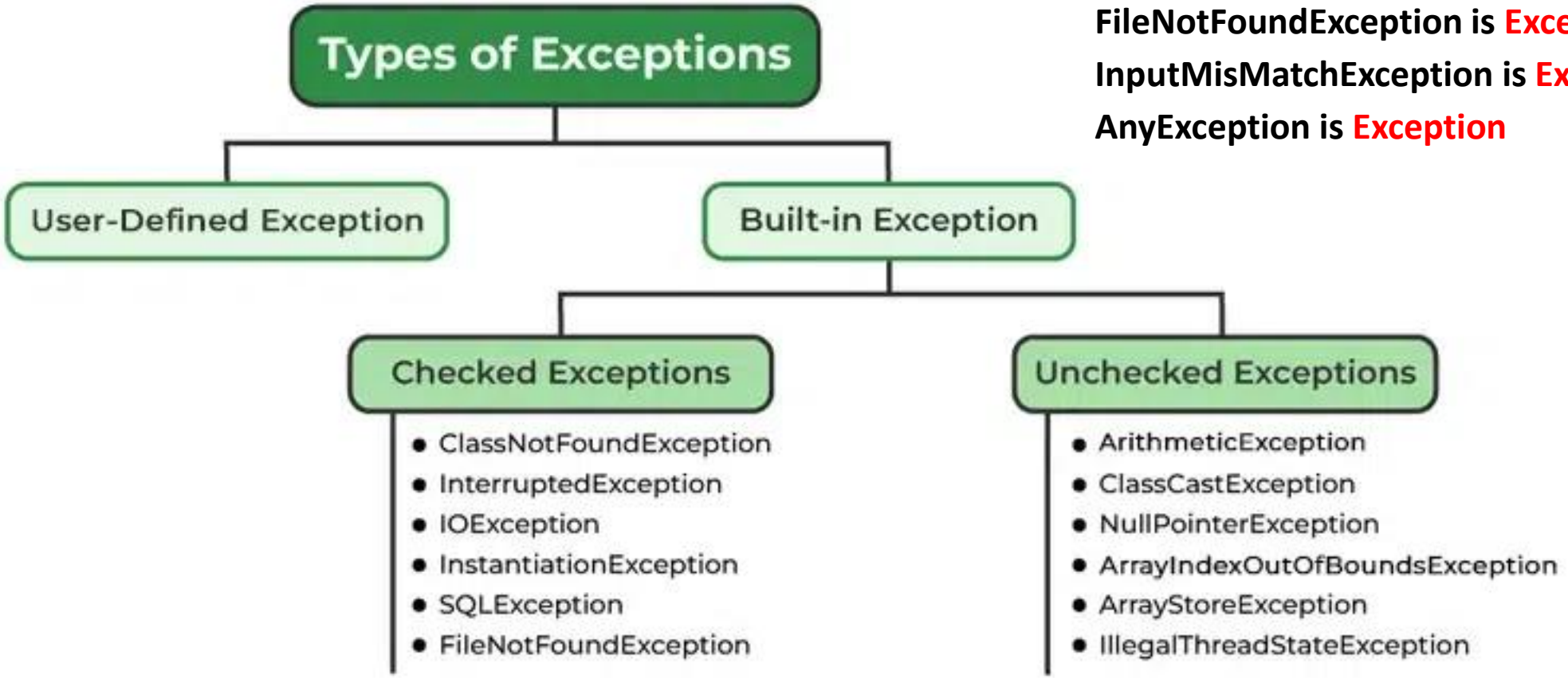
الثاني: يسمى Runtime Exception, وهو يتضمن الـ (Programming Bugs) والتي تعني أخطاء منطقية Logical Errors أو أخطاء سببها عدم إستخدام الأشياء المعرفة في لغة البرمجة بالشكل الصحيح (APIs errors).

الثالث: Error تعني خطأ يحدث بسبب الجهاز الذي نشغل البرنامج عليه، مثلاً إمتلأ ذاكرة الحاسب الذي يعمل عليه البرنامج, عندها ستظهر الرسالة التالية JVM is out of Memory لهذا السبب يتم حفظ مراحل الانجاز مما يعطي المستثمر الثقة لامكانية استعادة البيانات عند توقفه.

Exceptions Handling 2



Exceptions Handling 3



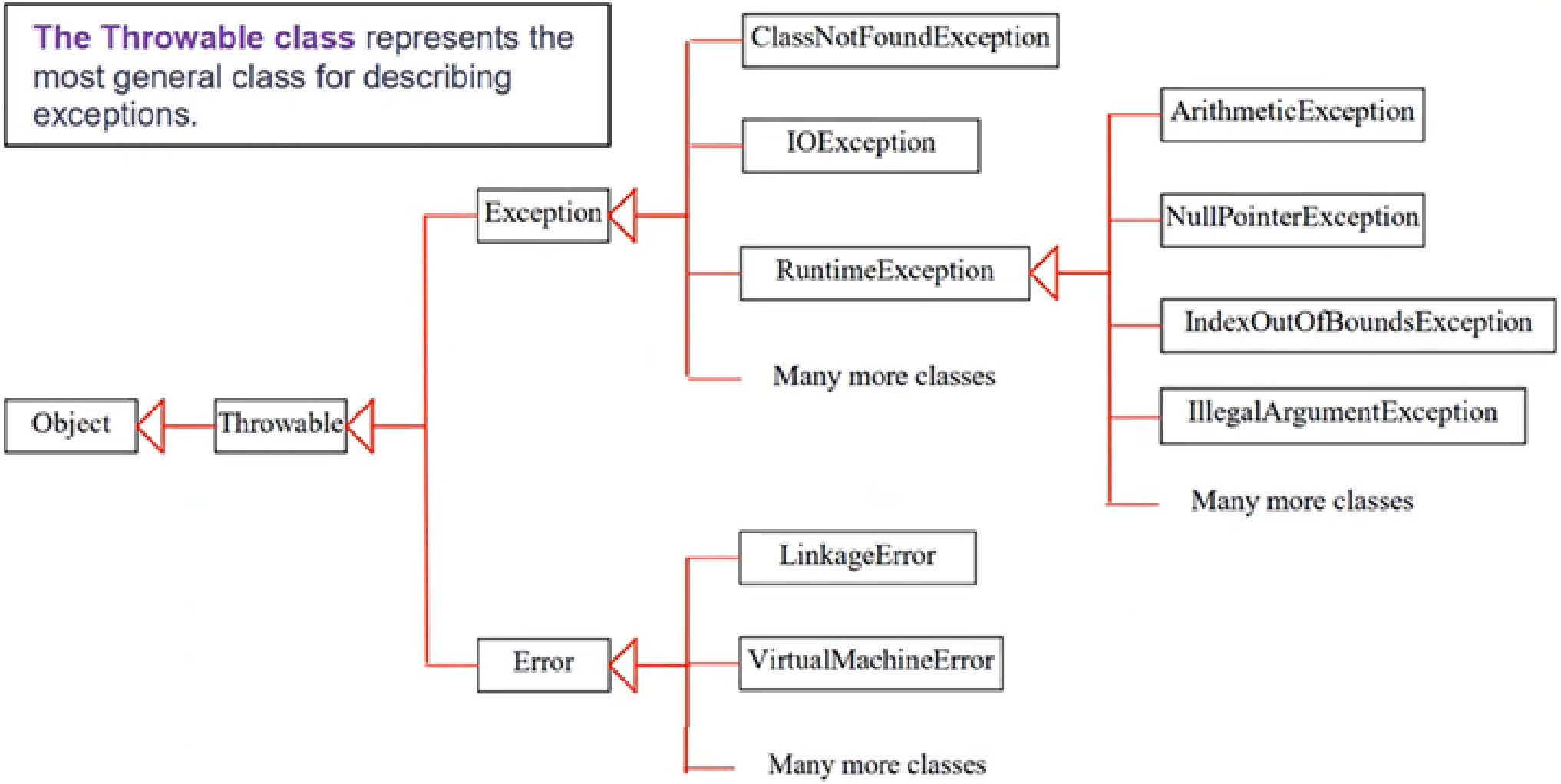
FileNotFoundException is Exception

InputMisMatchException is Exception

AnyException is Exception

Exception Classes

The Throwable class represents the most general class for describing exceptions.



Exceptions Handling 4

- تقسم الإستثناءات إلى عدة أنواع، وكل منها تم تمثيله في صنف منعزل، وجميعها ترث الصنف الأساس Exception
- الصنف Throwable يورث الصنف Exception وأية صنف يرث Exception هو صنف يمثل إستثناء معين.
- الصنف Throwable يورث الصنف Error المتضمن للأخطاء التي سببها الأجهزة.
- يُقال إن الاستثناء أُلقي"thrown" ويجب أن يلتقط Catch.
- تقع على عاتق المبرمجين مسؤولية كتابة التعليمات البرمجية التي تتوقع الاستثناءات وتعالجها.
- الاستثناءات التي لم تتم معالجتها ستؤدي إلى تعطيل crash البرنامج.
- تسمح لك Java بإنشاء معالجات للاستثناءات خاصة بك.

```
int x = 10 , y = 0 ;

System.out.println (x/y);
```

Divide by Zero

Exceptions Handling 5

- معالج الاستثناءات هو مقطع برمجي يستجيب للاستثناءات.
- لمعالجة الاستثناءات ، تستخدم العبارات try{ ... } catch() { ... }

```
try
  {      (try block statements... Or Checked block)   }
catch (ExceptionType ParameterName)
{        (catch block statements... Or Handling block )        }
```

- أولاً ، تشير الكلمة المفتاحية try إلى بداية الكتلة المتوقع حدوث الخطأ بها وتعرف بـ **Protected Code**.
- بعد كتلة try التي يتم منها قذف أو رمي الاستثناء عند حدوث الخطاء، تظهر كتلة catch (معالجة الخطاء **Error Handling Code**) التي يتم التقاطه ماتم رميه.

Exceptions Handling 6

- تبدأ جملة catch بالكلمة المفتاحية catch:

- catch (ExceptionType ParameterName)

  - ExceptionType هو اسم فئة الاستثناء.
  - ParameterName هو اسم متغير يشير إلى كائن الاستثناء المرمى من الكتلة try.

- يُعرف كود الكتلة التي تلي catch ببلوك catch أو المعالجة.
- يتم تنفيذ الكود الموجود في كتلة catch إذا توافق مع الاستثناء المرمى من الكتلة try.

تم تصميم هذا المقطع للتعامل مع FileNotFoundException إذا تم إلقاؤه.

```
try
{    File file = new File ("MyFile.txt"); Scanner inputFile = new Scanner(file); }
catch (FileNotFoundException e)
{     System.out.println("File not found.");          }
```

- في حال وجود خطأ تبحث Java Virtual Machine عن عبارة catch يمكن أن تتعامل مع الاستثناء.

Polymorphic References To Exceptions

```
Try
    {
        number = Integer.parseInt();
     }

    catch (Exception e)
     {
        System.out.println("The following error occurred: " + e.getMessage());
     }
```

- تطرح الطريقة parseInt() من الصنف Integer كائن الاستثناء المشتق من الصنف NumberFormatException وهو بدوره مشتق من الصنف Exception

Handling Multiple Exceptions

- قد تكون العبارات الموجوده في كتلة try قادرًا على إلقاء أكثر من نوع واحد من الاستثناءات.
- يجب كتابة عبارة catch لكل نوع من أنواع الاستثناءات التي يمكن القائها.
- سيقوم JVM بتشغيل أول جملة catch متوافقة تم العثور عليها.
- يجب أن يتم سرد جمل الالتقاط من الأكثر تحديدًا إلى الأكثر عمومية.
- يمكن أن يكون هناك العديد من أنواع الالتقاط متعددة الأشكال.
- قد تحتوي جملة try على عبارة catch واحدة فقط لكل نوع معين من الاستثناءات.

Exceptions Handling

```
//A Class that represents user-defined exception
public class LogicalException extends Exception {
public LogicalException(String m) { super(m);     } } //end class LogicalException

public class RecRevException {

private  int  length,width;

public RecRevException(){length=1;width=1;}

public static int peri(int l, int w){ return 2*(w+l); }

public static double dia(double l, double w){return Math.sqrt(w*w+l*l);}

public static int area(int l, int w){ return w*l; }} // end class RecRevException

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

public class RecRevExceptionTest{

public static void main(String[] args) {
RecRevException Rect= new RecRevException();
Scanner input=new Scanner(System.in);
```

Exceptions Handling

```
int l,w,a;String str="10";
try {int[] b = new int [5];
System.out.println("input L ");l=input.nextInt();
System.out.println("input w ");w=input.nextInt();
if (l < 0 &&  w <0) {throw new LogicalException("The dimensions of the rectangle must be
positive values. ");}
if (l < 0 ||  w <0)
{throw new Exception("One of the dimensions of the rectangle is a negative value. ");}
System.out.println("the l div w =  "+ l/w);
if (l < w) {throw new LogicalException( "The entrance length < width.? ");}
System.out.println(b[2]);//if index not between[0,4]Array Index Out Of Bounds
a=Integer.parseInt(str);//NumberFormatException
System.out.println("\nthe perimeter ob1= "+Rect.peri(l, w));
System.out.println("\nthe dia ob1=  "+Rect.dia(l, w));
System.out.println("\nthe area ob1= "+Rect.area(60, 44));
} // end try block
catch(NumberFormatException e)   {System.out.println( " is not a number."); }

catch(ArithmeticException ed)
  {System.out.println("The process is not allowed "+ ed.getMessage()); }
```

Exceptions Handling

```
catch(InputMismatchException e1)
  {System.out.println("Bad number InputMismatch."); }
catch (ArrayIndexOutOfBoundsException eb)
  {System.out.println("Array Index Out Of Bounds 0 t0 4:" +eb.toString());}
catch (LogicalException eb) {System.out.println(" L>W :" +eb.toString());}
catch (Exception e)
  {System.out.println(e.toString());  } } // end main
} //end class  // e.toString() return ExeceptionName and Message
  // e.getMessage()return message sanded to constructor (over loading)
```

1- عند وضع input L or w قيم غير عددية سنجد الخطأ التالي:

```
Exception in thread "main" java.util.InputMismatchException
Bad number InputMismatch.
```

2- عند وضع كل من w=0 & l=0 سيتم نداء الاستثناء المعرف من قبل المستخدم LogicalException وارسال رسالة إلى الباني مفادها البعدان يجب أن يكونان موجبات ويتم تنفيذ catch الخاصة بهذا الاستثناء لإخراج رسالة ومناداة eb.toString() لطباعة الاسم والرسالة المرسلة لباني Exception:

```
Entrance length<0 and width<0 :LogicalException: The dimensions of the rectangle must be positive values.
```

3- عند وضع كل من w=0 || l=0 سيتم نداء الاستثناء Exception وارسال رسالة إلى الباني مفادها أحد البعدين قيمته سالبة ويتم تنفيذ catch الخاصة بهذا الاستثناء وميناد eb.toString() لطباعة الاسم والرسالة المرسلة لباني Exception:

```
java.lang.Exception: One of the dimensions of the rectangle is a negative value.
```

Exceptions Handling

4-عند وضع input w =0 سنجد الخطأ التالي:

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
at RecRevExceptionTest.main(RecRevExceptionTest.java:15)
```

The process is not allowed / by zero

5-عند وضع كل من w<l سيتم نداء الاستثناء المعرف من قبل المستخدم LogicalException وارسال رسالة إلى الباني مفادها القيم المدخلة الطول أقل من العرض ويتم تنفيذ catch الخاصة بهذا الاستثناء لإخراج رسالة ومناداة eb.toString() لطباعة الاسم والرسالة المرسلة لباني Exception:

```
Entrance length<0 and width<0 :LogicalException: The entrance length < width.?
```

6- عند وضع `System.out.println( b[-2] );//Array Index Out Of Bounds` سنجد الخطأ التالي:

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: -2
at RecRevExceptionTest.main(RecRevExceptionTest.java:18)
Array Index Out Of Bounds 0 t0 4:java.lang.ArrayIndexOutOfBoundsException: -2
```

7- عند وضع String str="as"; وتنفيذ *parseInt(str);* سنجد الخطأ التالي:

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "as"
at java.lang.Integer.parseInt(Unknown Source)
at RecRevExceptionTest.main(RecRevExceptionTest.java:19)
is not a number.
```

8- عند إعطاء قيم صحيحة **لن يتم قذف أي Exception وسنحصل على نتائج التمرين من ناتج قسمة الطول على العرض، القيمة المخزنه في مكان مسموح ضمن المصفوفة ، ناتج المحيط ، القطر والمساحة كل على سطر منفرد.**

ملاحظة إذا غابت أية catch سيتم تنفيذ catch (Exception e)

The finally Clause

- قد تحتوي جملة try على جملة finally اختيارية optional finally.
- في حالة وجودها ، يجب أن تظهر الجملة finally بعد كل عبارات try catch.

```
{       (try block statements...)       }

catch (ExceptionType ParameterName)
{       (catch block statements...)     }
finally
{       (finally block statements...)   }
```

- الكتلة finally عبارة عن جملة واحده أو أكثر ،
  - يتم تنفيذها دائمًا بعد تنفيذ كتلة try و
  - بعد تنفيذ أي كتل تمثل الالتقاط إذا تم إلقاء استثناء.
- يتم تنفيذ العبارات الموجودة في الكتلة finally سواء حدث استثناء أم لا.

Throwing Exceptions

- يمكنك كتابة كود:

- يطرح أحد استثناءات Java القياسية ،
- أومثيل لصنف استثناءات مخصصة قمت بتصميمها.

- يتم استخدام تعليمة الرمي قصداً أو لرمي استثناء يدويًا.

throw new *ExceptionType(MessageString);*

- يتسبب بيان الرمي في إنشاء كائن الاستثناء وإلقائه.
- يمكن أن تحتوي وسيطة MessageString على رسالة خطأ مخصصة يمكن استردادها من طريقة getMessage() الخاصة بالكائن الاستثنائي.
- إذا لم تمرر رسالة إلى المنشئ ، فسيكون للاستثناء رسالة فارغة.

Input-Output File

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Scanner;

public class WriteAndReadFile {

public static void main(String[] args) {
try{File fw=new File("d:/test.txt");
PrintWriter pr= new PrintWriter(fw);
pr.println("the Third programing langue");
pr.println(" this is the file Writer");
Scanner inf = new Scanner(System.in);
String si = inf.nextLine();
```

end input from keyboard
The file has finished writing and will now be read from.
the Third programing langue
this is the file Writer
end input from keyboard

```
pr.println(si);
pr.close();
System.out.println("\n The file has finished
writing and will now be read from.\n ");

File fr=new File("d:/test.txt");

Scanner inFromFile = new Scanner(fr);
while (inFromFile.hasNext())
{String so=inFromFile.nextLine();
System.out.println(so);}
} // end try block
catch(FileNotFoundException er)
{System.out.println(er);}
}}
```

مسار الملف يمكن (“d:\test.txt or d:// test.txt”)
يفتح الملف من أجل الكتابة ضمنه وتم ادخال شريطين عبر println
وأخرى عبر اوحة المفاتيح ثم تم إغلاقة وطباعة عبارة بإنتهاء الادحال
تم فتحة وقراءة مافي داخلة وكتابته على وحدة الخرج القياسية

Using an ArrayList

- على غرار المصفوفة المعرفة من قبل المستحدم، تسمح ArrayList بتخزين الكائنات.
- على عكس المصفوفة ، فإن كائن ArrayList:
  - يتم توسيعها تلقائيًا عند إضافة عنصر جديد.
  - تتقلص تلقائيًا عند إزالة العناصر.

- **Requires:**

```
import java.util.ArrayList;
ArrayList <String> nameList = new ArrayList <String> ();
```

Notice the word String **written inside angled brackets <>**

This specifies that **the** ArrayList **can hold String objects.**

If we try to store any other type of object in this ArrayList**, an error will occur.**

Using an ArrayList

- **To populate the** ArrayList**, use the add method:**
  - nameList.add("James");
  - nameList.add("Catherine");
- **To get the current** size**, call the size method**
  - nameList.size(); // returns 2
- **To access items in an** ArrayList**, use the** get **method**

  nameList.get(1);
- **The** ArrayList **class's** toString **method returns a string representing all items in the** ArrayList

  System.out.println(nameList);

  This statement yields : [James, Catherine]

  The ArrayList **class's remove method** removes **designated item from the** ArrayList

  nameList.remove(1);

  This statement removes the second item.

Using an ArrayList

- **The** ArrayList **class's add method with one argument adds new items to the end of the** ArrayList **.**
- **To insert items at a location of choice, use the** add **method with two arguments:**

  nameList.add(1, "Mary");

  This statement inserts the String **"Mary" at index 1**
- **To replace an existing item, use the set method:** nameList.set(1, "Becky");

  This statement replaces "Mary" with "Becky"
- **An** ArrayList **has a capacity, which is the number of items it can hold without increasing its size.**
- **The default capacity of an** ArrayList **is 10 items.**
- **To designate a different capacity, use a parameterized constructor:**

  ArrayList list = new ArrayList(100);

Create an ArrayList

Methods of ArrayList

Constructor	Description
ArrayList()	It is used to build an empty array list (constructer).
ArrayList(Collection<? extends E> c)	It is used to build an array list that is initialized with the elements of the collection c.
ArrayList(int capacity)	It is used to build an array list that has the specified initial capacity.

Constructors of ArrayList

Method	Description
void add(int index, E element)	It is used to insert the specified element at the specified position in a list.
boolean add(E e)	It is used to append the specified element at the end of a list.
boolean addAll(Collection<? extends E> c)	It is used to append all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator.
boolean addAll(int index, Collection<? extends E> c)	It is used to append all the elements in the specified collection, starting at the specified position of the list.

Create an ArrayList

Constructors of ArrayList

Method	Description
void clear()	It is used to remove all of the elements from this list.
void ensureCapacity(int requiredCapacity)	It is used to enhance the capacity of an ArrayList instance.
E get(int index)	It is used to fetch the element from the particular position of the list.
boolean isEmpty()	It returns true if the list is empty, otherwise false.
Iterator()	
listIterator()	
int lastIndexOf(Object o)	It is used to return the index in this list of the last occurrence of the specified element, or -1 if the list does not contain this element.
<T> T[] toArray(T[] a)	It is used to return an array containing all of the elements in this list in the correct order.
Object clone()	It is used to return a shallow copy of an ArrayList.
boolean contains(Object o)	It returns true if the list contains the specified element.
int indexOf(Object o)	It is used to return the index in this list of the first occurrence of the specified element, or -1 if the List does not contain this element.

Create an ArrayList

Constructors of ArrayList

Method	Description
E remove(int index)	It is used to remove the element present at the specified position in the list.
boolean remove(Object o)	It is used to remove the first occurrence of the specified element.
boolean removeAll(Collection<?> c)	It is used to remove all the elements from the list.
boolean removeIf(Predicate<? super E> filter)	It is used to remove all the elements from the list that satisfies the given predicate.
protected void removeRange(int fromIndex, int toIndex)	It is used to remove all the elements lies within the given range.
void replaceAll(UnaryOperator<E> operator)	It is used to replace all the elements from the list with the specified element.
void retainAll(Collection<?> c)	It is used to retain all the elements in the list that are present in the specified collection.
E set(int index, E element)	It is used to replace the specified element in the list, present at the specified position.
void sort(Comparator<? super E> c)	It is used to sort the elements of the list on the basis of the specified comparator.
Spliterator<E> spliterator()	It is used to create a spliterator over the elements in a list.
List<E> subList(int fromIndex, int toIndex)	It is used to fetch all the elements that lies within the given range.
int size()	It is used to return the number of elements present in the list.
void trimToSize()	It is used to trim the capacity of this ArrayList instance to be the list's current size.

جَامعة
المَنارة
MANARA UNIVERSITY

انتهت محاضرة الأسبوع السادس

Exception Classes

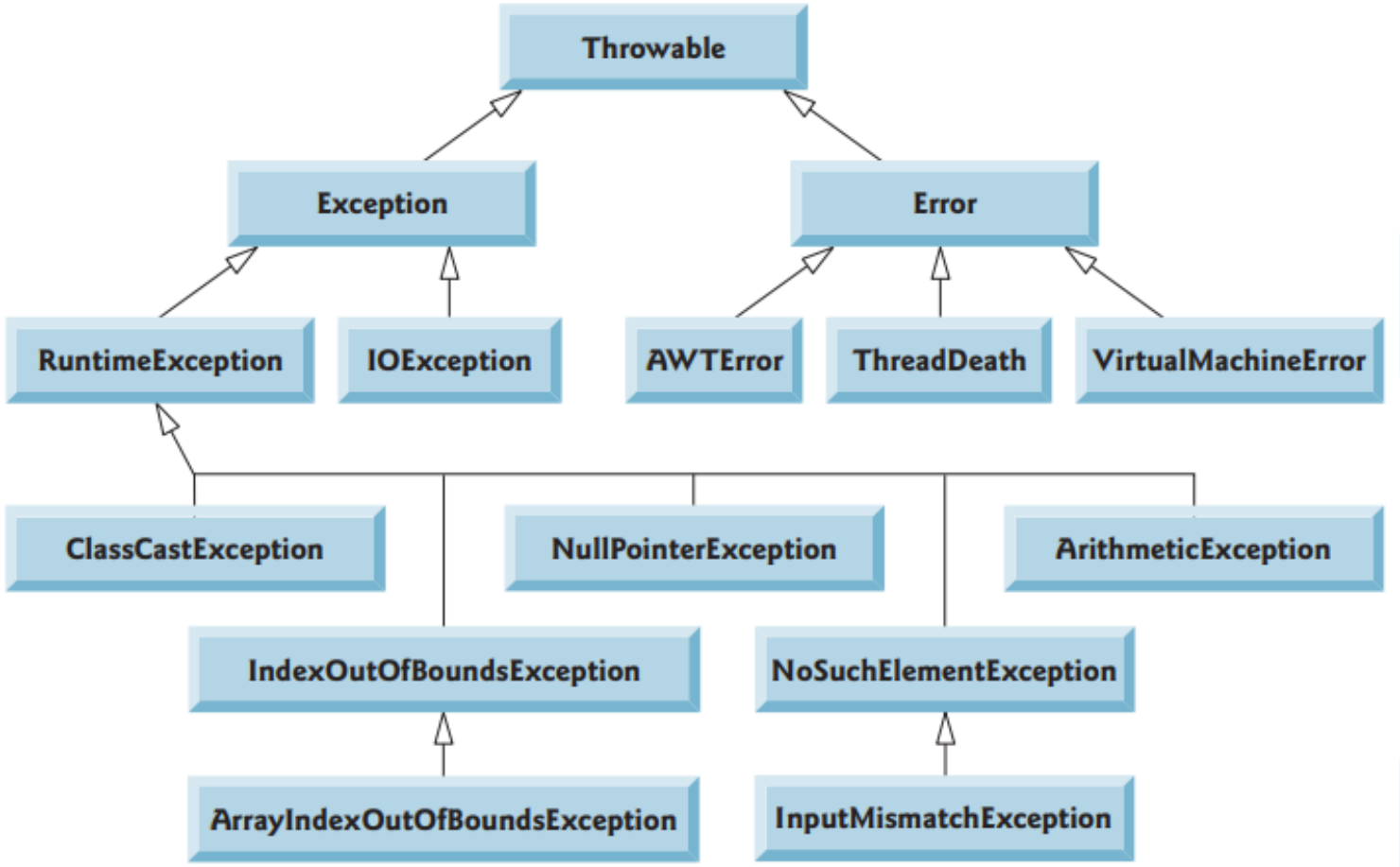


Fig. 11.4 | Portion of class `Throwable`'s inheritance hierarchy.

Using an ArrayList

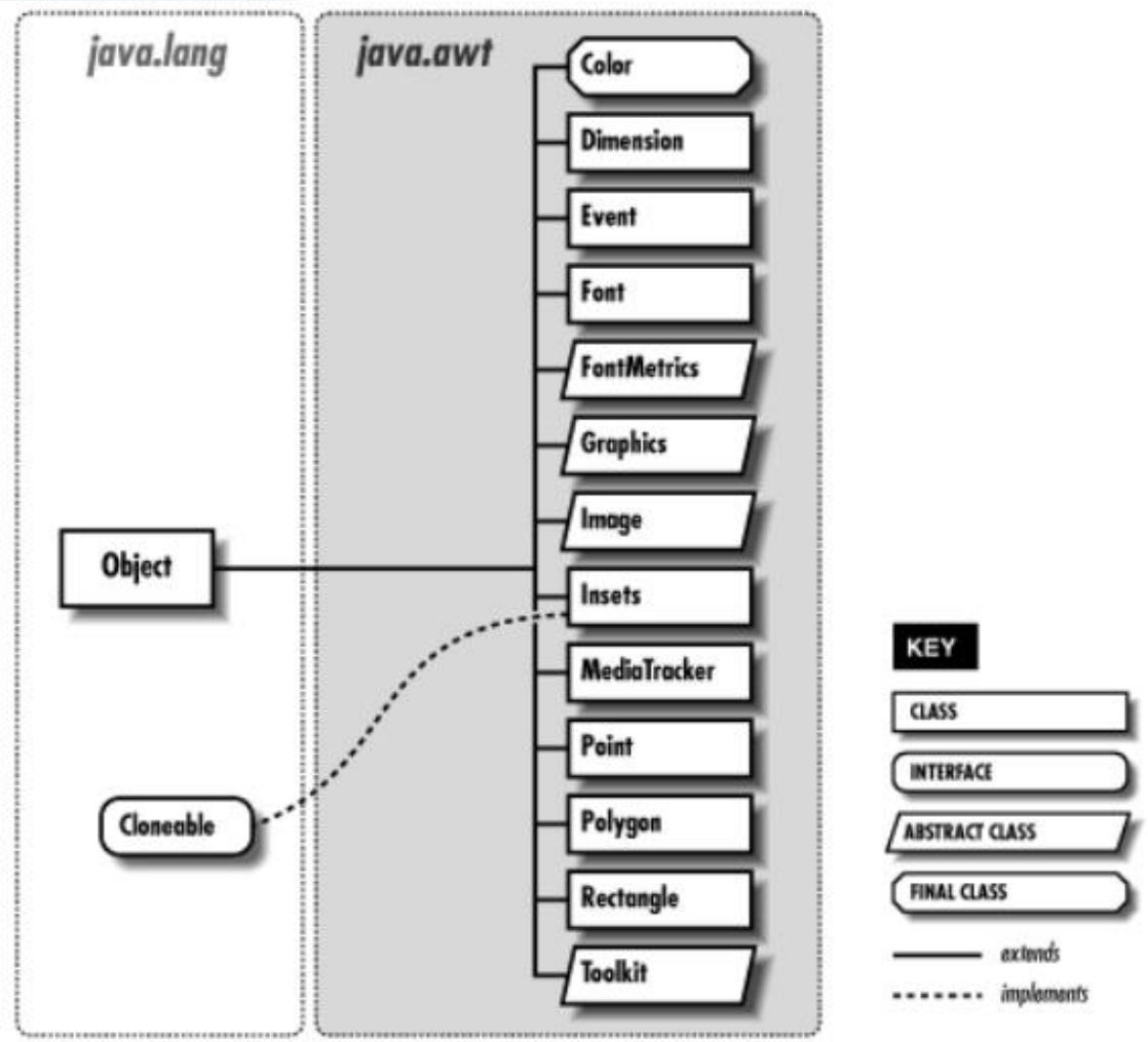
- **You can store any type of object in an ArrayList**

ArrayList<BankAcount>accountList=new ArrayList <BankAcount> ();

This creates an ArrayList **that can hold** BankAccount **objects.**

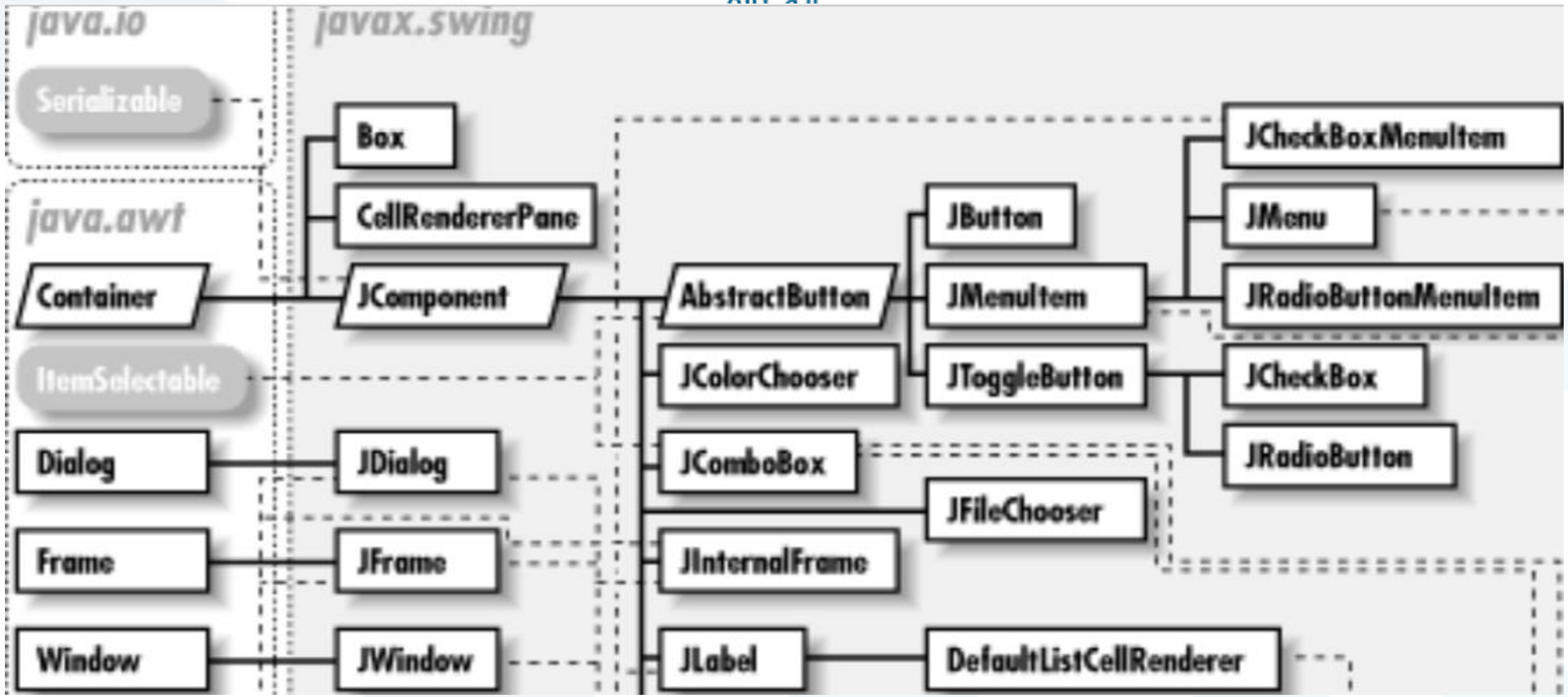
Create an ArrayList

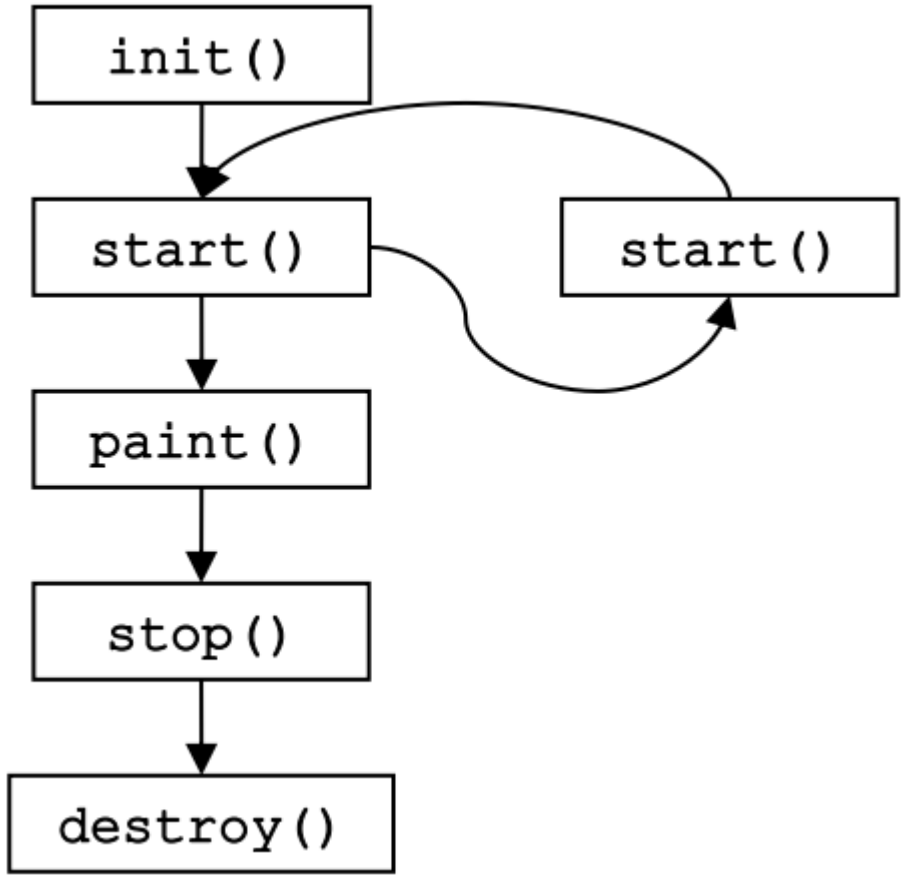
```
// Create an ArrayList to hold BankAccount objects.

 ArrayList list = new ArrayList();

 // Add three BankAccount objects to the ArrayList.

 list.add(new BankAccount(100.0));
list.add(new BankAccount(500.0));
list.add(new BankAccount(1500.0));

  // Display each item.

 for (int index = 0; index < list.size(); index++)
{ BankAccount account = list.get(index);
System.out.println("Account at index " + index +
 "\nBalance: " + account.getBalance());
}
```



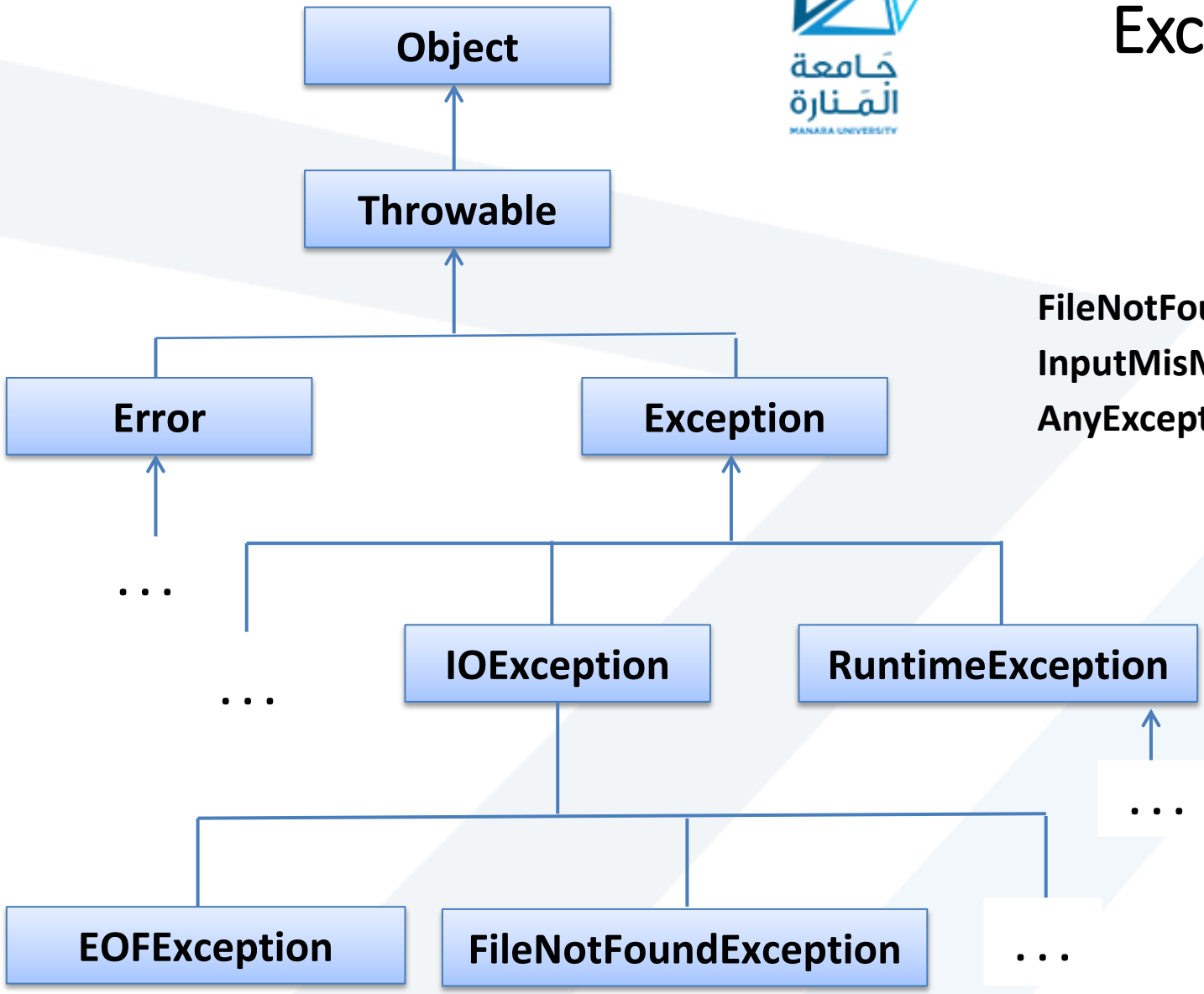
bookes\dataStru\java2021-2022\scrap

Noor-Book.com
مدخل أساسي إلى البرمجة
كائنية التوجه أساسيات
البرمجة بلغة جافا
Pag 96 Java


java.io
javax.swing
Serializable
Box
CellRendererPane
java.awt
Container
JComponent
AbstractButton
JButton
JMenuItem
JRadioButtonMenuItem
ItemSelectable
JColorChooser
JToggleButton
JCheckBox
Dialog
JDialog
JComboBox
JRadioButton
JFileChooser
Frame
JFrame
JInternalFrame
Window
JWindow
JLabel
DefaultListCellRenderer


init()
start()
start()
paint()
stop()
destroy()

Exception Classes



FileNotFoundException is Exception

InputMisMatchException is Exception

AnyException is Exception