قسم الهندسة المعلوماتية

برمجة 3

Java Programming

ا. د. علي عمران سليمان

محاضرات الأسبوع الثالث

الفصل الثاني 2023-2024

Contents 1

1. Objects and Classes .	11. Inheritance and Constructors.
2. UML class diagrams .	12. Overriding Superclass Methods.
3. Performing output Displaying with print, println, printf.	3.6 Class JOptionPane Using Dialog Boxes showMessageDialog(), showInputDialog()
4. Performing Input Scanner and some of its methods.	4.15 GUI &Graphics,
5. default constructor.	4.15 Creating Simple Drawings—Displaying and drawing lines on the screen
6. Overloaded Constructors, and methods.	5.11 Drawing Rectangles and Ovals—Using shapes to represent data.
7. Static Method , and Data fields.	
8. Call by value and references.	
9. copy constructor.	
10. inherited class.	

References

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

Using The == operators with objects

- **If we try the following:**

```
Rectangle r1 = new Rectangle(60,50);
Rectangle r2 = new Rectangle(60,50);
if (r1 == r2) // This is a mistake
        System.out.println("The objects are the same.");
Else
        System.out.println("The objects are not the same.");
```

سيتم طباعة :

The objects are not the same.

only the addresses of the objects are compared.

Methods That Copy Objects

- هناك طريقتان لنسخ كائن.

- لا يمكنك استخدام عامل النسب لنسخ محتوى المراجع (محتوى الكائن) بشكل مباشر بل يتم من خلال.

- نسخة مرجعية فقط: 1- هذا ببساطة هو نسخ عنوان كائن إلى متغير مرجعي لكائن آخر.

- نسخة عميقة Deep copy 2- يتضمن ذلك إنشاء مثيل جديد للفئة ونسخ القيم من كائن إلى كائن أخر.

الحالة الاولى :

```
Rectangle r1 = new Rectangle(60,50);
Rectangle r2 = new Rectangle(60,50);
r2=r1;
if (r1 == r2)     System.out.println("The objects are the same.");
Else              System.out.println("The objects are not the same.");
```

سيتم طباعة :

The objects are the same.

لو تم التعديل على إحداهما سينطبق على الاخر نظراً لإن الاثنان يؤشران لنفس المكان ضمن الذاكرة:

```
r2.setLength(66);        System.out.println(r1.getLength());
```

Methods That Copy Objects

الحالة الثانية:

- **A copy constructor accepts an existing object of the same class and clones it**

Rectangle r1 = new Rectangle(60,50);

```
public Rectangle(Rectangle r2)
    { length = r2.length; width = r2. width ; }   // end Create  copy constructor
```

Rectangle r2 = new Rectangle(r1);

OR

إذا وجد باني ببارمترين يمكن أن نهيئ كائن من خلال تمرير القيمتين.

```
Rectangle  r1= new Rectangle(13, 9);          // Create r3, a copy of r1
Rectangle  r3 = Rectangle (r1);
```

What is Inheritance?

- الكائنات الو اقعية هي عادةً نسخ متخصصة من كائنات أخرى أكثر عمومية.

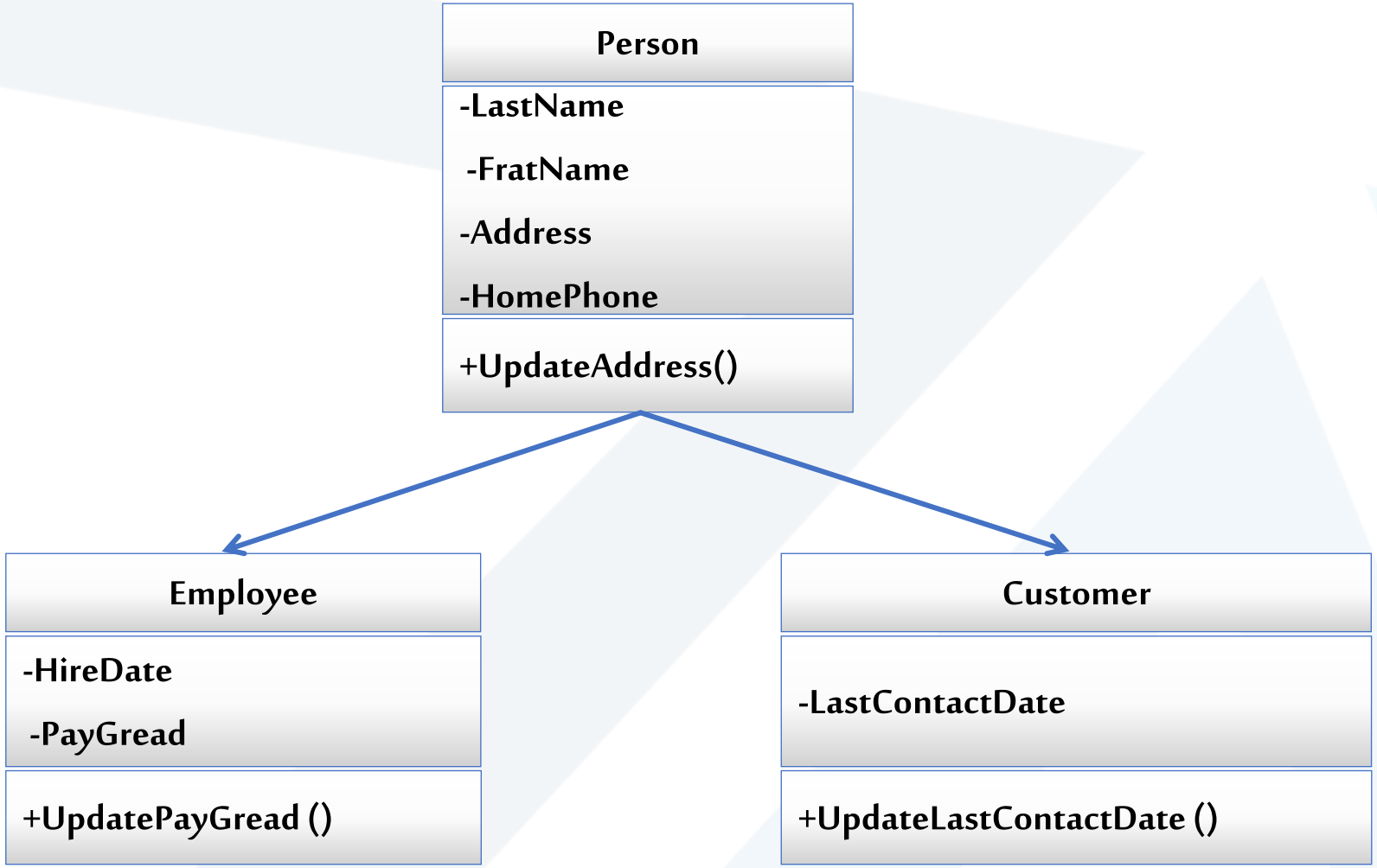
- يصف مصطلح "الطالب" نوعًا عامًا جدًا من الطلاب ذوي الخصائص المعروفة.

- طلاب الدراسات العليا والطلاب الجامعيين هم نسخ متخصصة من الطلاب.

- يتشاركون في الخصائص العامة للطالب.
- ومع ذلك ، لديهم خصائص خاصة بهم.
- بعد التخرج مجال بحث مثير للاهتمام.
- قبل الخريجين لديهم رقم مجموعة ورقم صف.

What is Inheritance?

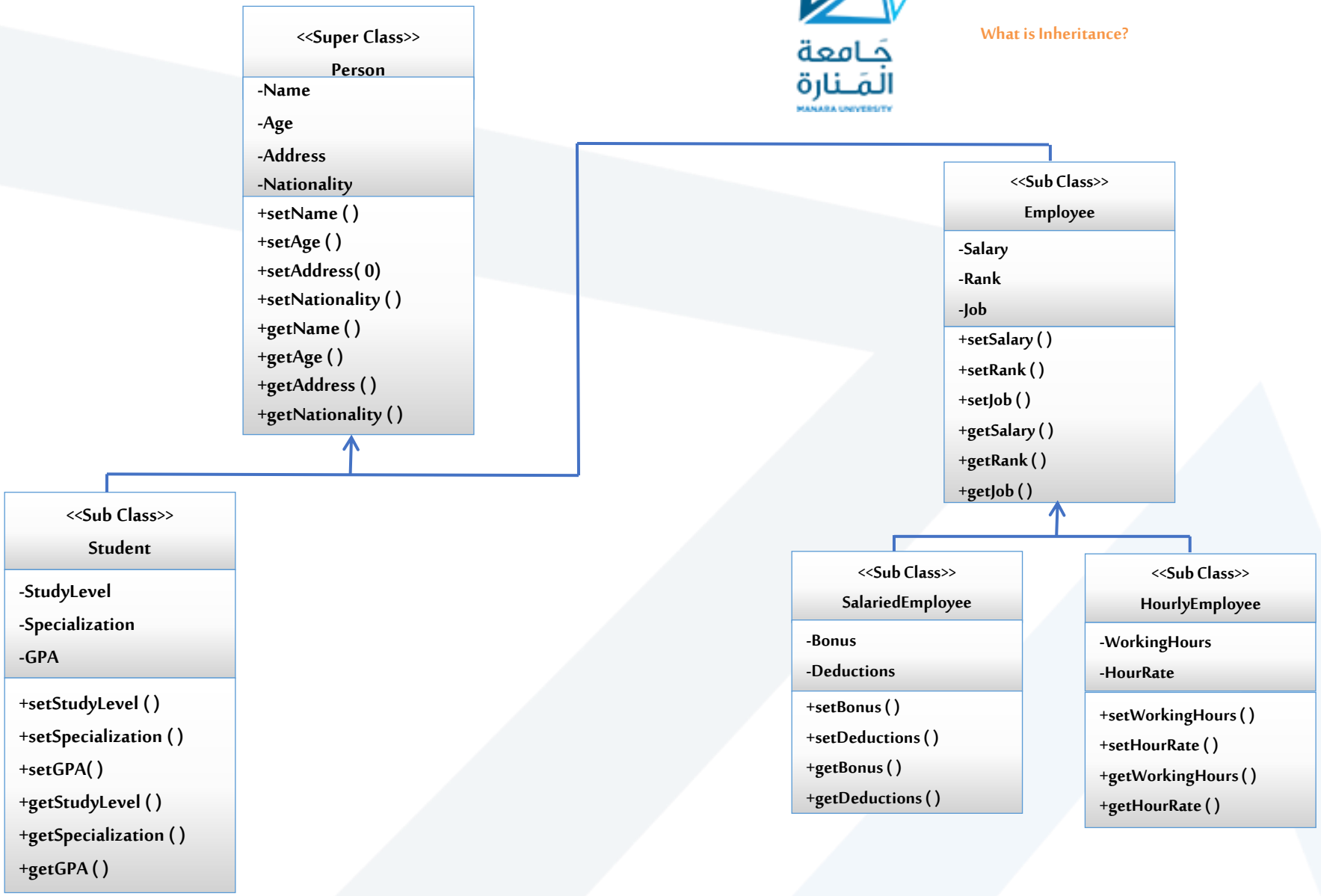
With Inheritance



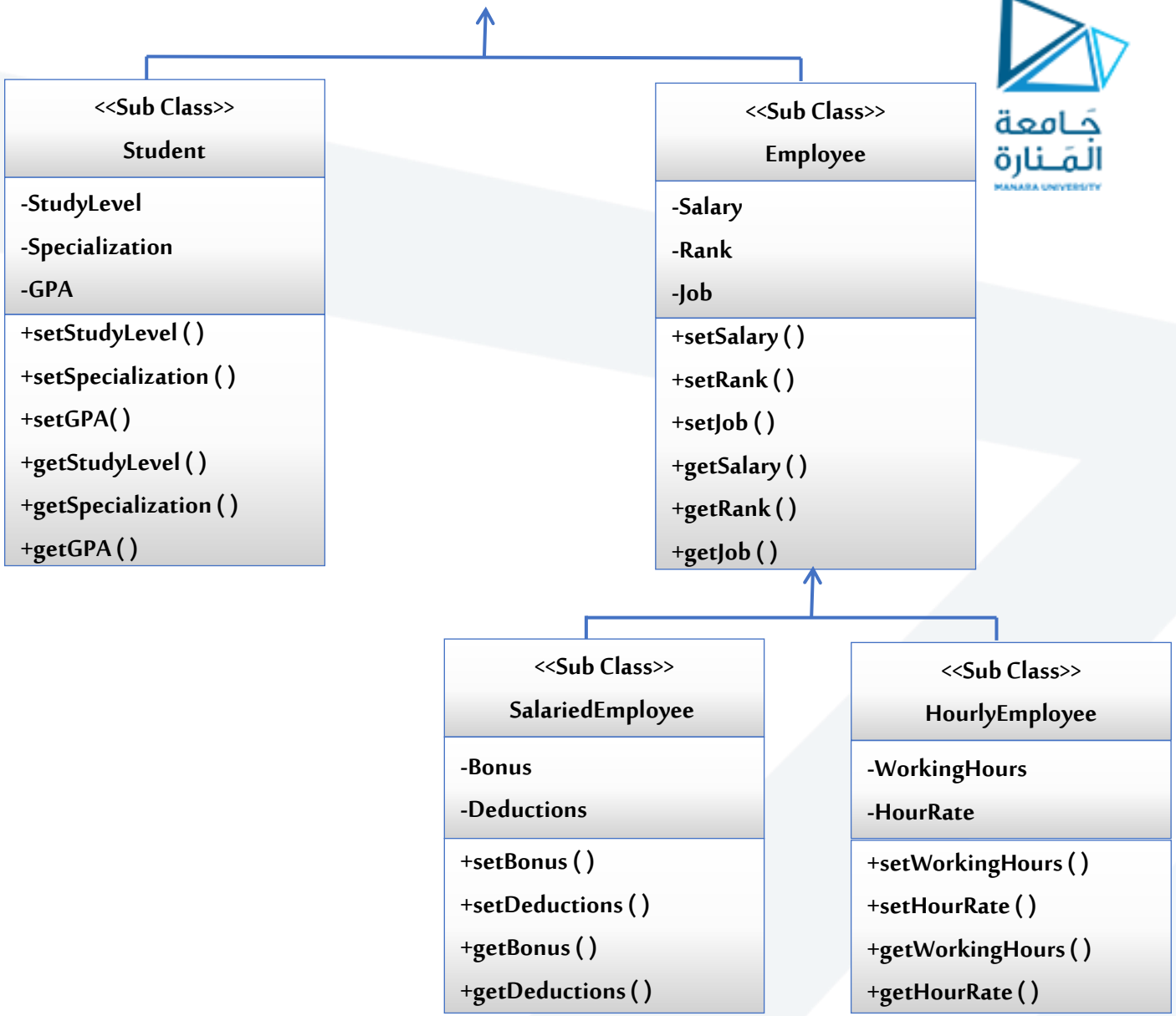
What is Inheritance?

Generalization vs. Specialization

What is Inheritance?



What is Inheritance?



The “is a” Relationship

- العلاقة بين الطبقة العليا superclass والطبقة الموروثة inherited class تسمى علاقة " is a ".
  - طالب الدراسات العليا " is a " طالب Student.
  - الموظف " is a " شخص Person.
  - الموظف بأجر" is a " موظف Employee.
  - السيارة " is a " مركبة vehicle.

- الكائن المتخصص له:
  - جميع خصائص الكائن العام وخصائص إضافية تجعلها مميزة (تميزها).

- في البرمجة OOP، يتم استخدام الوراثة لإنشاء علاقة " is a " بين الفئات وما تعرف بالوراثة.

The “is a” Relationship

- يمكننا توسيع extend قدرات الصنف.
- **Inheritance** يشمل الطبقة العليا **superclass** والطبقة الفرعية **subclass**.
  - صنف الأب هو الصنف العام أو المعمم **general**.
  - الصنف الفرعي هي الصنف المخصص **specialized**.

- **The subclass is based on, or extended from, the superclass.**

  - تسمى طبقات الأباء Superclasses أيضًا الطبقات الأساسية **classes** base ،
  - تسمى الطبقات الفرعية subclasses أيضًا الطبقات المشتقة **classes** derived.

- يمكن اعتبار العلاقة بين الأصناف بمثابة صنف للوالدين واصناف فرعية **as parent classes and child classes**.

Inheritance

- يرث الصنف الفرعية subclass (الوارث) الحقول والأساليب من superclass (المورث) دون إعادة كتابة أي منها.
- يمكن إضافة حقول وأساليب جديدة إلى الصنف الفرعي subclass (الوارث).
- يتم استخدام الكلمة المفتاحية extends في Java، في سطررأس الصنف مابعد الصنف الوارث وقبل الصنف المورث.

public class Employee extends **Person**

- كما هو معروف: - نستفيد في مبدأ إعادة الاستخدام بدون كتابة المشترك بين الأصناف مرتين.
  - عند إضافة حقل معطيات أو طريقة في صنف الأب مرة واحدة ستظهر عند كل الورثة.
  - يمكن التعديل على الطريقة التي نحتاج التعديل عليها ماعدا الطرق final.
- أعضاء الصنف الفائق superclass التي تم تعريفها على أنها خاصة: - غير موروثة من قبل الصنف الفرعي subclass، - موجودة في الذاكرة عند إنشاء كائن من الصنف الفرعي subclass ولا يمكن الوصول إلى الأعضاء الخاصة للصنف المورث من قبل الصنف الفرعي subclass إلا بالطرق العامة public methods للصنف الأعلى superclass.
- أعضاء الصنف superclass المعرفين public: - موروثة من قبل الصنف الفرعي.- يمكن الوصول إليها مباشرة من الصنف الفرعي subclass.

Inheritance, Fields and Methods

- When an instance of the subclass is created, the non-private methods of the superclass are available through the subclass object.

```
Employee emp1 = new Employee();
 emp1.set_Age(30);
System.out.println("Age = " + emp1.get_Age());
```

- Non-private methods and fields of the superclass are available in the subclass.

```
Set_Age(30);
```

- المنهج الباني لايورث أي لايمكن تعديل باني الصنف الأساس من النف المشتق، بل يمكن استدعائه.
- عندما يتم إنشاء مثيل لصنف فرعي ، يتم تنفيذ الباني الافتراضي لصنف الأب superclass أولاً.

يمكن استدعاء باني الصنف الأب بشكل صريح من الصنف الأبن باستخدام الكلمة المحجوزة super.

- إذا تم تعريف الباني مع البارامترات في صنف الأب .
  - يجب أن يوفر صنف الأب باني بدون بارامترات no-arg. أو
  - يجب أن يوفر الصنف المشتق باني ويجب أن ينادي باني الأب.
- يجب أن تكون الاستدعاءات لباني الأب هي أول عبارة جافا في باني الابن.

Overriding Superclass Methods

- قد يكون للصنف الفرعي طريقة لها نفس التوقيع مثل طريقة الصنف الأب.
- طريقة الصنف الفرعي تقوم بإعادة كتابة overrides طريقة الصنف الأب نظراً لاختلاف في التنفيذ.
- يُعرف هذا باسم أسلوب بإعادة كتابة overrides وبالتالي ستغطي الطريقة السابقة.
- يجب أن يكون لطريقة الصنف الفرعي التي overrides ولطريقة الصنف الأب التوقيع نفسه.

- لدينا طريقة اسمها getSalary() {return salary;} موجودة ضمن الصنف Employee.
- كذلك ضمن صنف SataredEmployee هي نفسها getSalary() {return salary+bonus-deductions ;} .
- وموجودة ضمن الصنف HourlyEmployee وبطريقة حساب مختلف return working_hours*hours_rate;.

- يستدعي كائن من الصنف الفرعي نسخة الطريقه للصنف الفرعي الخاصة به، وليس طريقة صنف الأب.
- يحبب استخدام التعليق التوضيحي @Override قبل إعلان طريقة الصنف الفرعي مباشرةً.

Calling The Superclass Constructor

Overriding Superclass Methods

Employee

```
public double get_salary( )
{
return salary;
}
```

Salaried Employee

```
public double get_salary( )
  {
    return salary+bonus-deductions ;
  }
```

Hourly Employee

```
public double get_salary( )
{   return working_hours*hours_rate;
  }
```

Overriding Superclass Methods

- عند إجراء overridden لطريقة من الصنف الأب ضمن الصنف الابن، يمكن استخدام الكلمة المفتاحية super من اجل نداء طريقة الصنف الأب من الصنف الأبن ، لمناداة الطريقة ()getSalary الخاصة بالصنف Employee من الصنف SataredEmployee نكتب ضمن الصنف SataredEmployee .

```
super. getSalary();
```

- هناك تمييز بين التحميل الزائد للطريقة و overridden.
- التحميل الزائد كما هو معروف عندما يكون للطريقة نفس اسم طريقة أخرى أو أكثر، ولكن بتوقيع مختلف.
- يمكن أن يحدث كل من التحميل الزائد و overriding في علاقة الوراثة inheritance.
- يمكن أن يحدث overriding فقط في علاقة الوراثة.

Preventing a Method from Being Overridden

- معدّل الوصول final سيمنع overriding طريقة صنف الأساس من قبل صنف المشتق.

```
public final double getSalary() {
        return salary;
}
```

- إذا حاول الصنف المشتق فئة فرعية override لطريقة final، يقوم المترجم بإنشاء خطأ.

- يجب التأكيد على استخدام طريقة خاصة ضمن الصنف المشتق بدلاً من نسخة معدلة من الصنف الأساس.

Protected Members

```
Package A1;
public class Shape
{
  private double height;  // To hold height.
  private double width;   //To hold width or base

  /**
   * The setValue method sets the data
   * in the height and width field.
   */

  public void setValues(double height, double width)
  {
     this.height = height;
     this.width = width;
  }
}
```

```
Package A2;

public class Rectangle extends Shape
{

   /**
    * The method returns the area
    * of rectangle.
    */

  public double getArea()
  {
   return height * width;
  //accessing protected members
  }
}
```

Protected Members

- توفر Java نوع وصول ثالث المحمية protected.
- يقع وصول الاعضاء المحميين ما بين الخاص والعامة.

- استخدام النوع المحمي protected بدلاً من الخاص يجعل بعض المهام أسهل.
- أي صنف مشتق من صنف أخر، أو في نفس الحزمة ، لها وصول غير مقيد إلى الأعضاء المحميين.
- من الأفضل دائمًا جعل جميع حقول المعطيات خاصة ثم توفير طرق عامة للوصول إليها.

- إذا لم يتم توفير محدد وصول لعضو في الصنف، فسيتم منح عضو الصنف وصولاً على مستوى الحزمة وهو الحال الافتراضي. أي يجوز لأي طريقة في نفس الحزمة الوصول إلى هذه الأعضاء .

- أعضاء الصنف المحميين:
  - يمكن الوصول إليها من خلال طرق في صنف فرعي .
  - وبالطرق في نفس الحزمة كما هو للصنف.

Protected Members

```
public class Person
{               private String   name;
                private double age;
                private String address;
                private boolean nationality;
                public Person()
                {System.out.println("constructor Parent run \"super class\" \n "); }
                public Person(String n, double age, String ad, boolean nat)
                {       name = n;     this.age=age;
                        address = ad; nationality= nat;    }
                public void setName(String n){name=n;}
                public void setAge(double a){age=a;}
                public void setAddress( String ad){address=ad;}
                public void setNationality(boolean b){nationality = b;}
                public String getName(){return name;}
                public double getAge(){return age;}
```

Protected Members

```
            public String getAddress(){return address;}
            public boolean getNationality(){return nationality;}
      public void printAllDatails(){System.out.println("name = "+name+ "\n   Age
"+ age +"\n   Address "+ address + "nationality = " +nationality);}
      }// end Person

 public class Student extends  Person
 {     private int stadyLevel;
       private String specialization;
       private double GPA;
       Student()
       {System.out.println("constructor Student run \"sub class\"\n");    }
       Student(String n, double age, String ad, boolean nat,int sL, String sp,
double gpa){/* The Student class has 7 parameters, 4 inherited from class Person
and 3 declared For this reason we will call the constructor of the parent class
and send it 4 parameters */
```

Protected Members

```
            super( n, age, ad, nat);
            stadyLevel=sL;        specialization=sp;
            GPA=gpa;      }
      //Override method printAllDatails

@Override
      public void printAllDatails()
      {       super.printAllDatails();
            System.out.println("\n stadyLevel = "+stadyLevel +"\n specialization
                  "+specialization+"\n GPA "+GPA); }

      /*public void printS() { System.out.println("\n name ="+ getName()+ "\n
age= "+ getAge()+"\n address ="+ getAddress()+ "\n nationality = "+
getNationality()+"\n stadyLevel = "+stadyLevel + "\n specialization
"+specialization+"\n GPA "+GPA);  }*/
```

Protected Members

```
        public void setStadyLevel(int stadyLevel) {this.stadyLevel = stadyLevel;}
        public int getStadyLevel() {return stadyLevel; }
        public void setSpecialization(String specialization) {
        this.specialization = specialization;    }
        public String getSpecialization() {return specialization;    }
        public void setGPA(double gpa) {  GPA = gpa;    }
        public double getGPA() {return GPA;}
}// end class Student

public class Employee extends Person
{       double salary;              double rank;        String job;
        Employee()    {}
        Employee(String n, double age, String ad, boolean nat, double sa,double ra,
                String jo)    {       super(n, age, ad, nat);    salary=sa;
                rank=ra;                    job=jo;       }
```

Protected Members

```
    //Override method printAllDatails
    @Override
    public void printAllDatails()
    {       super.printAllDatails();
            System.out.println("\n rank "+rank+"\n job "+job +"\n salary = "
    + getSalary());      }
  //If the method signature is changed in the base class,
    // Java will alert that this method has been Override
    //2public abstract double getSalary();
    public  double getSalary() {return salary;}
    public void setSalary(double salary) {  this.salary = salary;}1*/
    public double getRank() {return rank;    }
    public void setRank(double rank) {this.rank = rank;   }
    public String getJob() {return job;      }
            public void setJob(String job) {  this.job = job;}
}// end class Employee
```

Protected Members

```
      public class SataredEmployee extends Employee
{double bonus; double deduction;
 public SataredEmployee(){}
public SataredEmployee(String n, double age, String ad, boolean nat, double sa,
double ra, String jo, double bo, double det)
{      super(n, age, ad, nat, sa, ra, jo);bonus=bo; deduction=det; }

@Override
public double getSalary() {return salary + bonus-deduction;}
@Override
public void printAllDatails()
{      super.printAllDatails();
       System.out.println("\n bonus = "+bonus +"\n deduction "+deduction+ "Salary
= "+ getSalary() );}
}// end class SataredEmployee
```

Protected Members

```
public class HourlyEmployee extends Employee
{       double houreRate;           double numberOfHours;
        public HourlyEmployee() {   }
        // TODO Auto-generated constructor stub
public HourlyEmployee(String n, double age, String ad, boolean nat,double sa,
double ra, String jo, double hR, double nOH){super(n, age, ad, nat, sa, ra, jo);
              houreRate=hR;                         numberOfHours=nOH;   }
        @Override
        public void printAllDatails()
        {       super.printAllDatails();
System.out.println("\n houreRate = "+ houreRate+"\n numberOfHours"+numberOfHours +
"\n Salary = "+getSalary() );      }
        @Override
        public double getSalary() {return houreRate * numberOfHours;}

}// end class HourlyEmployee
```

Protected Members

```
public class StudentTest
{ public static void main(String [] argc){

/*  When creating an object of a child class without parameter,
 *  it will call the constructor of the parent class
 *  and then the constructor of the child */
Student stu0 = new Student();

// Create an object of class Person
//and call the public methods  to print the default value

Person p1=new Person("ahmad0",31,"tartus", false);// p1.printAllDatails();
System.out.println("---");
p1.printAllDatails();
/* Create  an object of class Student and call the public method printS() to print
its data*/
Student stu1=new Student("ahmad",33.5,"lattakia", true, 4, "IT", 4.2);
stu1.printAllDatails();
```

Protected Members

```
Employee e1= new Employee("ali",22.5, "hama", true, 200000,1.1,"Engineer");
e1.printAllDatails();
/* Appeal Override printAllDatails method 1 */
//Create an object of class SataredEmployee and call the public methods

SataredEmployee se1= new SataredEmployee("adam",33.3, "syr",true , 3000,2.2,
"Eng", 700,200);
se1.printAllDatails();

// object of type Employer that also reference to an object of type Employee
Employee e11= new Employee("nader",22.5,"tartus", true, 300000,3.1,"Engineer");
e11.printAllDatails();

//object of type Employer that also reference to an object of type SataredEmployee
Employee e2= new SataredEmployee("adam",33.3, "syr",true , 3000,2.2, "Eng",
700,200); e2.printAllDatails(); /*1 */
// getSalary() from SataredEmployee Here, the method must have a root in the base
class, otherwise it will not be called
```

Protected Members

```
//A reference from derived class cannot refer to a base class
System.out.println("\n\n");
HourlyEmployee hE1= new HourlyEmployee("Adam2", 33, "latakia", true, 10000, 1.1,
"eng", 20,100);
hE1.printAllDatails();
}// end main
```

Protected Members

```
constructor Parent run "super class"

constructor Student run "sub class"

---
name = ahmad0
  Age 31.0
  Address tartusnationality = false
name = ahmad
  Age 33.5
  Address lattakianationality = true

 stadyLevel = 4
 specialization IT
 GPA 4.2
name = ali
  Age 22.5
  Address hamanationality = true
rank 1.1
 job Engineer
salary = 200000.0
name = adam
  Age 33.3
  Address syrnationality = true

rank 2.2
 job Eng
 salary = 3500.0

 bonus = 700.0
 deduction 200.0Salary = 3500.0
name = nader
  Age 22.5
  Address tartusnationality = true
rank 3.1
 job Engineer
 salary = 300000.0
name = adam
  Age 33.3
  Address syrnationality = true

 rank 2.2
 job Eng
 salary = 3500.0

 bonus = 700.0
 deduction 200.0Salary = 3500.0

name = Adam2
  Age 33.0
  Address latakianationality = true

 rank 1.1
 job eng
 salary = 2000.0

 houreRate = 20.0
 numberOfHours 100.0
 Salary = 2000.0
```

Dialog Boxes

3.10 GUI &Graphics

رقم الفقرة	العنوان
3.9	استخدام مربعات الحوار: المدخلات والمخرجات الأساسية مع مربعات الحوار
4.14	إنشاء رسومات بسيطة عرض الخطوط ورسمها على الشاشة
5.10	رسم المستطيلات والأشكال البيضاوية استخدام الأشكال لتمثيل البيانات
6.13	الألوان والأشكال المعبأة رسم قوس قذح ورسومات عشو ائية
7.13	رسم الأقواس رسم الحلزونات بالأقواس
8.18	استخدام الكائنات مع الرسومات تخزين الأشكال ككائنات
9.8	عرض النص والصورباستخدام الملصقات توفير معلومات الحالة
10.8	الرسم باستخدام تعدد الأشكال: تحديد أوجه التشابه بين الأشكال التمرين
14.17	توسيع الواجهة: استخدام مكونات واجهة المستخدم الرسومية ومعالجة الأحداث

Using Dialog Boxes

- تحتوي حزمة javax.swing على العديد من الأصناف التي تساعدك على إنشاء واجهات مستخدم رسومية GUIs
- تسهل مكونات واجهة المستخدم الرسومية إدخال البيانات من قبل مستخدم البرنامج وعرض المخرجات للمستخدم.
- المنهج showMessageDialog() من الصنف JOptionPane من الحزمة javax.swing
- يعرض مربع حوار showMessageDialog(arg1,arg2) العرض نافذة ويتطلب بارامترين، أو تظهر نافذة فارغة وفق الحالة الافتراضية.
  - الأول arg1 يساعد تطبيق Java على تحديد مكان وضع مربع الحوار.إذا كانت الوسيطة الأولى null، فسيتم عرض مربع الحوار في وسط الشاشة.
  - الوسيط الثاني arg2 هو السلسلة String المراد عرضها في مربع الحوار.

Using Dialog Boxes

- ✓ منهج showMessageDialog() من الصنف JOptionPane هو منهج ثابتة static method.
- ✓ غالبًا ما تحدد مثل هذه المناهج المهام المستخدمة بشكل متكرر أو خدماتي.
- ✓ يتم استدعاؤها عادةً باستخدام اسم فئة المنهج متبوعًا بنقطة (.) واسم الطريقة و والوسطاء مابين قوسين، كما يلي:

ClassName.methodName(arguments)

- ✓ لاحظ أنك لم تقم بإنشاء كائن من الصنف JOptionPane لاستخدام static method المنهج الثابت showMessageDialog()

Using Dialog Boxes

- ✓ يسمح صندوق حوار الإدخال input- dialog للمستخدم بإدخال البيانات في البرنامج.
- ✓ يعرض المنهج ()showInputDialog من الصنف JOptionPane مربع حوارالإدخال.
  - يحتوي على مؤشر الادخال وحقل (يُعرف باسم حقل النص text field) حيث يمكن للمستخدم إدخال نص.
- ✓ المنهج ()showInputDialog يقوم بإرجاع سلسلة String تحتوي على الأحرف التي كتبها المستخدم في حقل النص.
- ✓ إذا تم الضغط على زرإلغاء Cancel في مربع الحوارأو ضغط على مفتاح *Esc* key على لوحة المفاتيح، فإن الطريقة ترجع null للدلالة على تجاهل الادخال.

- ✓ يقوم method المنهج من الصنف الثابت static String بإعادة String منسقاً.

- ✓ ينسيق المنهج format كما يعمل المنهج System.out.printf بإرجاع سلسلة منسقة، باستثناء أن هذا التنسيق يُرجع السلسلة المنسقة return بدلاً من عرضها في نافذة أوامر.

Dialog Boxes

3.10 GUI &Graphics

```
1        // Fig. 3.17: Dialog1.java
2        // Printing multiple lines in dialog box.
3   import javax.swing.JOptionPane; // import class JOptionPane
4
5   public class Dialog1
6   {
7    public static void main( String args[] )
8      {
9        // display a dialog with the message
10  JOptionPane.showMessageDialog( null,
                                "Welcome\nto\nJava" );
11       } // end main
12   } // end class Dialog1
```

Using Dialog Boxes

3.10 GUI &Graphics

```
// Fig. 3.18: NameDialog.java
// Basic input with a dialog box.
import javax.swing.JOptionPane;
public class NameDialog
{
public static void main( String args[] )
    {
     // prompt user to enter name
     String name = JOptionPane.showInputDialog( "What is your name?" );
     // create the message
     String message = String.format( "Welcome, %s, to Java Programming!", name );
      // display the message to welcome the user by name
     JOptionPane.showMessageDialog( null, message );
    } // end main
} // end class NameDialog
```

Displays an input Dialog to obtain data from user

Creates a formatted String containing the user entered in the user entered in the input dialog

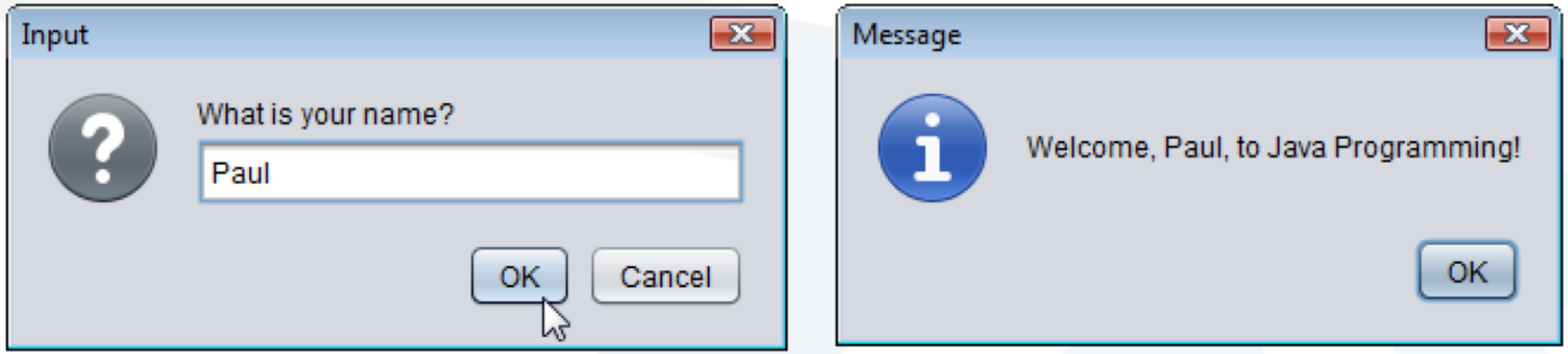


Fig. 3.18 | Obtaining user input from a dialog. (Part 2 of 2.)

نظرًا لأن الطريقة showInputDialog تُرجع سلسلة ، يجب عليك تحويل السلسلة التي يدخلها المستخدم إلى عدد لاستخدامها في العمليات الحسابية. تأخذ الطريقة الثابتة parseInt للفئة Integer (package java.lang) وسيطة سلسلة تمثل عددًا صحيحًا وترجع القيمة كـ int.

4.17 GUI &Graphics

- القسم الأول الرسومات في الجافا Graphics Java في هذا القسم نتعرف على الأدوات التي توفرها لغة جافا للرسم والتلوين على شاشة البرنامج.

- القسم الثاني واجهات المستخدم الرسومية (Graphical User Interface . GUI) في هذا القسم سنتحدث عن مجموعة من أدوات لغة جافا الخاصة بتصميم واجهات المستخدم أو شاشات البرنامج، والتي تساعد المستخدم على التفاعل مع البرنامج بصورة أسهل وأبسط ولاتتطلب المعرفة الدقيقة بالبرمجة ولغاتها.

Dialog Boxes

3.10 GUI &Graphics

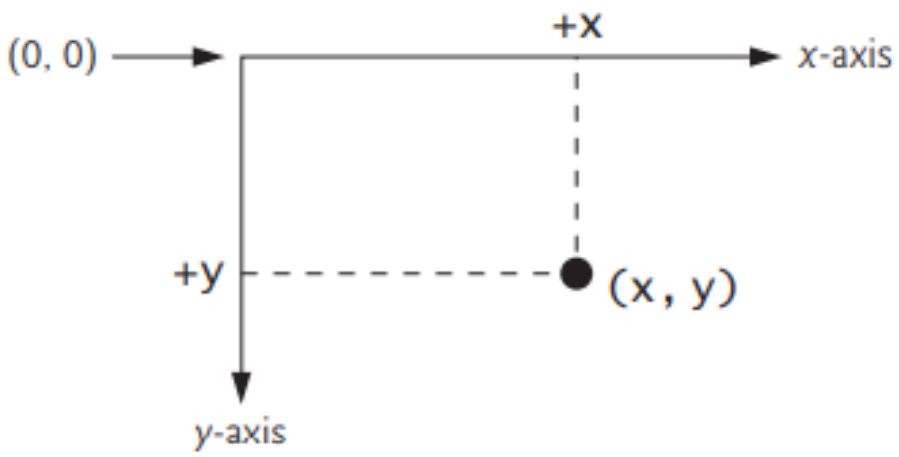
رقم الفقرة	العنوان
3.9	استخدام مربعات الحوار: المدخلات والمخرجات الأساسية مع مربعات الحوار
4.14	إنشاء رسومات بسيطة عرض الخطوط ورسمها على الشاشة
5.10	رسم المستطيلات والأشكال البيضاوية استخدام الأشكال لتمثيل البيانات
6.13	الألوان والأشكال المعبأة رسم قوس قذح ورسومات عشوائية
7.13	رسم الأقواس رسم الحلزونات بالأقواس
8.18	استخدام الكائنات مع الرسومات تخزين الأشكال ككائنات
9.8	عرض النص والصور باستخدام الملصقات توفير معلومات الحالة
10.8	الرسم باستخدام تعدد الأشكال: تحديد أوجه التشابه بين الأشكال التمرين
14.17	توسيع الواجهة: استخدام مكونات واجهة المستخدم الرسومية ومعالجة الأحداث

Creating Simple Drawings

Java's Coordinate System

- ✓ •نظام الإحداثيات في Javaعبارة عن مخطط scheme لتحديد النقاط على الشاشة.
- ✓ الزاوية العلوية اليسارية من نافذة واجهة المستخدم الرسومية لها الإحداثيات (0،0).
- ✓ يتكون زوج الإحداثيات من x-coordinate (الإحداثي الأفقيhorizontal coordinate)وإحداثي y-coordinate(إحداثي رأسي vertical coordinate).

•إحداثي x هو الموقع الأفقي الذي يتم التحرك عليه من اليسارإلى اليمين.
•إحداثي y هو الموقع الرأسي الذي يتم التحرك عليه من أعلى إلى أسفل.
• تشيرالإحداثيات إلى مكان عرض الرسومات على الشاشة.
يصف المحور x كل الإحداثيات الأفقية ، ويصف المحور y كل الإحداثيات الرأسية.
• تقاس وحدات التنسيق بالبكسل. يشير المصطلح المعروف بكسل إلى "عنصر الصورة"، وهو أصغروحدة دقة قياس في شاشة العرض.



نستخدم فئتين لبرمجة واجهة المستخدم وهما java.awt و javax .swing.

- قدمت لغة الجافا برمجة واجهات المستخدم لأول مرة وكانت كل الأصناف موجودة في مكتبة تسمي awt والتي تقوم بضبط اعدادات البرنامج تلقائيا حسب المنصة التي يتم تشغيل البرنامج عليها، وهى تنفع في بناء واجهات مستخدم بسيطة، ولكن لا تجدي نفعا في بناء واجهات مستخدم محترفة ومتميزة.
- تم استبدال حزمة awt بحزمة اكثر تميزوكفاءة هي فئة swing وعرفت ب light components weight اي المكونات الخفيفة لإنها تعتمد علي انشاء الكائنات من دون الإعتماد علي منصة التشغيل على خلاف awt والتي تعرف heavy components weight المكونات الثقيلة لانها تعتمد وتتعامل مع منصة التشغيل.
- ومن اجل التفريق بين اصناف حزمة awt واصناف حزمة swing يتم اضافة السابقة J قبل اسم كل صنف من اصناف فئة swing .
- طريقة تصميم وترتيب ال GUI Application Program Interface تعتبر من افضل الأمثلة علي استخدام الوراثة والأصناف والواجهات .Interfaces.

كل برنامج رسومي يستخدم نافذة إطار frame window أو أكثر ولكل نافذة اطار شريط عنوان titel bar وحدود border لكي تظهر الإطار نستخدم الصنف JFream من الحزمة javax.swing ويجب:

1- إنشاء كائن من JFream وفق JFrame appli = new JFrame("First");

2- تجديد مقاس الإطار من الطريقة setSize . appli.setSize(300, 300);

3- إضافة الرسمة أو ماتم تجميعة إلى الإطار appli.add(panel);

4- جعل الإطار مرئي تستخدم الطريقة show لجعل مدير عرض النافذة window manager يعرضها.

appli.setVisible(true);

5- عند تنفيذ البرنامج يتم إظهار الإطار وينتهي تنفيذ main ولكن يظل البرنامج يعمل والإطار ظاهر على الشاشة ويمكن تحريكة وتغيير حجمه و ... ، وعند إغلاق نافذة الإطار بالضغط على أيقونة الغلق من شريط العنونه يظل البرنامج يعمل ولايحدث شيئ سوى إختفاء الإطار، ومن أجل إنهاء البرنامج يجب إستخدام System.exit(0) والتي يجب أن تكون بنهاية main ولكن تخلق مشكلة جديدة وهي ظهور النافذة للحظة وجيزة وينتهي فوراً والرغبة هي انهاء البرنامج عند يضغط المستخدم على أيقونة الغلق في شريط العنوان وهنا نجد اسهل طريقة استخدام المنهج : appli.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

أو معالجة حدث النقر على أيقونة الغلق من أجل انهاء البرنامج إضافة على إغلاق النافذة

Creating Simple Drawings

4.17 GUI &Graphics

```java
// Fig. 4.18: DrawPanel.java
// Using drawLine to connect the corners of a panel.
import java.awt.Graphics;
import javax.swing.JPanel;

public class DrawPanel extends JPanel
{
   // draws an X from the corners of the panel
   public void paintComponent( Graphics g )
   {
      // call paintComponent to ensure the panel displays correctly
      super.paintComponent( g );

      int width = getWidth(); // total width
      int height = getHeight(); // total height

      // draw a line from the upper-left to the lower-right
      g.drawLine( 0, 0, width, height );

      // draw a line from the lower-left to the upper-right
      g.drawLine( 0, height, width, 0 );
   } // end method paintComponent
} // end class DrawPanel
```

- Import the classes `Graphics` and `JPanel` for use in this source code file.
- `DrawPanel` inherits the existing capabilities of class `JPanel`
- `paintComponent` *must* be displayed as shown here
- This should be the first statement in method `paintComponent`
- Determines the width and height of the `DrawPanel` with inherited methods
- Draws a line from the top-left to the bottom-right of the `DrawPanel`
- Draws a line from the bottom-left to the top-right of the `DrawPanel`

Fig. 4.18 | Using `drawLine` to connect the corners of a panel.

Creating Simple Drawings

4.17 GUI &Graphics

```java
// Fig. 4.19: DrawPanelTest.java
// Application to display a DrawPanel.
import javax.swing.JFrame;

public class DrawPanelTest
{
   public static void main( String[] args )
   {
      // create a panel that contains our drawing
      DrawPanel panel = new DrawPanel();

      // create a new frame to hold the panel
      JFrame application = new JFrame();

      // set the frame to exit when it is closed
      application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );

      application.add( panel ); // add the panel to the frame
      application.setSize( 250, 250 ); // set the size of the frame
      application.setVisible( true ); // make the frame visible
   } // end main
} // end class DrawPanelTest
```

Imports class `JFrame` for use in this source code file

Creates a `JFrame` in which the `DrawPanel` will be displayed

Terminate application when window closes

Attach the `DrawPanel` to the `JFrame`

Sets the size of the `JFrame`

Displays the `JFrame` on the screen

Fig. 4.19 | Creating `JFrame` to display `DrawPanel`. (Part 1 of 2.)

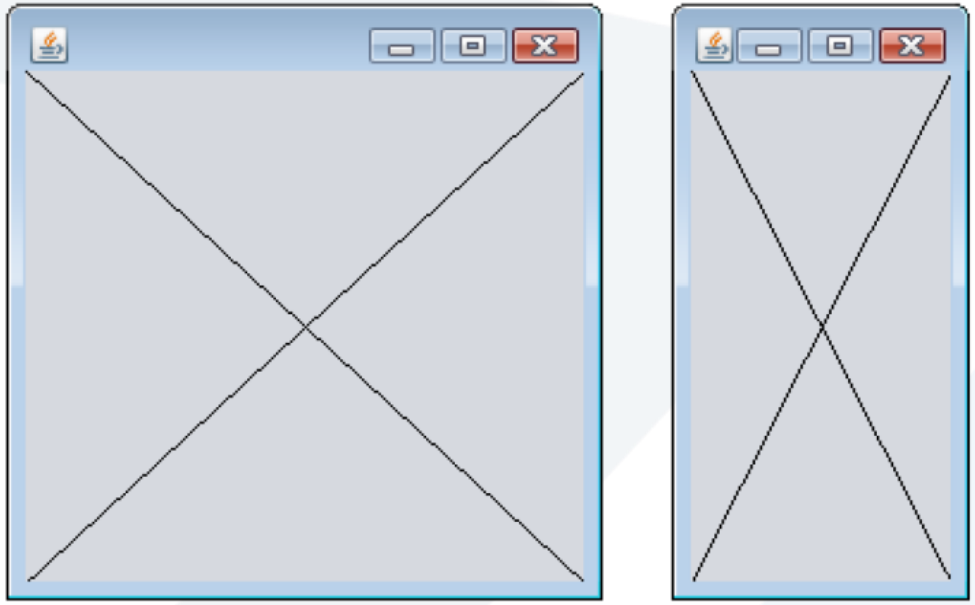


Fig. 4.19 | Creating `JFrame` to display `DrawPanel`. (Part 2 of 2.)

Java's Coordinate System

- class Graphics من الحزمة (java.awt)، والتي توفر طرقًا مختلفة لرسم النص والأشكال على الشاشة.
- الصنف JPanel من الحزمة (javax.swing)، والتي توفر مساحة يمكن الرسم عليها.

```
public class DrawPanel extends JPanel
```

extends الكلمة الأساسية للإشارة إلى أن الصنف DrawPanel هو نوع محسن من JPanel.

- الكلمة الأساسية extends تمثل ما يسمى بعلاقة الوراثة التي يبدأ فيها صنفنا الجديد DrawPanel بالأعضاء الحاليين (البيانات والأساليب) من فئة Jpanel .
- كل لوحة JPanel بما في ذلك DrawPanel، لديها طريقة paintComponent.
- ينادي النظام تلقائيًا في كل مرة يحتاج فيها إلى عرض DrawPanel. المنهج paintComponent() ويجب التصريح عنها public void paintComponent(Graphics g)، خلاف ذلك، لن يسمح النظام بمناداتها والعبارة الأولى فيها عندما تكتبها (تحملها تحميلاً زائداً) هي super.paintComponent(g);.
- يتم استدعاء هذه الطريقة عندما يتم عرض JPanel لأول مرة على الشاشة، وعندما يتم تغطيتها ثم الكشف عنها بواسطة نافذة أخرى على الشاشة، وعندما يتم تغيير حجم النافذة التي تظهر فيها.

GUI and Graphics
Creating Simple Drawings

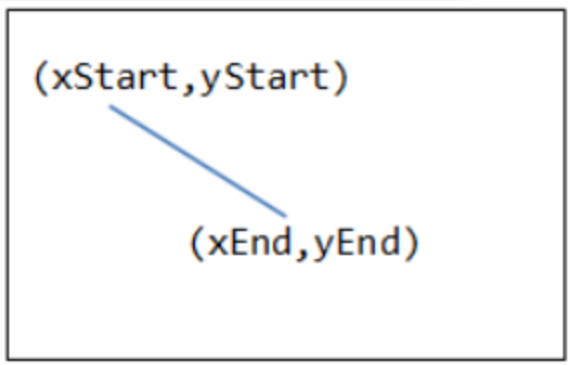
- تقوم أساليب getWidth و getHeight من الصنف JPanel بإرجاع عرض وارتفاع JPanel على التوالي.
- طريقة الرسم drawLine تحتاج أربع وسطاء، أول اثنين هما إحداثيات x و y تعبر عن نقطة البداية نهاية واحدة ، وآخر وسيطتين هما إحداثيات نقطة النهاية الأخرى وترسم خطًا بين نقطتي البداية والنهاية.
- لعرض لوحة الرسم DrawPanel على الشاشة ، توضع في نافذة **place it in a window** .
- يتم إنشاء نافذة مع كائن من فئة JFrame.
- أسلوب JFrame setDefaultCloseOperation مع الوسيطة JFrame.EXIT_ON_CLOSE يشير إلى أنه يجب إنهاء التطبيق عندما يغلق المستخدم النافذة.
- تقوم add method من JFrame بإرفاق DrawPanel أو (أي مكون GUIآخر) بإطار JFrame .
- تأخذ setSize method من JFrame معلمتين تمثلان عرض وارتفاع JFrame، على التوالي.
- تعرض setVisible method من JFrame مع الوسيط true الإطار JFrame.
- عندما يتم عرض **JFrame**، يتم استدعاء طريقة **paintComponent** الخاصة بـ **DrawPanel** ضمنيًا.

Drawing Rectangles, Ovals

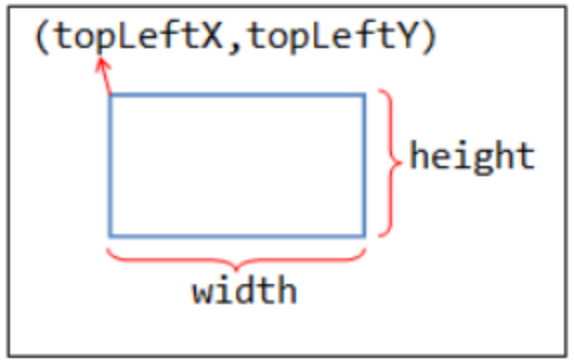
- يتطلب المنهج drawRect أربع وسطاء. يمثل أول اثنان إحداثيات x و y للزاوية اليسارية العلوية للمستطيل ، ويمثل الخياران التاليان عرض المستطيل وارتفاعه.
- البرنامج يرسم القطع الناقص. يقوم بإنشاء مستطيل وهمي يسمى المستطيل المحيط ويضع بداخله شكل بيضاوي يلامس نقاط المنتصف لجميع الجوانب الأربعة.
- يتطلب أسلوب drawOval نفس الوسطاء الأربع مثل طريقة drawRect.
- مثلاً لرسم مستطيل بحواف دائرية و مثلة ممتلئ باللون الاقتراضي نكتب:

```
g.drawRoundRect(200,150,60,50, 15,15);
g.fillRoundRect(290,150,60,50,30,40);
```

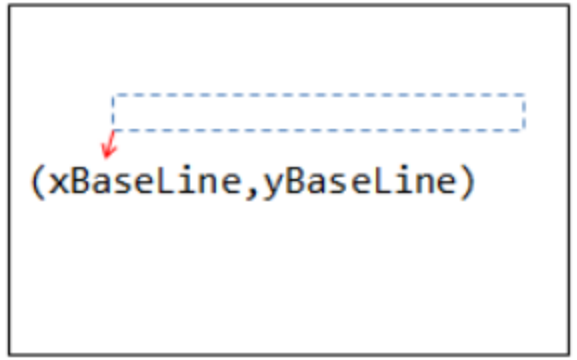
- في JOptionPane، يجب عليك استخدام \n لبدء سطر جديد من النص ، بدلاً من %n ،
- يستخدم منهج parseInt لتحويل السلسلة التي أدخلها المستخدم إلى عدد صحيح ويخزن النتيجة في متغير للنوع الصحسح يتم اختياره.



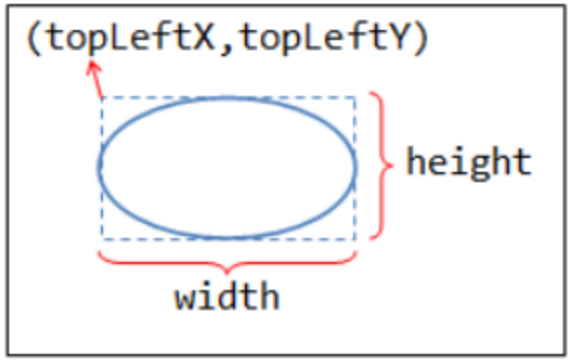
drawLine()



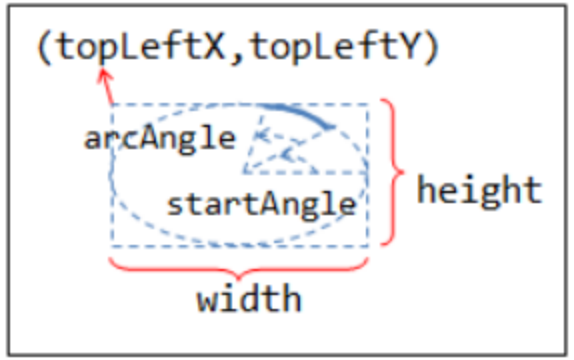
drawRect()



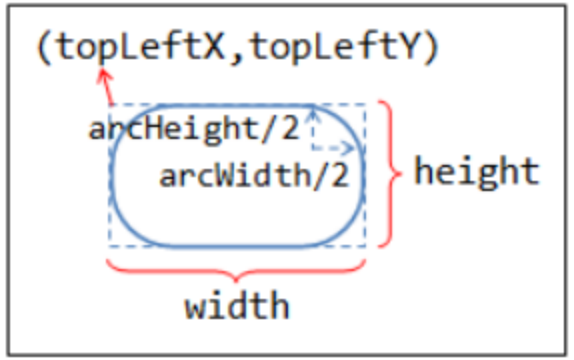
drawString()



drawOval()



drawArc()



drawRoundRect()

Drawing Rectangles and Ovals

5.26 GUI &Graphics

```java
// Fig. 5.26: Shapes.java
// Demonstrates drawing different shapes.
import java.awt.Graphics;
import javax.swing.JPanel;

public class Shapes extends JPanel
{
   private int choice; // user's choice of which shape to draw

   // constructor sets the user's choice
   public Shapes( int userChoice )
   {
      choice = userChoice;
   } // end Shapes constructor

```

Fig. 5.26 | Drawing a cascade of shapes based on the user's choice. (Part 1 of 2.)

```java
    // draws a cascade of shapes starting from the top-left corner
    public void paintComponent( Graphics g )
    {
        super.paintComponent( g );

        for ( int i = 0; i < 10; i++ )
        {
            // pick the shape based on the user's choice
            switch ( choice )
            {
                case 1: // draw rectangles
                    g.drawRect( 10 + i * 10, 10 + i * 10,
                       50 + i * 10, 50 + i * 10 );
                    break;
                case 2: // draw ovals
                    g.drawOval( 10 + i * 10, 10 + i * 10,
                       50 + i * 10, 50 + i * 10 );
                    break;
            } // end switch
        } // end for
    } // end method paintComponent
} // end class Shapes
```

Draws a rectangle starting at the *x-y* coordinates specified as the first two arguments with the width and height specified by the last two arguments

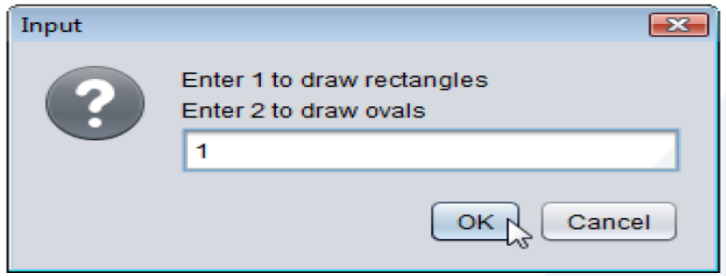
Draws an oval in the bounding rectangle starting at the *x-y* coordinates specified as the first two arguments with the width and height specified by the last two arguments

Fig. 5.26 | Drawing a cascade of shapes based on the user's choice. (Part 2 of 2.)

```java
// Fig. 5.27: ShapesTest.java
// Test application that displays class Shapes.
import javax.swing.JFrame;
import javax.swing.JOptionPane;

public class ShapesTest
{
   public static void main( String[] args )
   {
      // obtain user's choice
      String input = JOptionPane.showInputDialog(
         "Enter 1 to draw rectangles\n" +
         "Enter 2 to draw ovals" );

      int choice = Integer.parseInt( input ); // convert input to int

      // create the panel with the user's input
      Shapes panel = new Shapes( choice );

      JFrame application = new JFrame(); // creates a new JFrame

```

Fig. 5.27 | Obtaining user input and creating a `JFrame` to display `Shapes`. (Part 1 of 3.)

```
        application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        application.add( panel ); // add the panel to the frame
        application.setSize( 300, 300 ); // set the desired size
        application.setVisible( true ); // show the frame
    } // end main
} // end class ShapesTest
```



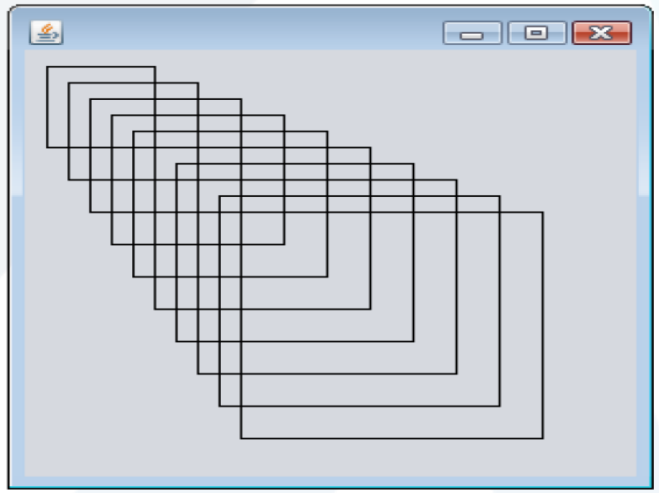


Fig. 5.27 | Obtaining user input and creating a `JFrame` to display `Shapes`. (Part 2 of 3.)

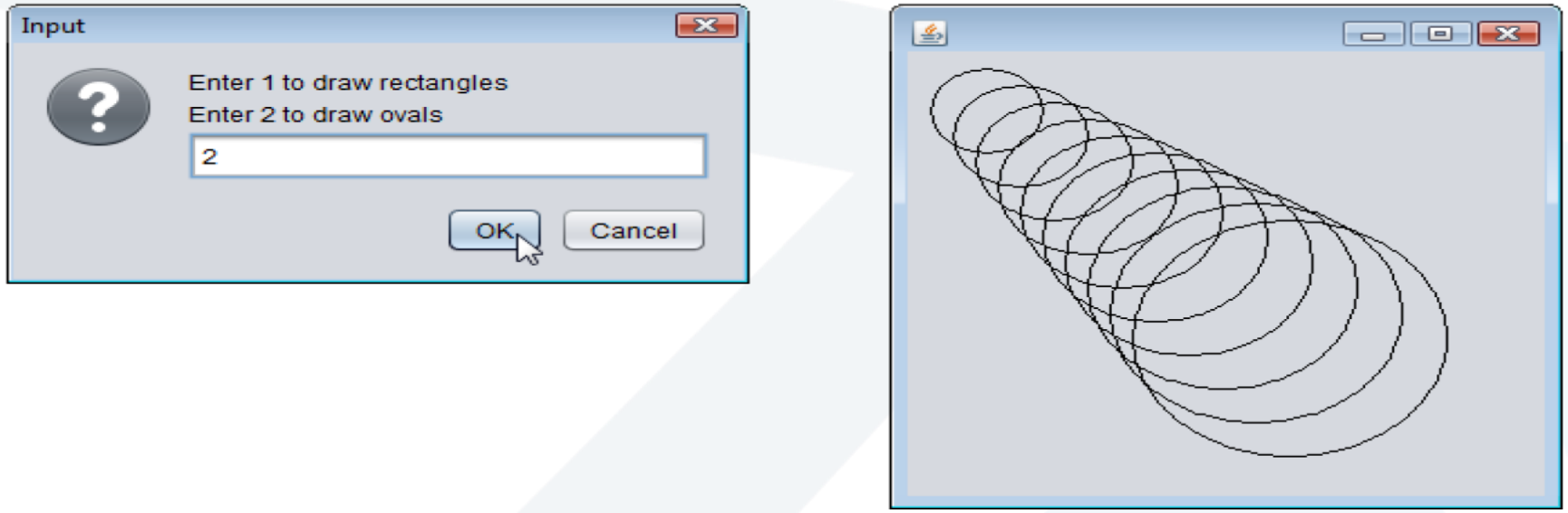


Fig. 5.27 | Obtaining user input and creating a `JFrame` to display `Shapes`. (Part 3 of 3.)

Java's Coordinate System

```
// Fig. 5.28: ShapesTest.java
// Obtaining user input and creating a JFrame to display Shapes.
import javax.swing.JFrame; //handle the display
import javax.swing.JOptionPane;
public class ShapesTest
{ public static void main(String[] args)
        {
        // obtain user's choice
String input = JOptionPane.showInputDialog(
"Enter 1 to draw rectangles " +
"Enter 2 to draw ovals");
int choice = Integer.parseInt(input); // convert input to int
```

```
        // create the panel with the user's input
        Shapes panel = new Shapes(choice);

        JFrame application = new JFrame(); // creates a new JFrame

        application.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        application.add(panel);
         application.setSize(300, 300);
        application.setVisible(true);
         }
} // end class ShapesTest
```

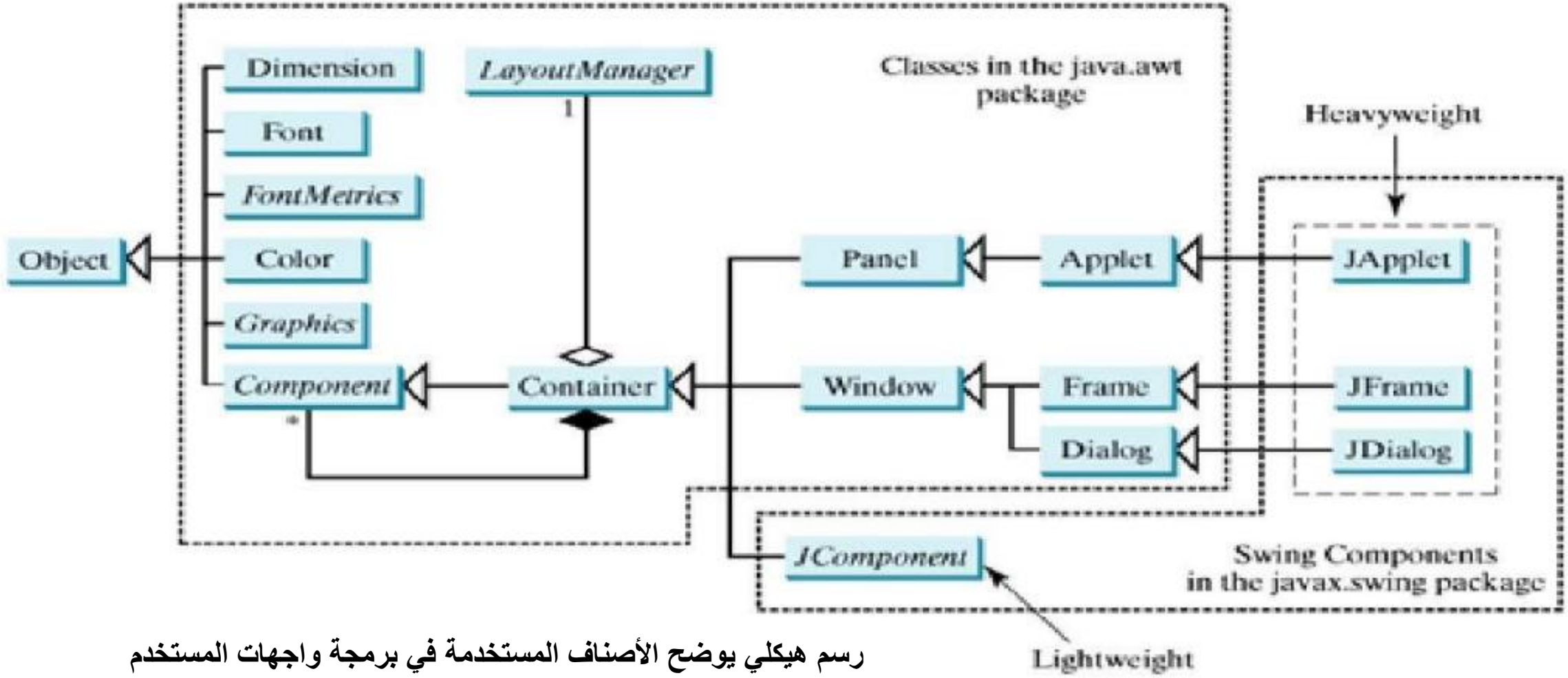
```
// Fig. 5.26: Shapes.java
// Demonstrates drawing different shapes.
import java.awt.Graphics;
import javax.swing.JPanel;
public class Shapes0  extends JPanel
{   private int choice; // user's choice of which shape to draw
    // constructor sets the user's choice
  public Shapes0( int userChoice )
  {     choice = userChoice;              } // end Shapes constructor
    // draws a cascade of shapes starting from the top left corner
  public void paintComponent( Graphics g )
  {             super.paintComponent( g );
        for ( int i = 0; i <= 10; i++ )
    {      // pick the shape based on the user's choice
      switch ( choice )
      {        case 1: // draw rectangles
          g.drawRect( 10+ i * 10, 10+ i * 10, 50 + i * 10, 50+ i * 10);        break;
```

```
case 2: // draw ovals
    g.drawOval( 10+ i * 10, 10+ i * 10,    50 + i * 10, 50+ i * 10);        break;
case 3: //draw line fig4.20a Lines fanning from a corner. P138
            g.drawLine( 10, 10,260 - i * 25, 10 + i * 25); break;
case 4:  //draw line fig4.20b Lines fanning from a corner.P138 9th
            g.drawLine( 10 , 10, 260 - i * 25, 10 + i * 25);
            g.drawLine( 260, 260,260 - i * 25, 10 + i * 25);
            g.drawLine( 10, 260, 10 + i * 25, 10 + i * 25);
            g.drawLine( 260, 10, 10 + i * 25, 10 + i * 25); break;
case 5:  // draw Fig4.21a Line art with loops and drawLine. P138 9th
            g.drawLine( 10 , 10 + i*25, 10 + i * 25, 260);  break;
case 6: // draw Fig4.21b Line art with loops and drawLine. P138 9th
            g.drawLine( 10 , 10 + i*25,10 + i * 25, 260);
            g.drawLine( 10 + i*25, 10, 260 , 10 + i * 25);
            g.drawLine( 260 -i*25, 10, 10 , 10 + i * 25);
            g.drawLine( 10+i*25, 260 , 260 ,260 - i*25);  break;
 case 7:  // draw  concentric circles Fig. 5.29  P190 10th
            g.drawOval( 130 - i * 10, 130 - i * 10,i * 20,  i * 20 );
```

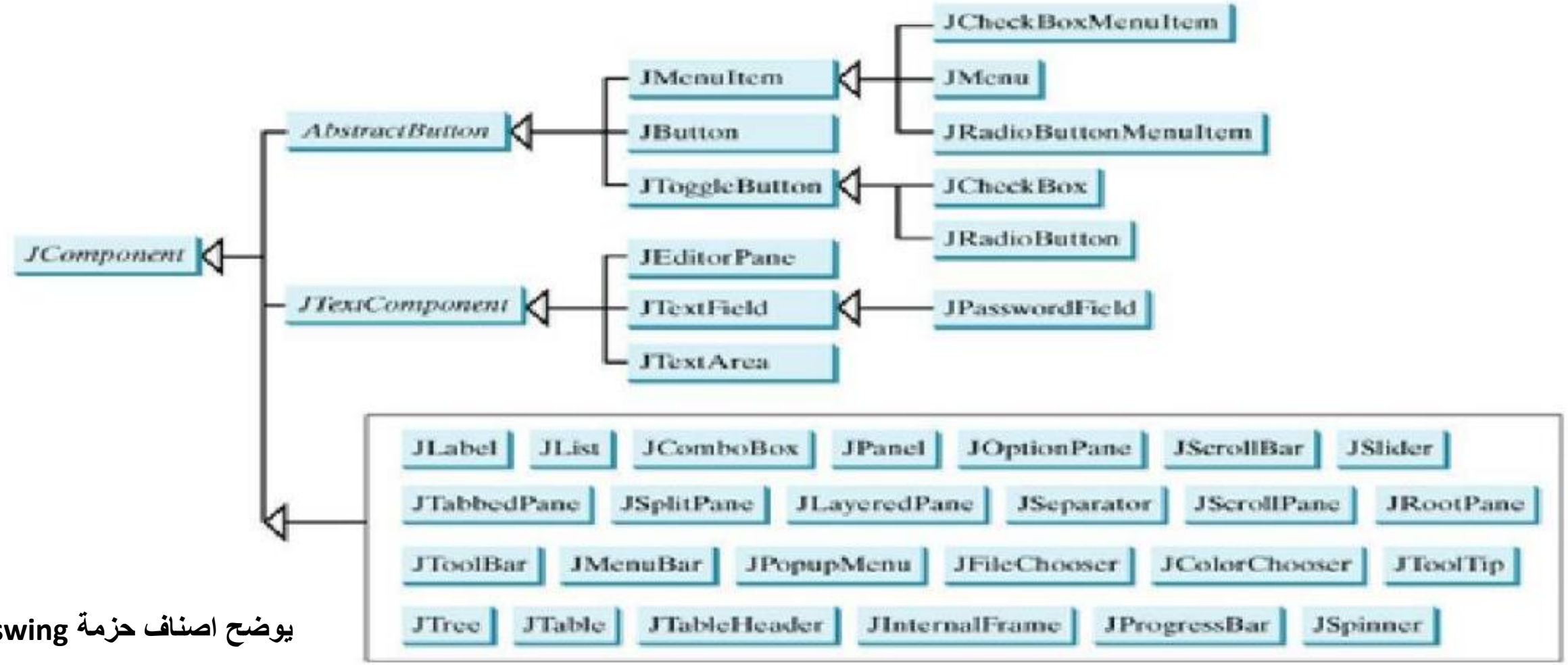
```
case 7:  // draw  concentric circles Fig. 5.29  P190 10th
        g.drawOval( 130 - i * 10, 130 - i * 10,i * 20,  i * 20 );
        // g.drawOval( 30 + i * 10, 30 + i * 10,200-i * 20, 200- i * 20 );
       int x1,y1,w,h;  x1=130 - i * 10;  y1=130 - i * 10;w= i * 20; h=i * 20;
        System.out.println("x1=" +x1+ " y1= "+y1+ " w= "+w+" h= "+h);  break;
case 8:  // draw  Draw a right triangle of stars, do not use the Graphics class
       for ( int a = 1; a <= 5; a++ )
                {
                        for (  int j = 1; j <= a; j++ )
                                {
                                        System.out.print( '*' );
                                } // end inner for
                        System.out.println();
                } break;        // end outer for
       } // end switch
   } // end for
  } // end method paintComponent
 } // end class Shapes
```

```
// Fig. 5.27: ShapesTest0.java              // Test application that displays class Shapes.
import javax.swing.JFrame;
import javax.swing.JOptionPane;
public class ShapesTest0
{  public static void main( String args[] )
  {// obtain user's choice
    String input = JOptionPane.showInputDialog(  "Enter 1 to draw rectangles\n" +"Enter 2 to draw ovals \n“
                  +"Enter 3 to draw 10 line \n“+"Enter 4 to draw 40line \n“ +"Enter 5 to draw Fig42.2a \n"
                 +"Enter 6 to draw Fig422b \n“+"Enter 7 to draw Fig. 5.28  P187 \n"+"Enter 8 to draw for \n");
     int choice = Integer.parseInt( input ); // convert input to int
     Shapes0 panel = new Shapes0( choice ); // create the panel with the user's input
    JFrame application = new JFrame(); // creates a new JFrame
    application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
    application.add( panel ); // add the panel to the frame
    application.setSize( 280,  305); // set the desired size
    application.setVisible( true ); // show the frame
  } // end main
} // end class ShapesTest
```

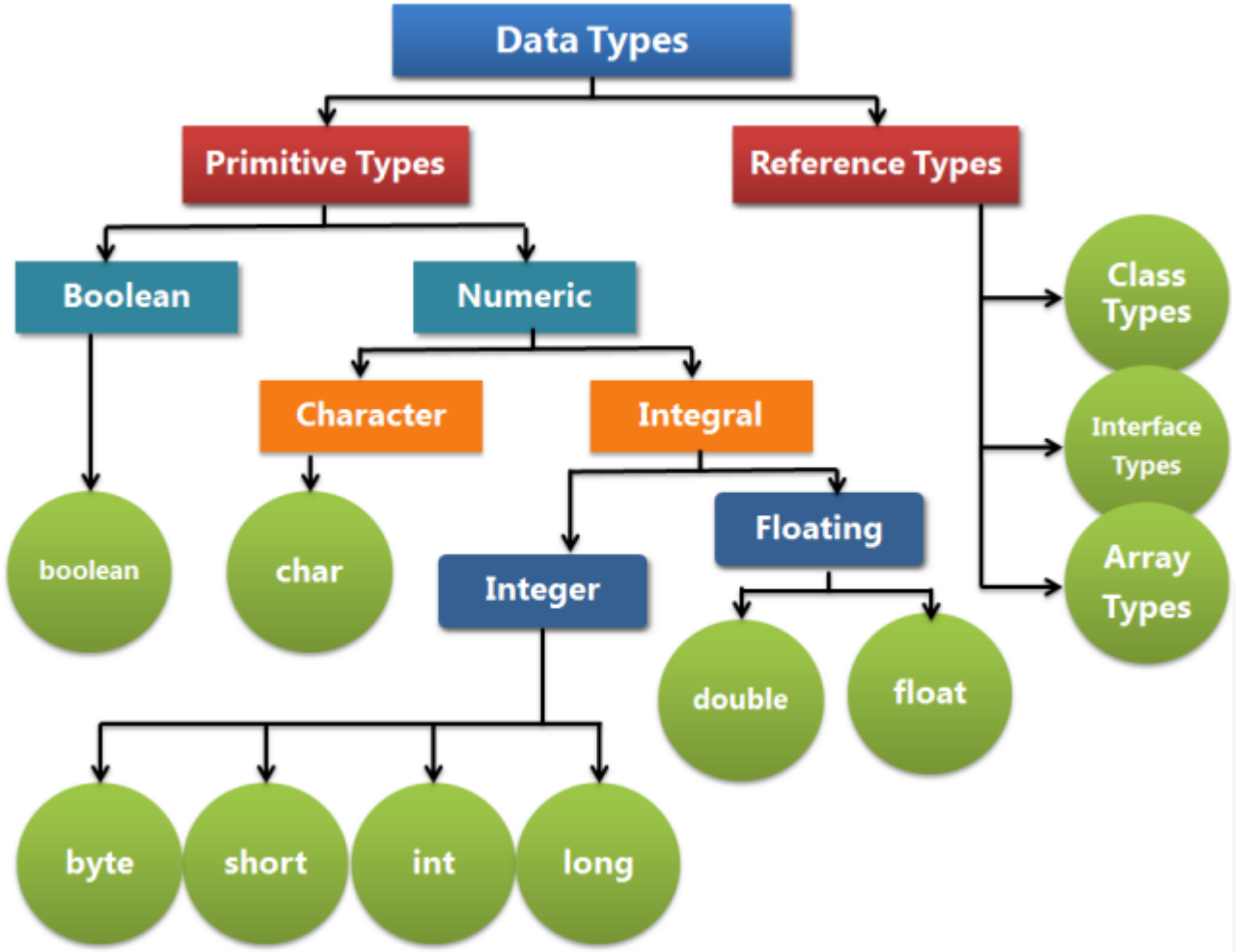
انتهت محاضرة الأسبوع 3

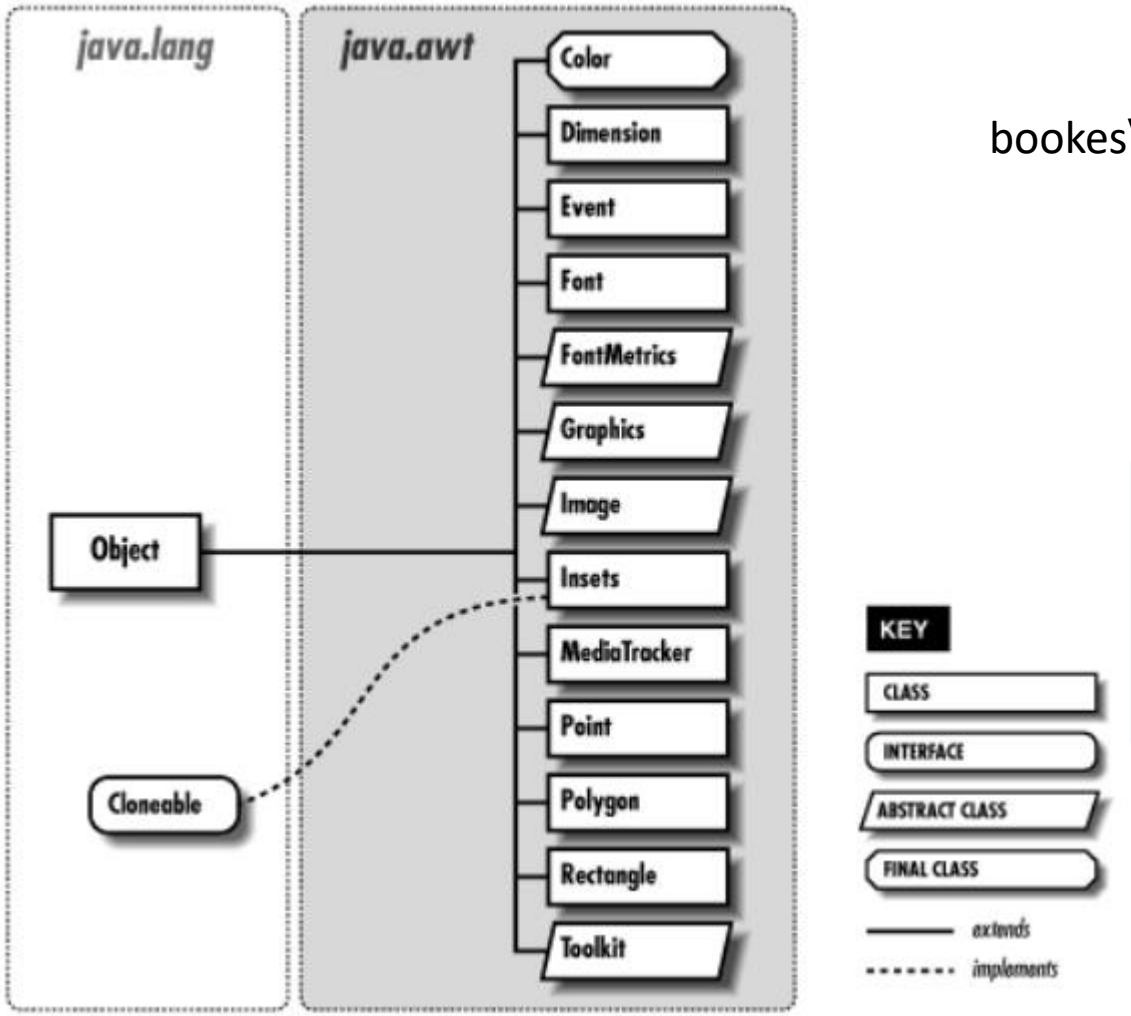


رسم هيكلي يوضح الأصناف المستخدمة في برمجة واجهات المستخدم



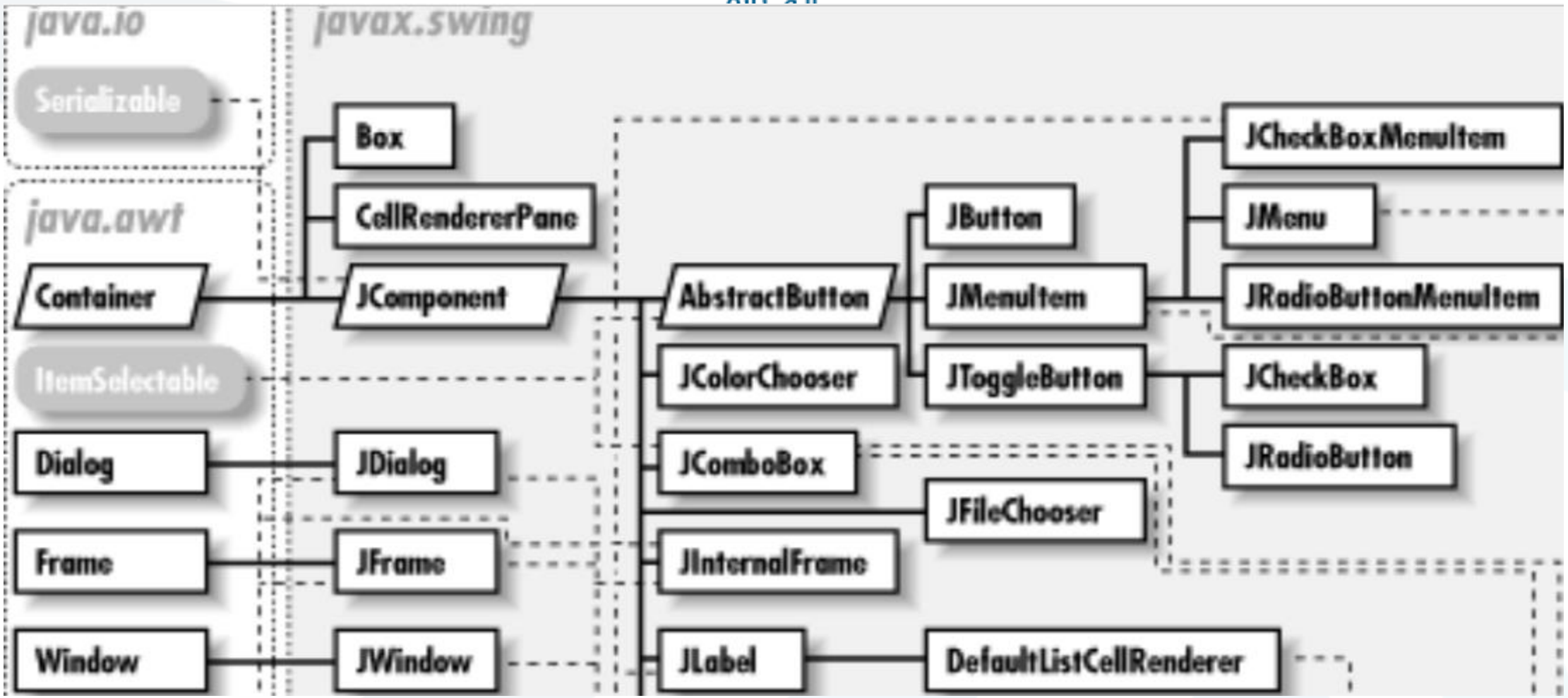
يوضح اصناف حزمة swing

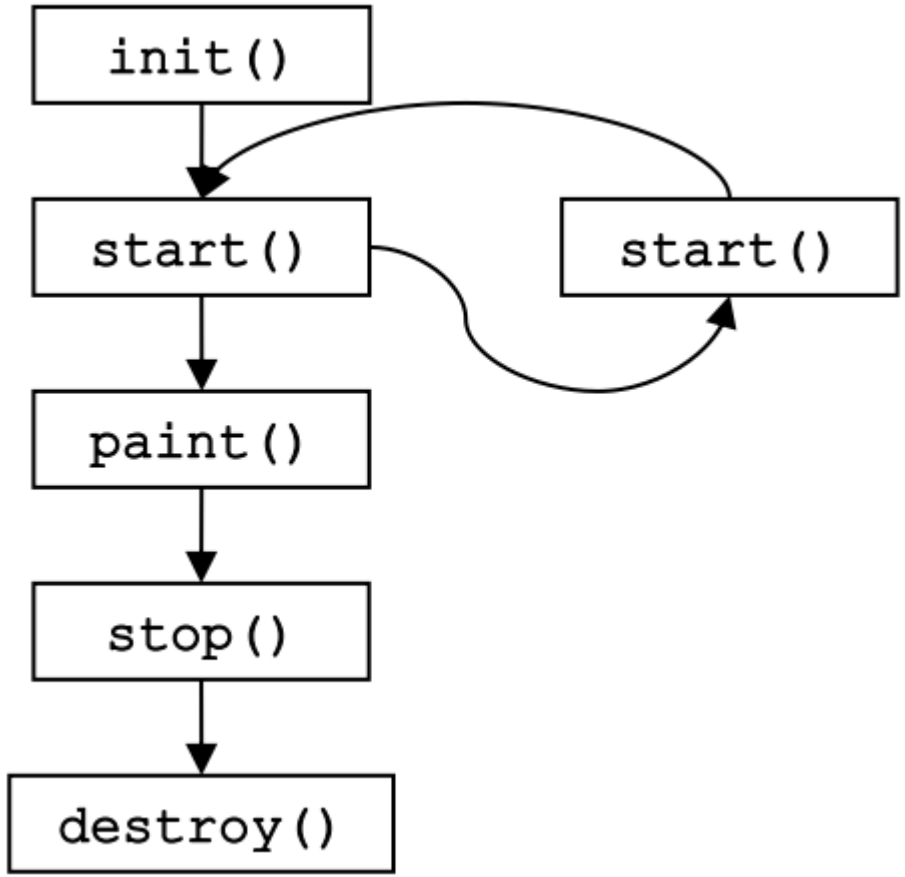

Data Types
Primitive Types
Reference Types
Boolean
Numeric
Class Types
Interface Types
Array Types
Character
Integral
Floating
boolean
char
Integer
double
float
byte
short
int
long



bookes\dataStru\java2021-2022\scrap

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


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


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

colors and filled shapes

Chapter 6.13 p227

public Color(int r, int g, int b)

Graphics methods fillRect and fillOval draw filled rectangles and ovals.