جَامعة المَنارة
MANARA UNIVERSITY

كلية الهندسة المعلوماتية

برمجة 3

Java Programming

ا. د. علي عمران سليمان

محاضرات الأسبوع التاسع

الفصل الثاني 2023-2024

content

- Exceptions Handling .
- Exception Classes
- Handling Multiple Exceptions.
- The finally Clause.
- Throwing Exceptions .
- 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 تحويله لها.

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

- تقسم الإستثناءات في جافا إلى عدة أنواع وكل نوع تم تمثيله في صنف منعزل. جميع هذه الأصناف ترث من صنف أساسي إسمه Exception
- الصنف Exception بدوره يرث من الصنف Throwable، أي صنف يرث من الصنف Exception هو صنف يمثل إستثناء معين.
- الأخطاء التي سببها الأجهزة والتي تسمى Errors, تم إنشاء صنف خاص لهم إسمه Error وهو يرث مباشرةً من الصنف Throwable.
- 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 2

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

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

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

Exceptions Handling 3

- تبدأ جملة catch بالكلمة المفتاحية catch:

- catch (ExceptionType ParameterName)

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

- يُعرف كود الكتلة التي تلي catch ببلوك catch أو المعالجة.
- يتم تنفيذ الكود الموجود في كتلة catch إذا توافق مع الاستثناء المقذوف 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.");          }
```

- يبحث Java Virtual Machine عن عبارة catch يمكن أن تتعامل مع الاستثناء.

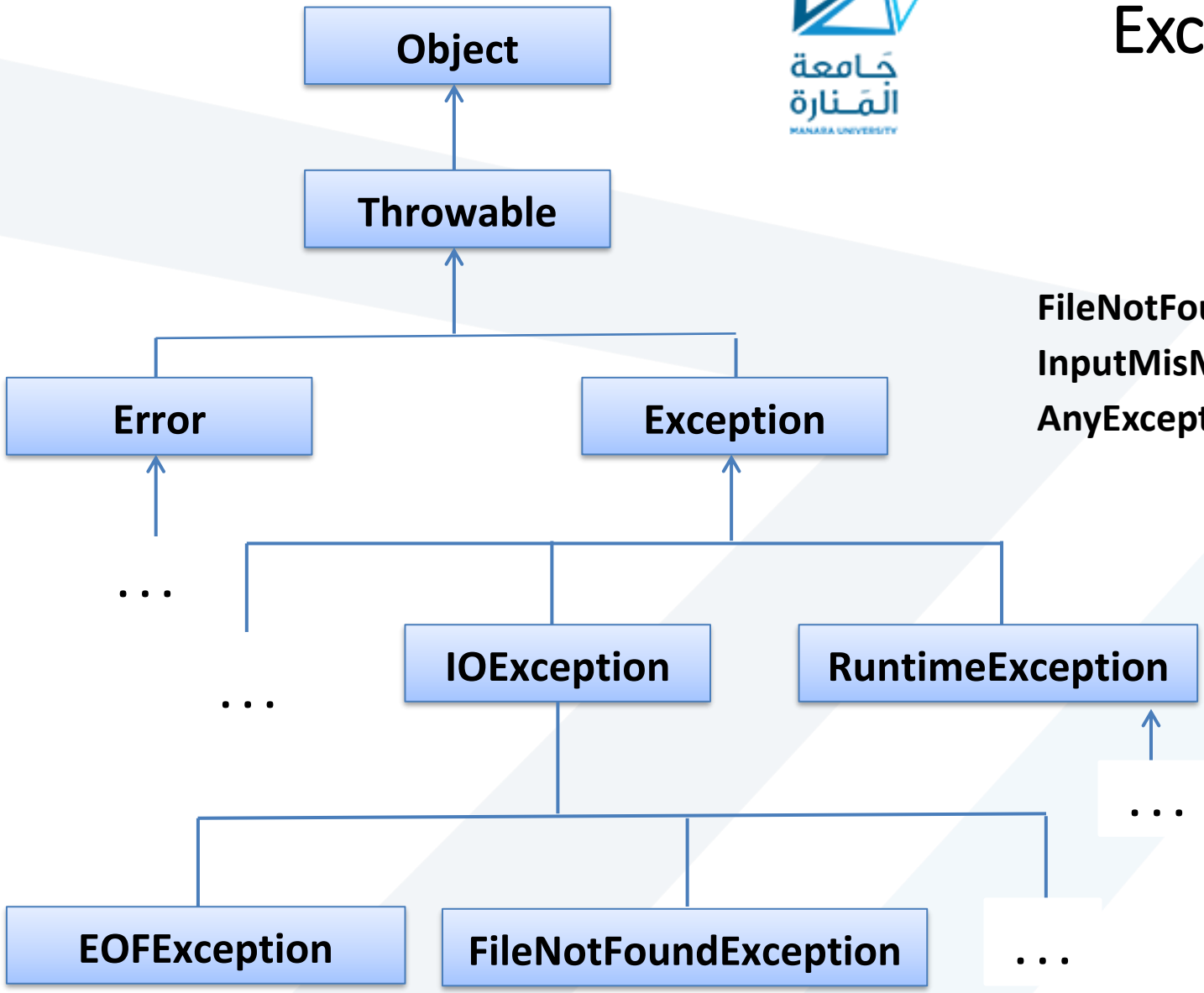
Polymorphic References To Exceptions

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

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

- تطرح الطريقة parseInt() من الصنف Integer كائن الاستثناء NumberFormatException.
- يتم اشتقاق صنف NumberFormatException من الصنف Exception.

Exception Classes



FileNotFoundException is Exception

InputMisMatchException is Exception

AnyException is Exception

Exception Classes

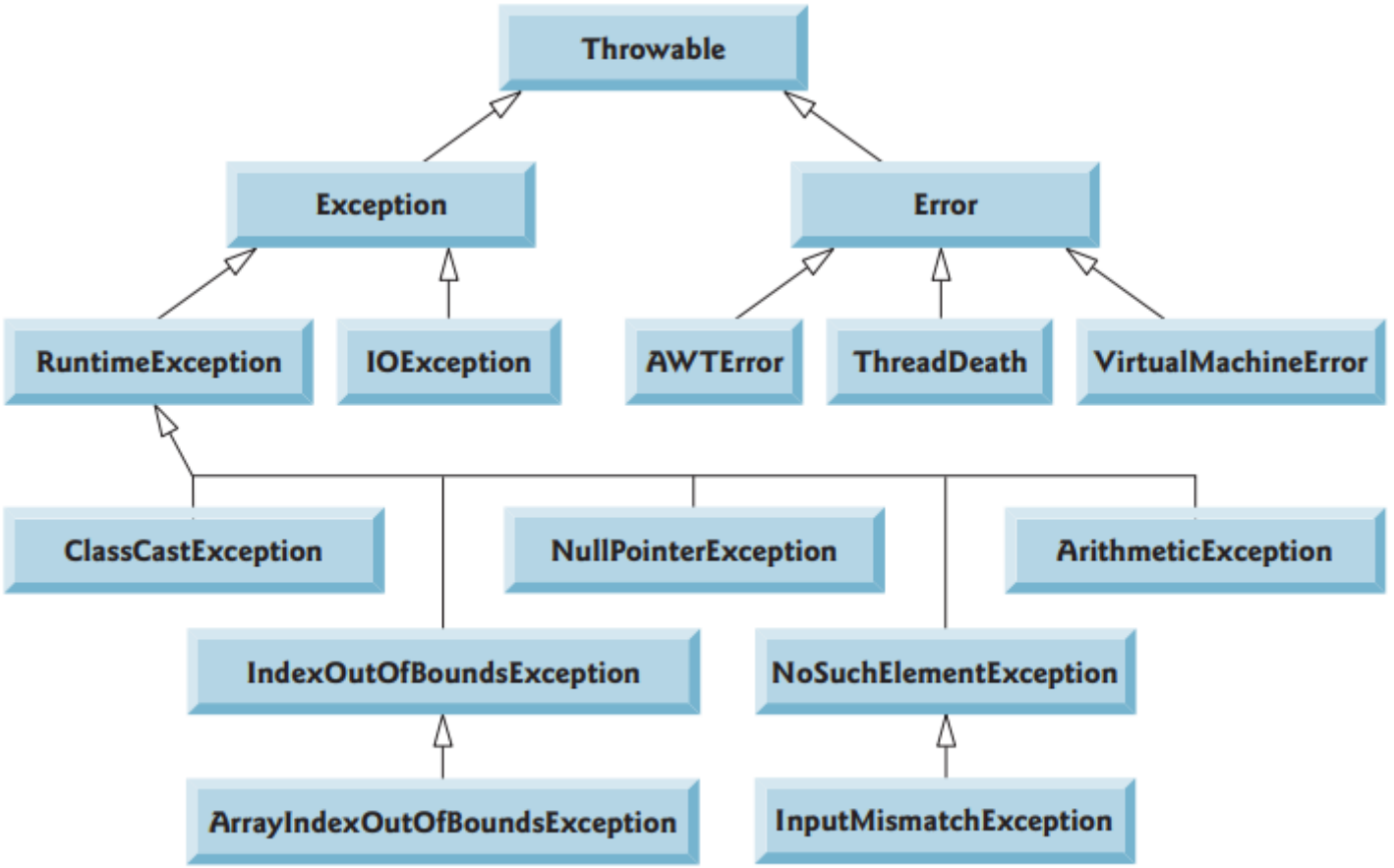


Fig. 11.4 | Portion of class `Throwable`'s inheritance hierarchy.

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

```
public class RestException
{
private  int  length,width;
private  static int noOOb=0;
public RestException(){length=0;width=0;noOOb++;}

public  int getnoOOb() {return noOOb;}

public static int add(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; }
}
```

Exceptions Handling

```
import java.util.Scanner;
import java.util.InputMismatchException;
public class RestExceptionTest{
public static void main(String[] args) {
RestException Rect= new RestException();
Scanner input=new Scanner(System.in);
int l,w,a;String str="as";
try {int[] b = new int [5];
System.out.println("input L ");l=input.nextInt();
System.out.println("input w ");w=input.nextInt();
System.out.println("the l div w =  "+ l/w);
if (l < w)
{throw new IllegalArgumentException( "In Rectangle , The length is at least
equal to the 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 value add ob1= "+Rect.add(l, w));
```

Exceptions Handling

```
System.out.println("\nthe value sub ob1=  "+Rect.dia(l, w));
System.out.println("\nthe value mult ob1= "+Rect.area(10, 44));
System.out.println("\nnumber of object= "+Rect.getnoOOb());
}
catch(NumberFormatException e)
{System.out.println( " is not a number."); }

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

catch(InputMismatchException e1)
{System.out.println("Bad number InputMismatch."); }

catch (Exception e)
{System.out.println("l<w not Rectangle: "+e.getMessage());
System.out.println("L<W ExeceptionName: "+e.toString()); }}
} // e.toString() return ExeceptionName and Message over loading
```

Exceptions Handling

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

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: -2
at RestExceptionTest.main(RestExceptionTest.java:18)
```

2- عند وضع String str="as"; سنجد الخطأ التالي:

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "as"
at java.lang.Integer.parseInt(Unknown Source)
at RestExceptionTest.main(RestExceptionTest.java:19)
```

3- عند وضع `input w =0` سنجد الخطأ التالي:

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

4- عند وضع `input w =m` سنجد الخطأ التالي:

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

5- عند إعطاء `w=44, l=22` سيتم تنفيذ `catch (Exception e)` سنحصل على:

```
l<w not Rectangle: In Rectangle ,The length is at least equal to the width.
L<W ExeceptionName: java.lang.IllegalArgumentException: In Rectangle ,The length
is at least equal to the width.
```

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 التقاط 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 القذف او لطرح استثناء يدويًا.

throw new *ExceptionType(MessageString);*

- يتسبب بيان الرمي في إنشاء كائن الاستثناء وإلقائه.
- 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.

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);

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

Using an ArrayList

This statement removes the second item.

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);

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());
}
```

Create an ArrayList

Methods of ArrayList

Constructor	Description
ArrayList()	It is used to build an empty array list.
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

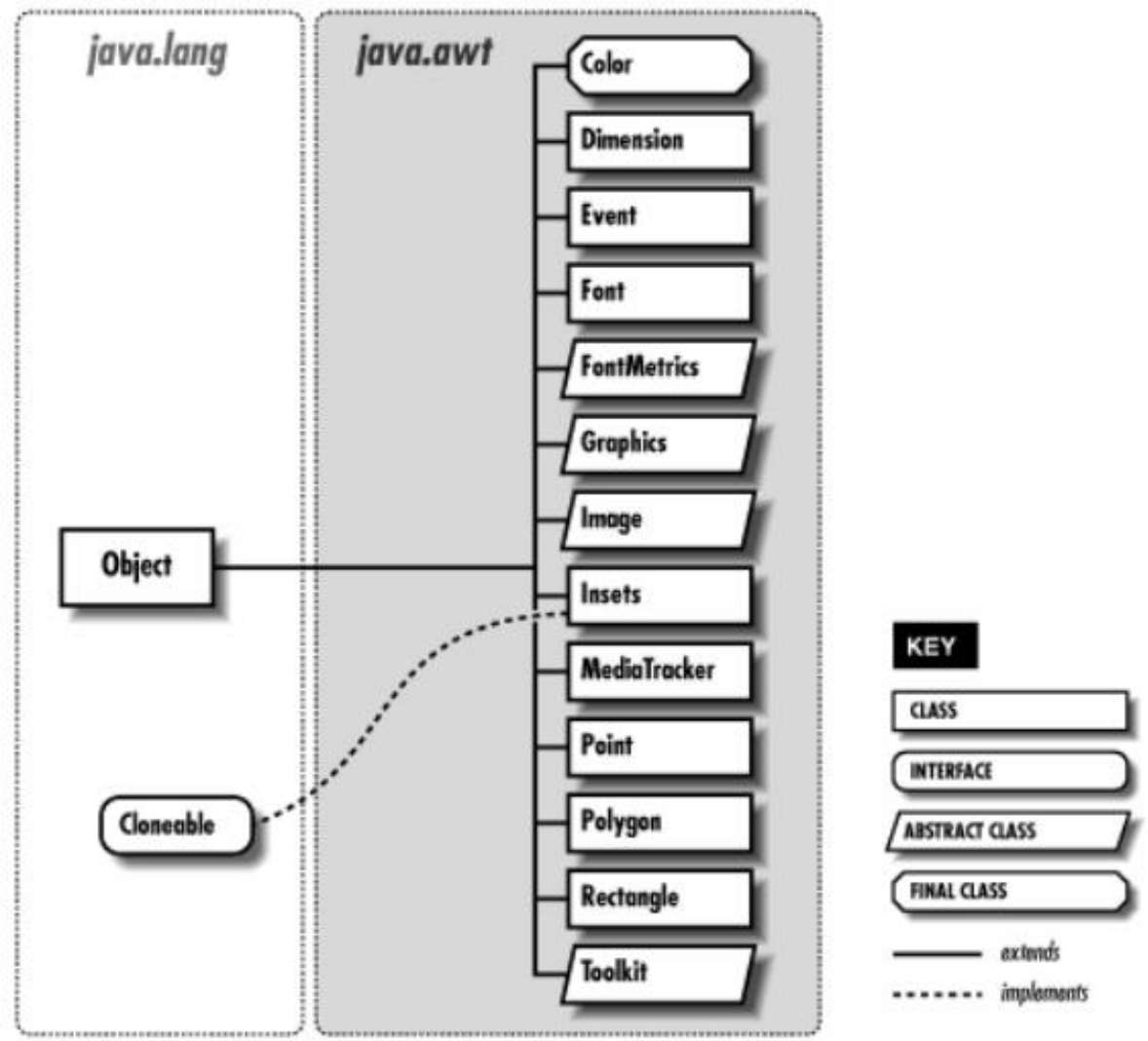
Method	Description
void clear()	It is used to remove all of the elements from this list.
void ensureCapacity(int requiredCapacity)	It is used to enhance the capacity of an ArrayList instance.
E get(int index)	It is used to fetch the element from the particular position of the list.
boolean isEmpty()	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.
Object[] toArray()	It is used to return an array containing all of the elements in this list in the correct order.
<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

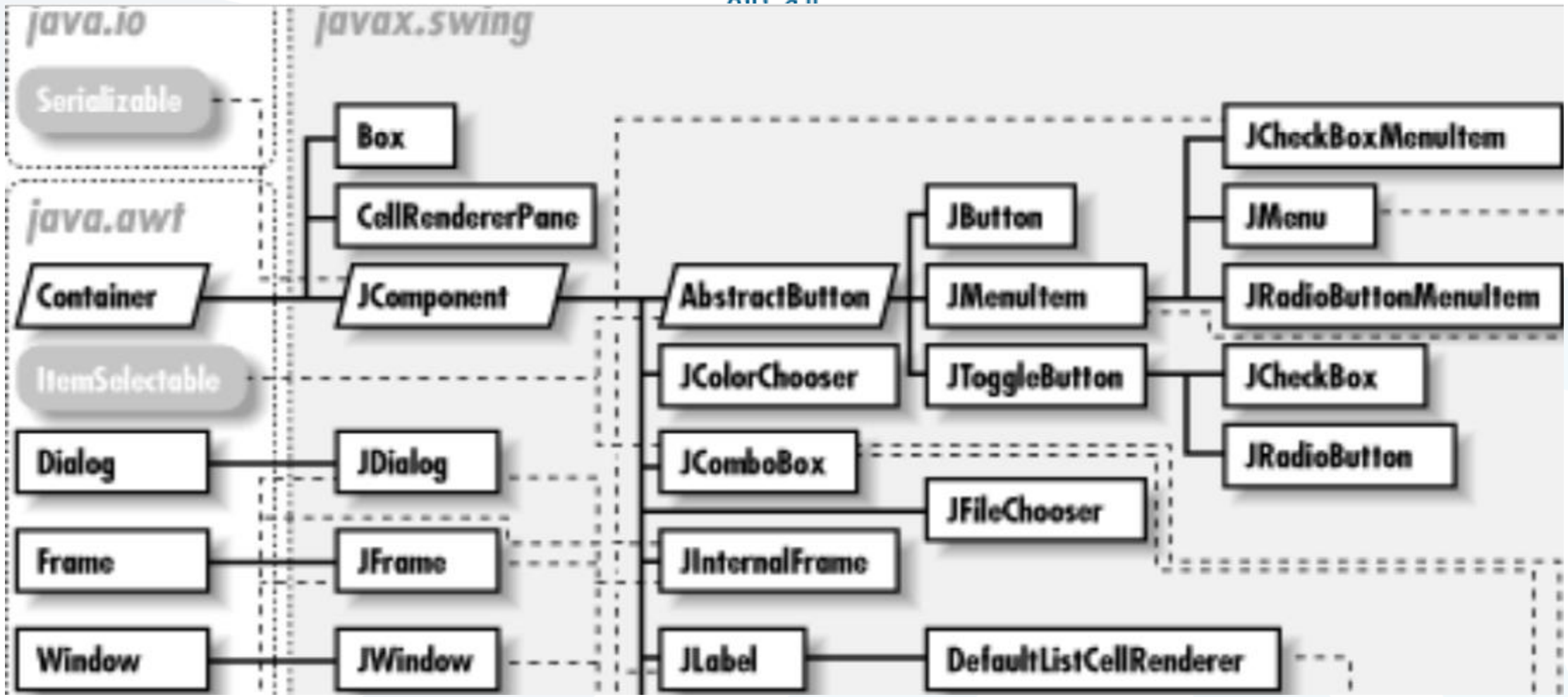
Method	Description
E remove(int index)	It is used to remove the element present at the specified position in the list.
boolean remove(Object o)	It is used to remove the first occurrence of the specified element.
boolean removeAll(Collection<?> c)	It is used to remove all the elements from the list.
boolean 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.

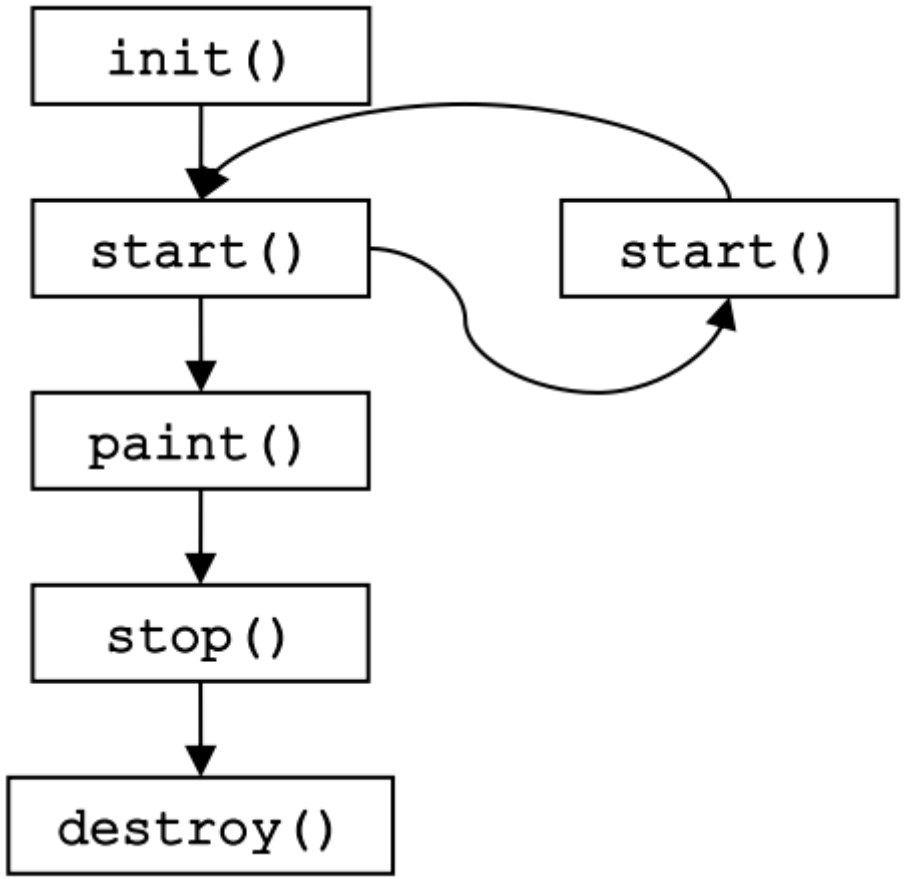
انتهت محاضرة الأسبوع التاسع



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
JCheckBoxMenuItem
JMenu
JRadioButtonMenuItem
JColorChooser
JToggleButton
JCheckBox
JRadioButton
ItemSelectable
Dialog
JDialog
JComboBox
JFileChooser
Frame
JFrame
JInternalFrame
Window
JWindow
JLabel
DefaultListCellRenderer


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