جَامعة المَنارة
MANARA UNIVERSITY

كلية الهندسة المعلوماتية

برمجة 3

Java Programming

ا. د. علي عمران سليمان

محاضرات الأسبوع الثالث

الفصل الصيفي 2023-2024

content

- Secure Random-Number Generation.
- Enumerated Types.
- Enumerated Types – Methods.
- Enhanced for Statement.
- Garbage Collection and Method finalize.
- Colors and Filled Shapes—Drawing a bull's-eye and random graphics.
- Drawing Arcs—Drawing spirals with arcs.

References

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

Secure Random-Number Generation 1

- **يحتاج المبرمج للأعداد العشوائية بكثره عند تصميم برامج** Lottery، simulation، games ... etc.
- **يمكن إنشاء كائنات عشوائية من أصناف الحزمة** (package java.security) ومن أنماط مختلفة:

(boolean, byte, float, double, int, long and Gaussian values).

(تنعيم Gaussian هو مرشح تمرير منخفض يمكن استخدامه لإزالة الضوضاء عالية التردد من صورة تتمتع وظيفة الغوسية بتردد قطع طبيعي يتم بعدها تخفيفها بسرعة. هذا يعني أنه تتم إزالة الضوضاء عالية التردد مع الحفاظ على الهيكل الكلي للصورة).

- **استخدمت java في نسخها القديمة لإنتاج القيم العشوائية، الصنف** "random" **إلا أن هذه القيم كانت هدف بعض المبرمجين (المغرضين) لتوقعها وتمكنوا من ذلك مما افقدها غايتها.**
- **الصنف** SecureRandom **ينتج قيم عشوائية صامدة حتى الآن من الاختراق ومتوفرفي العديد من اللغات الأخرى.**
- **يتطلب الأمر استدعاء الصنف واشتقاق كائن منه وفق التالي:**

```
import java.security.SecureRandom;
SecureRandom randomNumbers = new SecureRandom();
```

- **سنناقش الان الأعداد الصحيحة فقط وللمزيد عن الأنواع الاخرى موجود في الرابط التالي:**

see docs.oracle.com/javase/7/docs/ api/java/security/SecureRandom.html

- nextInt() تنتج قيم عشوائية من نطاق الأعداد الصحيحة;**int randomValue =randomNumbers.nextInt()**

Secure Random-Number Generation 2

- **يمكن التحكم بنطاق الأعداد العشوائية المنتجة من خلال إرسال قيمة للطريقة ()nextInt مابين القوسين.**
- **مثلاً للتعبير عن الشعار بـ 0 والنقش بـ 1 في قطعة النقد نرسل له (2) ولمحاكات قطعة النرد نرسل له (6) ولتمثيل الاتجاهات الأربع لزوم تحديد الاتجاهان للعبه نرسل (4) الناتج قيم تبدأ من الصفر وإلى العدد السابق للعدد للمرسل.**

```
int randomValue = randomNumbers.nextInt(2);//returns 0 or 1.
int randomValue = randomNumbers.nextInt(6);//returns 0 or 1 or 2 or 3 or 4 or 5.
```

- **يتطلب الأمر بعض الازاحات للوصول لمحاكات حبة النرد وفق التالي:**

```
int face = 1 + randomNumbers.nextInt(6);
```

- **لها الصيغة العامة:**

```
int number = shiftingValue + randomNumbers.nextInt(scalingFactor);
```

- **حيث** shiftingValue **تعبرعن الرقم الأول في النطاق المطلوب من الأعداد الصحيحة المتتالية.**
- **يحدد** ScalingFactor**عامل التحجيم عدد الأرقام الموجودة في النطاق.**
- اختيار أعداد صحيحة عشوائيًا من مجموعات قيم بخلاف نطاقات الأعداد الصحيحة المتتالية. مثلاً ، للحصول على القيم العشوائية 2 و 5 و 8 و 11 و 14 ، نستخدم. ;(int number = 2 + 3 * randomNumbers.nextInt(5)

```
int number = shiftingValue + differenceBetweenValues *randomNumbers.nextInt(scalingFactor);
```

Enumerated Type s1

- عند تعريف متغير من نوع معين سيتقبل هذا النوع في المكان المحجوز له int x; ستخزن القيم الصحيحة فقط.
- لنعمم ذلك أكثر لنفترض اننا نملك مجموعة خاصة من الصفات ونرغب بتخزينها وإجبار المستخدم بها مثل تخزين أيام الأسبوع مثلاً وبالتالي عند ما يكتب اية مستخدم للصنف قيمة خارج أيام الأسبوع سيعترض المطابق.

الصيغة العامة **Syntax: enum typeName { one or more enum constants }**

- **Enum Type** النوع التعداد الأساسي: يحدد نوعاً ومجموعة من الثوابت الممثلة كمعرفات فريدة يمكن أن تستخدم لهذا النوع (كتعريف لنوع جديد وتحدد مجموعة القيم لهذا النوع الجديد ومن يختار هذا النوع ملزم بقيمه).

مثال : **limited to list in the combo box** عند أنتقاء خيار من خيارات صندوق الحوار.

Enum Day { SUNDAT, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY SATURDAY}

- كل من يُعرف صفه من النوع **Day** هو ملزم بهذه القيم.
- من أجل الاستثمار لابد من متغير من هذا النمط : مثل **Day WorkDay ;** أي **WorkDay** من نمط **Day**
- وهنا عند كتابة Day WorkDay = Day.WEDNESDAY; عند كتابة النقطة سنحصل على الثوابت المعرفه ليتم الاختيار منها.

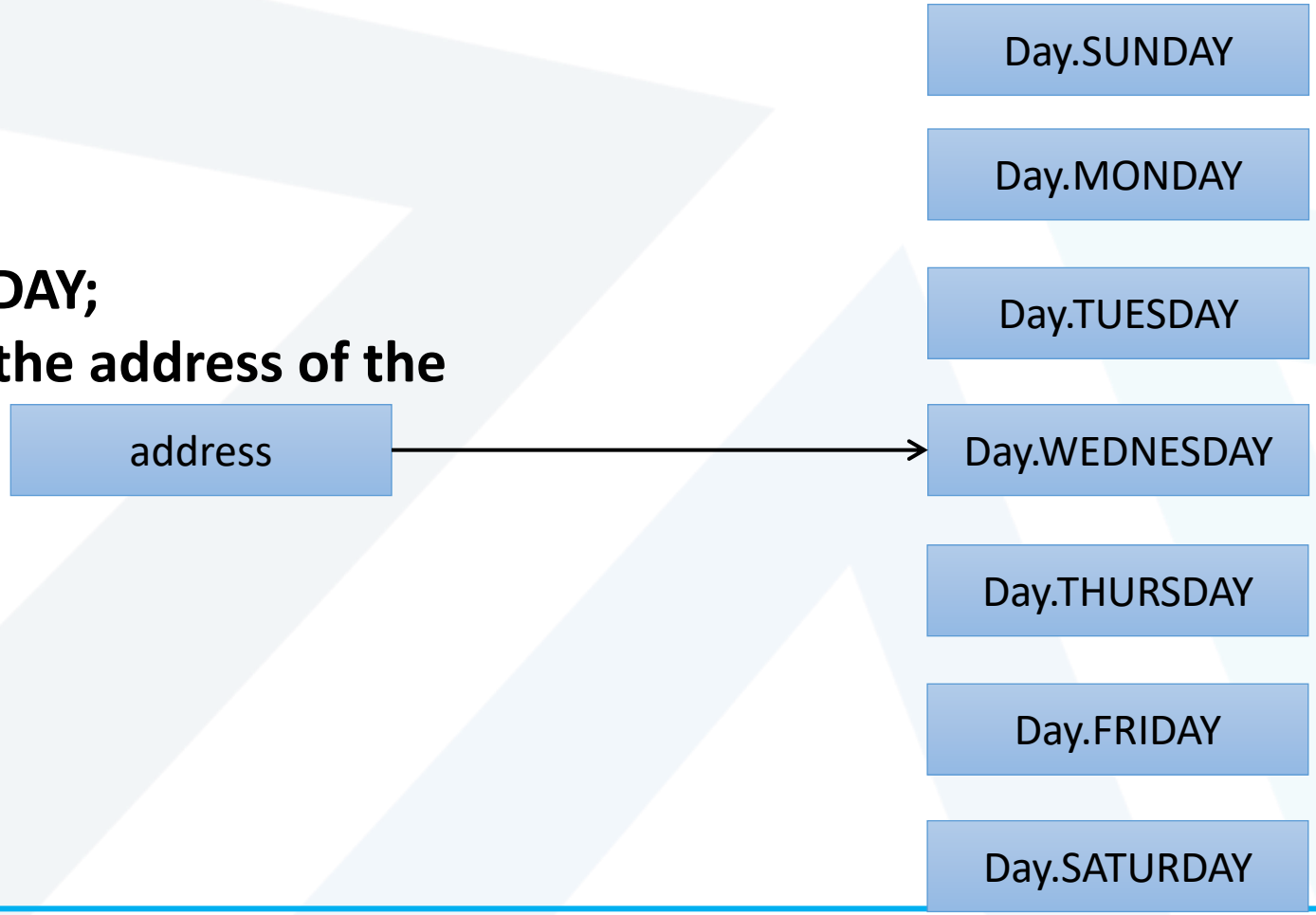
- **Enum** هو صنف خاص وأن **WorkDay** ليس سوى مرجعاً للكائن **Day.WEDNESDAY;** .

Enumerated Type s1

إن Enum هوصنف خاص.

Each are objects of type Day's specializer Class.

Day WorkDay =Day.WEDNESDAY;
The workDay Variable holds the address of the Day.WEDNESDAY



Enumerated Type s2

- الأنواع التعداد هي أنواع مرجعية.

- يعلن كل تصريح تعداد عن فئة تعداد مع القيود التالية:
  - باني Enum هو نهائي final ضمناً.
  - باني Enum هو static ضمناً.
  - أي محاولة لإنشاء كائن من نوع التعداد باستخدام عامل new ينتج عنه خطأ في المطابقة.

- ثوابت Enum يمكن أن تستخدم في أماكن استخدام الثوابت مثلاً كما في حالات case ضمن بنية switch ودليل ضبط العبارات المحسنة.

- كيفية التصريح عن متغيرات المثيل والباني والطرق في النوع التعداد Enum.
  - سيتم عرض المتغيرات المرجعية، والباني وطرق في نوع التعداد.

- يمتلك النوع التعدادي عدد من المناهج نذكر منها.

Enumerated Types - Methods 1

- toString **– returns name of calling constant.**
- ordinal **– returns the zero-based position of the constant in the enum. For example the ordinal for** Day.THURSDAY **is 4.**
- equals **– accepts an object as an argument and returns true if the argument is** equal **to the calling enum constant .**
- compareTo **- accepts an object as an argument and returns** an integer value based on the given cases,

•It returns **0** when this Enum object is equal to or same as the given Enum object.
•It returns **positive value** when this Enum object is **greater** than the given Enum object.
•It returns **negative value** when this Enum object is **less** than the given Enum object.

Enhanced for Statement

```
enum Course {Programming1,Database,Programming2,
Datastructure1};
enum Semester {Fall ,Winter, Summer};
public class EnumRegisterForm
{
 String stuName;
 Course  crs;
 Semester sem;
 public  EnumRegisterForm()
 {stuName = "Adam";
 crs=Course.Database;
 sem=Semester.Summer;
   }
}
```

Enhanced for Statement

```
public class EnumRegisterFormTest {
public static void main(String[] args)
{ Course cor1=Course.Database;
   Course cor2=Course.Programming1;
   System.out.println(cor1.toString());
   System.out.println(cor1.ordinal());
   System.out.println(cor1.compareTo(cor2));
   System.out.println(cor1.equals(cor1));
  } //end main
} // end Class
```

Database
1
1
true

Enumerated Type Exa.1

RegisterForm
- StdName: String - StdGender : Gender - LectureTime : Time - CourseName: Course - CrsSemester: Semester

- لنفر اننا نرغب بتطوير البرنامج السابق ليمثل نموذج لتسجيل الطلبة في المقررات.RegisterForm بتضمن **enum**
  - يختار اسم الطالب.
  - يختار الجنس.
  - يختار وقت المحاضرة LectureTime.
  - يختار المقرر Course .
  - يختار Semester

  جميعها من القوائم المتاحه.

- كتابة باني بدون بارامترات وباني مع كل البارامترات ويختبر الطرق التالية على الأقل:

  toString **()**
  - Ordinal()
  - equals()
  - compareTo()

Enhanced for Statement

- **عبارة for المعززة:** تستخدم للوصول لعناصر مصفوفة أو مجموعة دون استخدام عداد.

The syntax of an enhanced for statement is:

for (parameter : arrayName) statement

- parameter **هو متغير من نوع القيم في المصفوفة.**
- arrayName **اسم المصفوفة التي سيتم التكرار من خلالها وهنا لايمكن تحرير العناصر ولا معرفة الفهرس للعنصر.**

```
for (int i = 0; i < myArray.length; i++) {     System.out.println(myArray[i]);}
```

Can be written as:

```
for (int myValue : myArray) {     System.out.println(myValue);  }
```

Enumerated Type Exa.2.1

- إعلان التعداد Enum في المثال التالي يحتوي على جزأين - ثوابت Enum والأعضاء الأخرى من نوع التعداد.
- الجزء الأول يتضمن 6 ثوابت ، كل منها يمكن أن تتبع ببارامترات ويمكن أن تمرر من الباني.
- يمكن للباني أن يسند قيم وأن يحمل تحميلاً زائداً.

الباني في مثالنا يمكن أن يحتاج إلى متغيرين من النمط String لتجهيز كل ثابت من الثوابت الست السابقة.

- الجزء الثاني يتضمن تعريف أعضاء من Enum وهما متغيرين والباني وطريقتين.

- المتغيرين هما title, copyrightYear وكل ثابت من Enum Book يملك هذين المتغيرين.

- الباني يأخذ متغيربن من النوع String الأول يسند إلى title والثاني إلى copyrightYear .

- الطريقتين سيعيدان return the book title and copyright year .
- سيتم الاخبار من خلال استخدام book كدليل لحلقة for المحسنة وسيتم استخدام Book.values() لمعرفة عدد الثوابت وكذلك EnumSet.*range*(*Book.JHTP*, *Book.CPPHTP*) لتحديد مجال منها.

Enumerated Type Exa.2.2

```
package firstPro;

// Fig. 8.10: Book.java
// Declaring an enum type with a constructor and explicit instance fields
// and accessors for these fields

public enum Book {

// declare constants of enum type

JHTP("Java How to Program", "2015"),
CHTP("C How to Program", "2013"),
IW3HTP("Internet & World Wide Web How to Program", "2012"),
CPPHTP("C++ How to Program", "2014"),
VBHTP("Visual Basic How to Program", "2014"),
CSHARPHTP("Visual C# How to Program", "2014");
```

Enumerated Type Exa.2.3

```
// instance fields
 private final String title; // book title
 private final String copyrightYear; // copyright year

 // enum constructor
 Book(String title, String copyrightYear)
 {  this.title = title; this.copyrightYear = copyrightYear; }

 // accessor for field title
 public String getTitle() { return title; }

 // accessor for field copyrightYear
 public String getCopyrightYear() { return copyrightYear; }
 } // end enum Book
```

Enumerated Type Exa.2.4

```
package firstPro;
     // Fig. 8.11: EnumTest.java
      // Testing enum type Book.
 import java.util.EnumSet;
public class EnumTest
 {
 public static void main(String[] args)
 { System.out.println("All books:");
  // print all books in enum Book enhanced for
  for (Book book : Book.values())
 System.out.printf("%-10s%-45s%s%n", book,book.getTitle(), book.getCopyrightYear() );

 System.out.printf("%nDisplay a range of enum constants:%n");
  // print first four books
 for (Book book : EnumSet.range(Book.JHTP, Book.CPPHTP))
  System.out.printf("%-10s%-45s%s%n", book, book.getTitle(), book.getCopyrightYear());
        } // end main method
  } // end class EnumTest
```

Enumerated Type Exa 2.5

```
All books:
JHTP       Java How to Program                           2015
CHTP       C How to Program                              2013
IW3HTP     Internet & World Wide Web How to Program      2012
CPPHTP     C++ How to Program                            2014
VBHTP      Visual Basic How to Program                   2014
CSHARPHTP  Visual C# How to Program                      2014

Display a range of enum constants:
JHTP       Java How to Program                           2015
CHTP       C How to Program                              2013
IW3HTP     Internet & World Wide Web How to Program      2012
CPPHTP     C++ How to Program                            2014
```

Garbage Collection and Method finalize

- تشغل الكائنات موارد النظام ، الذاكرة مثلاً .
- لا بد من طريقة منضبطة لإعادة الموارد إلى النظام عندما تنتهي الحاجة للحجز، كي لاتحدث "تسربات في الموارد resource leaks" .
- تقوم **JVM** بجمع البيانات المهملة "الكائنات" تلقائيًا، عندما لا تملك من يؤشر عليها.
- يحدث التجميع عادةً من قبل **JVM** بتنفيذ **Garbage Collection**. والمشكلة يحدث الأمر بشكل غير منضبط وقد يتأخر حتى انتهاء البرنامج.
- أنما يحصل وهو على خلاف من اللغات الأخرى مثل **C** و **C++** التي لن يحدث تلقائياً.
- إضافةً لذلك قد يفتح التطبيق ملفًا على القرص لتعديل محتوياته ولا يقوم بغلقه، فيجب أن يتم إغلاقه قبل أن يتمكن أي تطبيق آخر من استخدام الملف.
- تحتوي كل فئة في **Java** على طرق صنف **Object (package java.lang)**، واحدة منها هي الطريقة finalize ونادرًا ما تستخدم هذه الطريقه لأنها يمكن أن تسبب مشاكل في الأداء وهناك بعض الشكوك حول ما إذا كان سيتم استدعائها أم لا قبل إنهاء البرنامج .
- بما أن الطريقة جزء من كل صنف تناقش هنا من أجل التسهيل لفهمها والمساهمة في تطويرها لاحقاً.
- تعمل الكائنات القابلة للإغلاق تلقائيًا على تقليل احتمالية تسرب الموارد عند استخدامها مع بيان "تحرير الموارد". يتم إغلاق كائن قادر على AutoClosable بمجرد انتهاء بيان **try-with-resources** من استخدام الكائن.

جَامعة المَنارة
MANARA UNIVERSITY

كلية الهندسة المعلوماتية

برمجة 3
Java Programming

ا. د. علي عمران سليمان

محاضرة 2 الأسبوع الثالث
GUI
الفصل الصيفي 2022-2023

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

تذكرة

كل برنامج رسومي يستخدم نافذة إطار frame window أو أكثر ولكل نافذة اطار شريط عنوان titel bar وحدود border لكي نظهر 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 ونرغب بعرضه إلى الإطار `appli.add( panel );`

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

```
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 الاغلاق 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

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 وارث له.

- 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 hidden ثم الكشف عنها بواسطة نافذة أخرى على الشاشة، وعندما يتم تغيير حجم النافذة التي تظهر فيها.

Creating Simple Drawings

5.25 GUI &Graphics

- ✓ رسم الأقواس في Java يشبه رسم الأشكال البيضاوية - القوس هو مجرد جزء من الشكل البيضاوي.
- ✓ منهج الرسومات `fillArc` من الصنف `Graphics` يرسم قوسًا ممتلئًا.
- ✓ يتطلب المنهج `fillArc` ستة معاملات.
  - الأربعة الأولى تمثل المستطيل المحيط الذي سيرسم فيه القوس.
  - المعامل الخامس هو زاوية البداية ، مع عدم وجود معامل تشير إلى الصفر على محور x.
  - والسادس يحدد مقدار المسح ، أو مقدار القوس المراد تغطيته.
  - يتم قياس زاوية البداية والمسح بالدرجات.
  - المسح الموجب يرسم القوس بعكس اتجاه عقارب الساعة والسالب معها.
- ✓ تتطلب طريقة `drawArc` نفس المعلمات مثل `fillArc`، ولكنها ترسم حافة القوس بدلاً من تعبئتها.
- ✓ منهج setBackground يغير لون الخلفية لمكون واجهة المستخدم الرسومية.

Drawing Rainbow 1

7.25 GUI &Graphics

```
1  // Fig. 7.25: DrawRainbow.java
2  // Demonstrates using colors in an array.
3  import java.awt.Color;
4  import java.awt.Graphics;
5  import javax.swing.JPanel;
6
7  public class DrawRainbow extends JPanel
8  {
9     // define indigo and violet
10    private final static Color VIOLET = new Color( 128, 0, 128 );
11    private final static Color INDIGO = new Color( 75, 0, 130 );
12
13    // colors to use in the rainbow, starting from the innermost
14    // The two white entries result in an empty arc in the center
15    private Color[] colors =
16       { Color.WHITE, Color.WHITE, VIOLET, INDIGO, Color.BLUE,
17         Color.GREEN, Color.YELLOW, Color.ORANGE, Color.RED };
18
19    // constructor
20    public DrawRainbow()
21    {
22       setBackground( Color.WHITE ); // set the background to white
23    } // end DrawRainbow constructor
24
```

Fig. 7.25 | Drawing a rainbow using arcs and an array of colors. (Part 1 of 2.)

Drawing Rainbow 2


```
25      // draws a rainbow using concentric arcs
26      public void paintComponent( Graphics g )
27      {
28         super.paintComponent( g );
29
30         int radius = 20; // radius of an arc
31
32         // draw the rainbow near the bottom-center
33         int centerX = getWidth() / 2;
34         int centerY = getHeight() - 10;
35
36         // draws filled arcs starting with the outermost
37         for ( int counter = colors.length; counter > 0; counter-- )
38         {
39            // set the color for the current arc
40            g.setColor( colors[ counter - 1 ] );
41
42            // fill the arc from 0 to 180 degrees
43            g.fillArc( centerX - counter * radius,
44               centerY - counter * radius,
45               counter * radius * 2, counter * radius * 2, 0, 180 );
46         } // end for
47      } // end method paintComponent
48   } // end class DrawRainbow
```

Fig. 7.25 | Drawing a rainbow using arcs and an array of colors. (Part 2 of 2.)

Drawing Rainbow 3


```
1  // Fig. 7.26: DrawRainbowTest.java
2  // Test application to display a rainbow.
3  import javax.swing.JFrame;
4
5  public class DrawRainbowTest
6  {
7     public static void main( String[] args )
8     {
9        DrawRainbow panel = new DrawRainbow();
10       JFrame application = new JFrame();
11
12       application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
13       application.add( panel );
14       application.setSize( 400, 250 );
15       application.setVisible( true );
16    } // end main
17 } // end class DrawRainbowTest
```

Fig. 7.26 | Creating `JFrame` to display a rainbow. (Part 1 of 2.)

Drawing Rainbow 4

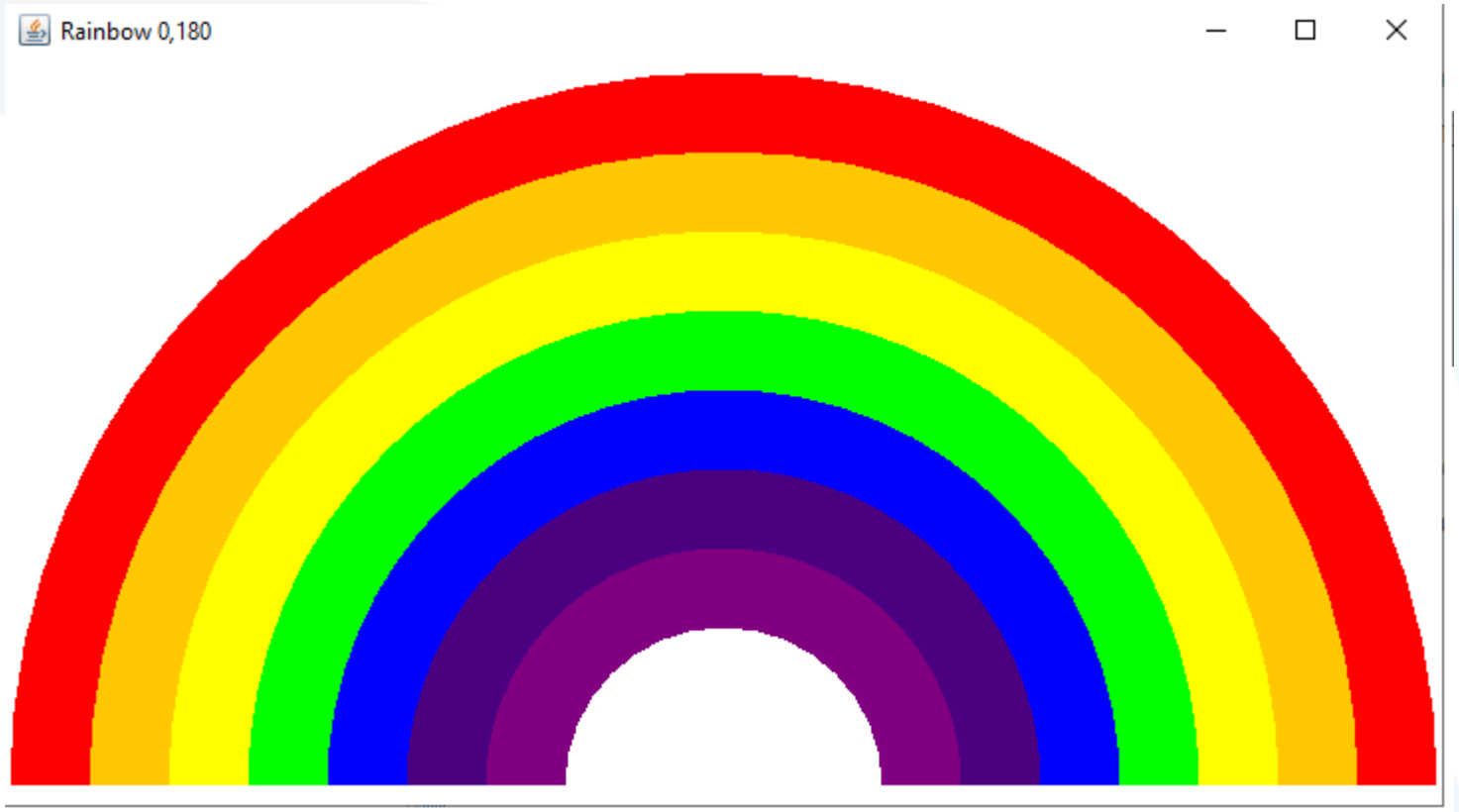


Fig. 7.26 | Creating `JFrame` to display a rainbow. (Part 2 of 2.)

Create a special color

اتعريف لونين جديدين

```
final Color VIOLET = new Color( 128, 0, 128 );
final Color INDIGO = new Color( 75, 0, 130 );
```

- أن ألوان قوس قزح هي **الأحمر والبرتقالي والأصفر والأخضر والأزرق والنيلي والبنفسجي.**
- تحتوي Java فقط على ثوابت سابقة التعريف للألوان الخمسة الأولى.
- الاسطر تهيئة مصفوفة مع ألوان قوس قزح، بدءاً من الأقواس الأعمق أولاً يبدأ المصفوفة بعنوين Color.WHITE، والتي، كما سترى قريبًا، هي لرسم الأقواس الفارغة في مركز قوس قزح.

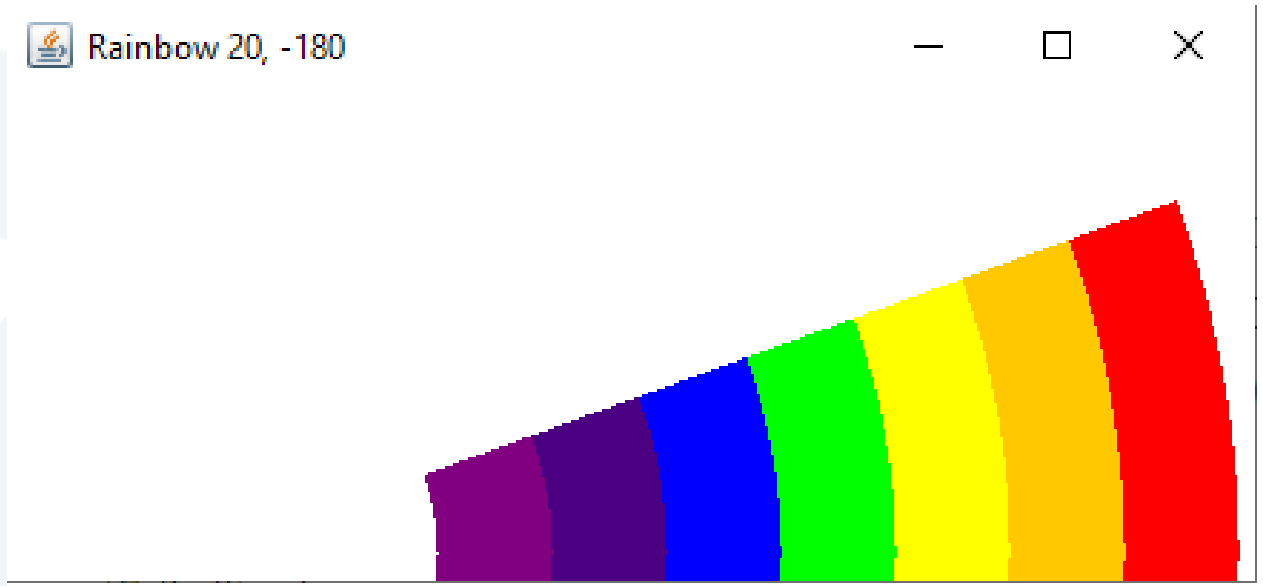
```
private Color colors[] = { Color.WHITE, Color.WHITE, VIOLET, INDIGO,
Color.BLUE, Color.GREEN, Color.YELLOW, Color.ORANGE, Color.RED };
```

- يمكن تهيئة متغيرات الحالة عند إعلانها. يحتوي المُنشئ على عبارة واحدة تستدعي طريقة setBackground (الذي تم توارثه من الفئة JPanel) بالمتغير Color.WHITE. تأخذ مجموعة setBackground وسيطة لون واحدة وتجعل الخلفية للمكون من ذلك اللون.
- الطريقة paintComponent تعلن عن متغير يعبر عن نصف قطر الدائرة التي سترسم، أي يحدد سمك كل قوس. تحدد المتغيرات المحلية centerX وcenterY موقع نقطة الوسط على قاعدة قوس قزح.(منتصف X وعلى بعد 10 من كامل h).
- تستخدم الحلقة counter للتحكم والعد العكسي من نهاية المصفوفة، مع رسم أكبر قوس أولاً ووضع كل قوس أصغر متتابع أعلى القوس السابق. * يعين السطر 40 اللون لرسم القوس الحالي من المصفوفة. السبب في أن لدينا مداخل Color.WHITEفي بداية المصفوفة هي إنشاء قوس فارغ في المركز. خلاف ذلك، فإن مركز قوس قزح يكون مجرد دائرة نصف دائرة البنفسجي الصلبة.

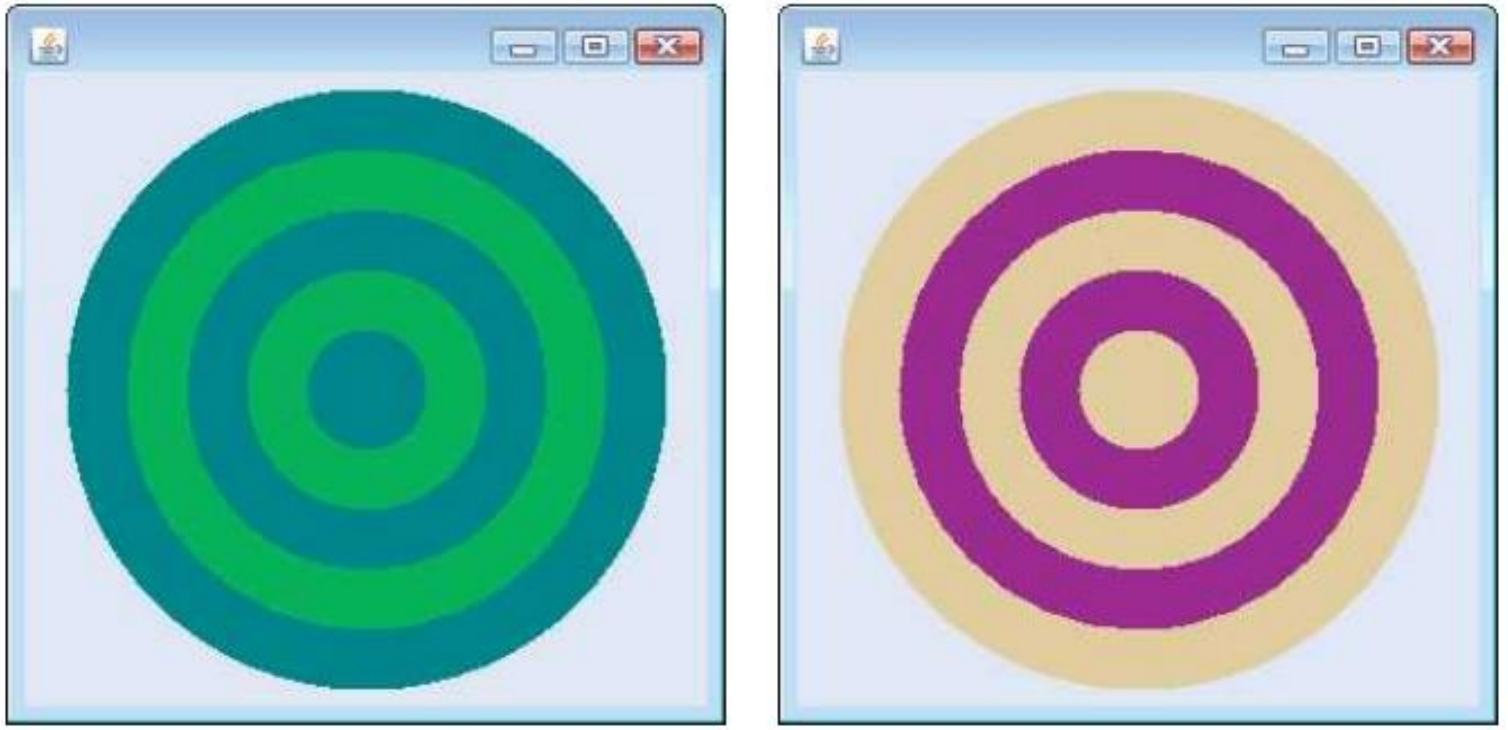
[ملاحظة: يمكنك تغيير ألوان فردية وعدد الإدخالات في المصفوفة لإنشاء تصميمات جديدة.]

Rainbow 20, 18

Rainbow 20, -18



A bulls-eye with two alternating, random colors.



ليكن المطلوب رسم الشكل السابق

Draw many shapes 1

جَامعة المَنارة
MANARA UNIVERSITY

DrawRainbow21 1

```
// Fig. 5.26: Shapes2.java
 // Demonstrates drawing different Shapes2.
 import java.awt.Color;
import java.awt.Graphics;
 import javax.swing.JPanel;
 import java.security.SecureRandom;
 //import java.util.Random; // program uses class Random
  public class DrawRainbow21 extends JPanel
  {
   // Define indigo and violet
   final Color VIOLET = new Color( 128, 0, 128 );
   final Color INDIGO = new Color( 75, 0, 130 );
   int x1,y1,w,h;
   // colors to use in the rainbow, starting from the innermost
   // The two white entries result in an empty arc in the center
   private Color colors[] =
      { Color.WHITE, Color.WHITE, VIOLET, INDIGO, Color.BLUE,
        Color.GREEN, Color.YELLOW, Color.ORANGE, Color.RED };
```

Draw many shapes 2

DrawRainbow21 2

```
// constructor
   private int choice; // user's choice of which shape to draw

// constructor sets the user's choice

public DrawRainbow21(int userChoice)
   {choice = userChoice;
      setBackground( Color.WHITE ); // set the background to white
   } // end DrawRainbow21 constructor

   // draws a rainbow using concentric circles
   public void paintComponent( Graphics g )
   {      super.paintComponent( g );

// draws a cascade of Shapes21 starting from the top-left corner///
      SecureRandom randomNumbers = new SecureRandom();
// SecureRandom number generator
```

Draw many shapes 3

DrawRainbow21 3

```
// pick the shape based on the user's choice
switch ( choice )
{case 1: // draw DrawRainbow2
  int radius = 40;// radius of an arch
     // draw the rainbow near the bottom-center
      int centerX = getWidth() / 2;
      int centerY = getHeight() - 10;

      // draws filled arcs starting with the outermost
      for ( int counter = colors.length; counter > 0; counter-- )
      {  // set the color for the current arc
         g.setColor( colors[ counter - 1 ] );

         // fill the arc from 0 to 180 degrees
         g.fillArc( centerX - counter * radius,
            centerY - counter * radius,
            counter * radius * 2, counter * radius * 2, 45, 90 );
```

Draw many shapes 4

DrawRainbow21 4

```
 // set the color for the current arc
          /*g.setColor( colors[ counter - 1 ] );
          g.fillOval( 10 + counter * 10, 10 + counter * 10,
      240 - counter * 20, 240 - counter * 20 );}*/
          } break;
case 2: // set the color for the current arc
  for ( int counter = 8; counter > 0; counter-- )
  {  g.setColor( colors[ counter - 1 ] );
       g.fillOval(130-counter * 10,130-counter *10,counter * 20,counter * 20 );
  //g.fillOval(30+counter *10,30+counter *10,200-counter *20,200-counter *20);
  /*g.fillOval(10+counter*10,10+counter*10,240-counter*20,240-counter*20); */
  x1=130 - counter *10; y1=130 - counter * 10; w= counter * 20; h=counter * 20;
            System.out.println("x1=" +x1+ " y1= "+y1+ " w= "+w+" h= "+h);
     }  break;
case 3: // stores each random integer generated shapesType,shapesColor,(x1,y1)
begin ,widnth ,height
```

Draw many shapes 5

DrawRainbow21 5

```
for ( int counter = 10; counter > 0; counter-- )
   {   int shapesType, shapesColor, x1, y1, x2, y2;
    shapesType = 1 + randomNumbers.nextInt( 2 );
    x1 = 1 + randomNumbers.nextInt( 200 ); y1 = 1 + randomNumbers.nextInt( 200 );
    w = 1 + randomNumbers.nextInt( 188 );  h = 1 + randomNumbers.nextInt( 188);
          System.out.println("x1=" +x1+ " y1= "+y1+ " w= "+w+" h= "+h);
  //if((x1+x2)>400) x2=380;  if((y1+y2)>400) y2=370;
          if(shapesType==1)
 {  // draw rectangles   and set the color for the current rectangles
  shapesColor = 2 + randomNumbers.nextInt( 6 );
   g.setColor( colors[ shapesColor ] ); g.fillRect( x1, y1, w, h ); }
 else { // draw ovals and set the color for the current ovals
 shapesColor = 2 + randomNumbers.nextInt( 6 );
 g.setColor( colors[ shapesColor ] );  g.fillOval( x1, y1, w, h );}  //break;
         } } // end for
}// end method paintComponent
   }// end class DrawRainbow21
```

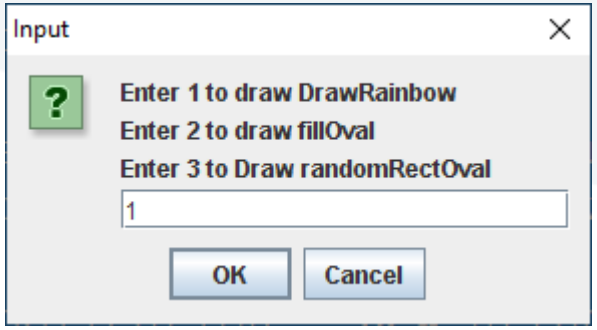
Draw many shapes 6

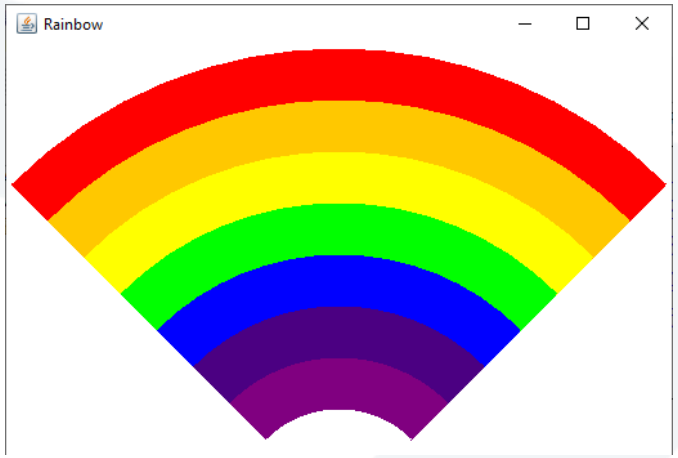
DrawRainbow21Test

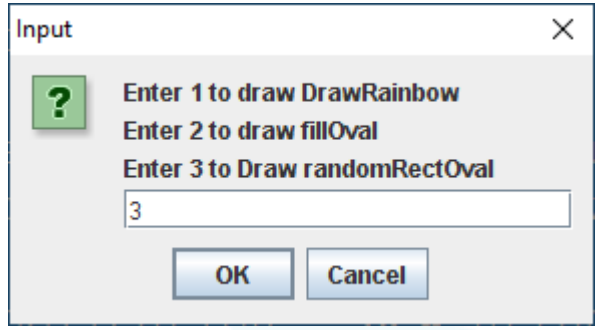
```
public class DrawRainbow21Test
 {  public static void main( String[] args )
 {  // obtain user's choice
 String input = JOptionPane.showInputDialog(
 "Enter 1 to draw DrawRainbow \n" +
 "Enter 2 to draw fillOval\n"+
 "Enter 3 to Draw randomRectOval\n");
 int choice = Integer.parseInt( input ); // convert input to int
 // create the panel with the user's input
 DrawRainbow21 panel = new DrawRainbow21( choice );
 JFrame application = new JFrame(); // creates a new JFrame
 application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
 application.setSize( 400, 400 ); // set the desired size
 application.add( panel ); // add the panel to the frame
 application.setVisible( true ); // show the frame
 } // end main
 } // end class DrawRainbow1Test
```

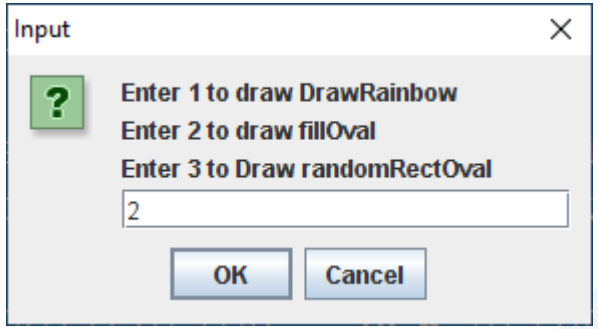
Drawing Ranom Rect and Oval

Draw many shapes 7

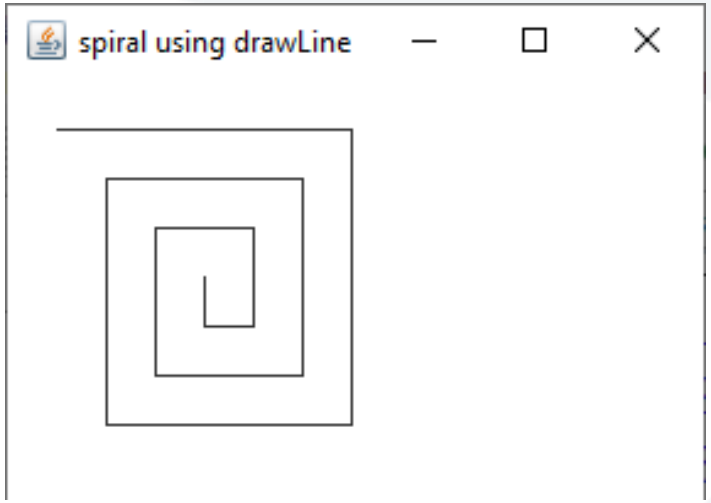


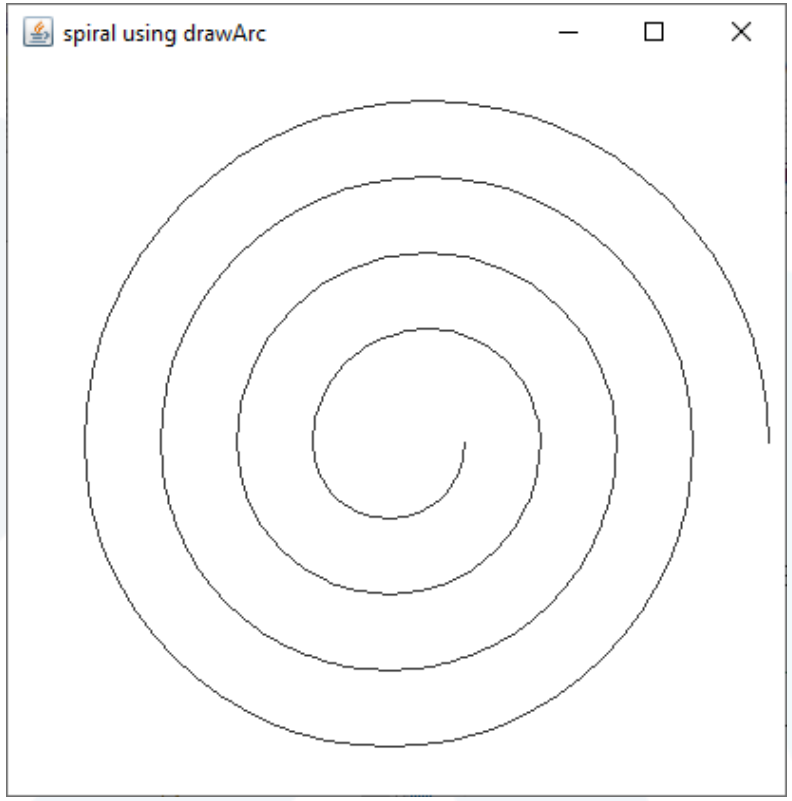






Drawing spiral





ليكن المطلوب رسم الشكل السابق

Creating Simple Drawings

5.25 GUI &Graphics

- ✓ يخزن المثال التالي معلومات حول الأشكال المعروضة حتى نتمكن من إعادة إنتاجها في كل مرة يستدعي فيها النظام `paintComponent`.
- ✓ سننشئ فئات شكل "ذكية" يمكنها رسم نفسها باستخدام كائن رسومات من نمط `Graphics`.
- ✓ يوضح الشكل 8.21 صنف `MyLine`، اللذي يحتوي على كل هذه القدرات.
- ✓ يتكرر المنهج `paintComponent` في الصنف `DrawPanel` من خلال رسم مجموعة من الكائنات `MyLine`.
  - في كل استدعاء للكائن من الصنف `MyLine` يتم تكرار الاستدعاء لمنهج الرسم `draw` ويمرره كائن الرسومات `Graphics` من أجل الرسم على اللوحة.

MyLine الصنف

```
import java.awt.Color;
import java.awt.Graphics;
public class MyLine
{  private int x1;  // x-coordinate of first endpoint
   private int y1;  // y-coordinate of first endpoint
   private int x2;  // x-coordinate of second endpoint
   private int y2;  // y-coordinate of second endpoint
   private Color myColor; // color of this shape
   // constructor with input values
   public MyLine( int x1, int y1, int x2, int y2, Color color )
   {  this.x1 = x1; // set x-coordinate of first endpoint
      this.y1 = y1; // set y-coordinate of first endpoint
      this.x2 = x2; // set x-coordinate of second endpoint
      this.y2 = y2; // set y-coordinate of second endpoint
      myColor = color; // set the color
   }          // end MyLine constructor       // Actually draws the line
   public void draw( Graphics g )
   {  g.setColor( myColor );       g.drawLine( x1, y1, x2, y2 );
   }          // end method draw
}             // end class MyLine
```

DrawPanel2.java الصنف

uses class MyLine to draw random lines 1

```
// Fig. 8.19: DrawPanel.java
// Program that uses class MyLine
// to draw random lines.
import java.awt.Color;
import java.awt.Graphics;
import java.util.Random;
import javax.swing.JPanel;
public class DrawPanel2 extends JPanel
{  private Random randomNumbers = new Random();
   private MyLine[] lines; // array on lines
   // constructor, creates a panel with random shapes
   public DrawPanel2()
   { setBackground( Color.WHITE );
     lines = new MyLine[ 5 + randomNumbers.nextInt( 5 ) ];  // create lines
      for ( int count = 0; count < lines.length; count++ )
      { // generate random coordinates
         int x1 = randomNumbers.nextInt( 300 );
         int y1 = randomNumbers.nextInt( 300 );
         int x2 = randomNumbers.nextInt( 300 );
         int y2 = randomNumbers.nextInt( 300 );
```

DrawPanel2.java الصنف

uses class MyLine to draw random lines 2

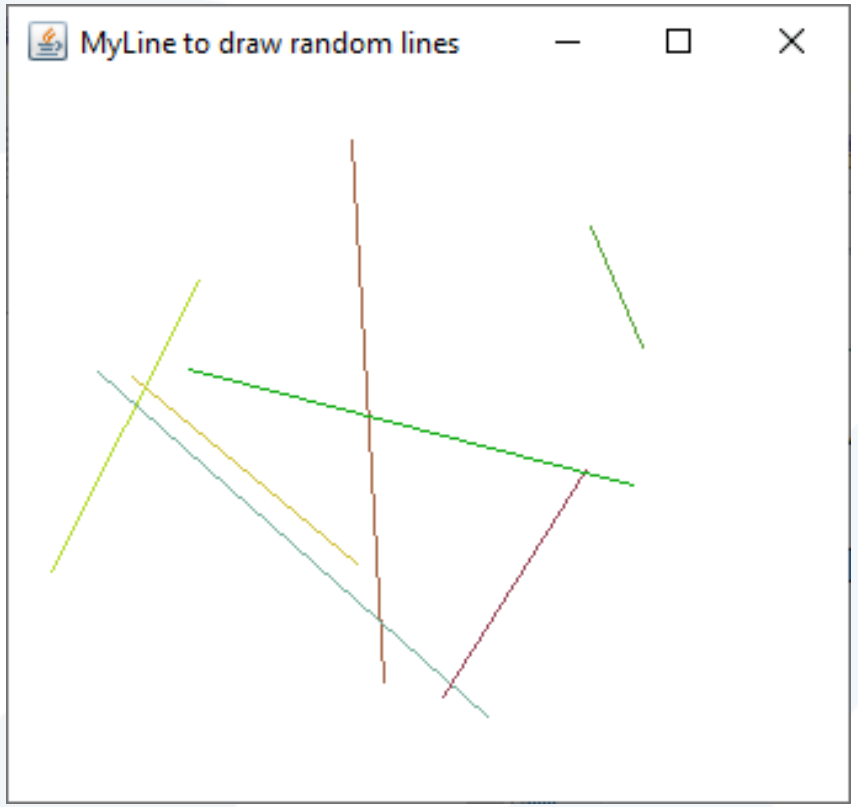
```
            // generate a random color
            Color color = new Color( randomNumbers.nextInt( 256 ),
               randomNumbers.nextInt( 256 ), randomNumbers.nextInt( 256 ) );
            // add the line to the list of lines to be displayed
            lines[ count ] = new MyLine( x1, y1, x2, y2, color );
         }        // end for
      }           // end DrawPanel constructor
      // for each shape array, draw the individual shapes
      public void paintComponent( Graphics g )
      {
         super.paintComponent( g );
            // draw the lines
         for ( MyLine line : lines )
            line.draw( g );
      }           // end method paintComponent
}              // end class DrawPanel2
```

DrawTest2.java الصنف

Creating a JFrame to display a DrawPanel

```
// Fig. 8.20: TestDraw.java
// Creating a JFrame to display a DrawPanel.
import javax.swing.JFrame;
public class DrawTest2
{
   public static void main( String[] args )
   {
      DrawPanel2 panel = new DrawPanel2();
      JFrame application = new JFrame("MyLine to draw random lines");
      application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
      application.add( panel );
      application.setSize( 300, 300 );
      application.setVisible( true );
   } // end main
} // end class TestDraw
```

uses class MyLine to draw random lines 3



رسم مستقسمات عشوائية اللون والطول والعدد لايقل عن 5 ولايزبد عن 10

9.12 GUI &Graphics

Demonstrates the use of labels

- Labels هي طريقة مناسبة لتحديد مكونات واجهة المستخدم الرسومية على الشاشة وإبقاء المستخدم على اطلاع بالحالة الحالية للبرنامج.
- يمكن لـ `JLabel` من الحزمة `javax.swing` عرض نص أو صورة أو كليهما.
- يوضح المثال في الشكل **9.13** العديد من ميزات `JLabel`، بما في ذلك تسمية النص العادي وتسمية الصورة والتسمية مع كل من النص والصورة.

JLabel with text and with images

LabelDemo 1

```
package ch12GUI;
//Fig 9.13: LabelDemo.java
//Demonstrates the use of labels.
import java.awt.BorderLayout;//contains constants that specify where we can place GUI components
import javax.swing.ImageIcon;//represents an image that can be displayed on a Jlabel.
import javax.swing.JLabel;    // to create an object can display text, an image or both.
import javax.swing.JFrame;    // represents the window that will contain all the labels.
public class LabelDemo
{public static void main( String[] args )
{  // Create a label with plain text
   JLabel northLabel = new JLabel( "North" );
      // create an icon from an image so we can put it on a JLabel
      // ImageIcon can load images in GIF, JPEG and PNG image formats.
   ImageIcon labelIcon = new ImageIcon( "GUItip.gif" );
      // create a label with an Icon instead of text
   JLabel centerLabel = new JLabel( labelIcon );
      // create another label with an Icon
   JLabel southLabel = new JLabel( labelIcon );
```

الصنف LabelDemo

JLabel with text and with images

LabelDemo 2

```
    // set the label to display text (as well as an icon)
    southLabel.setText( "South" );
    // create a frame to hold the labels
    JFrame application = new JFrame("LabelDemo");
    application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );

    // add the labels to the frame; the second argument specifies
    // where on the frame to add the label
    application.add( northLabel, BorderLayout.NORTH );
    application.add( centerLabel, BorderLayout.CENTER );
    application.add( southLabel, BorderLayout.SOUTH );

    application.setSize( 300, 300 ); // set the size of the frame
    application.setVisible( true ); // show the frame
} // end main
} // end class LabelDemo
```

الصنف LabelDemo

Demonstrates the use of labels

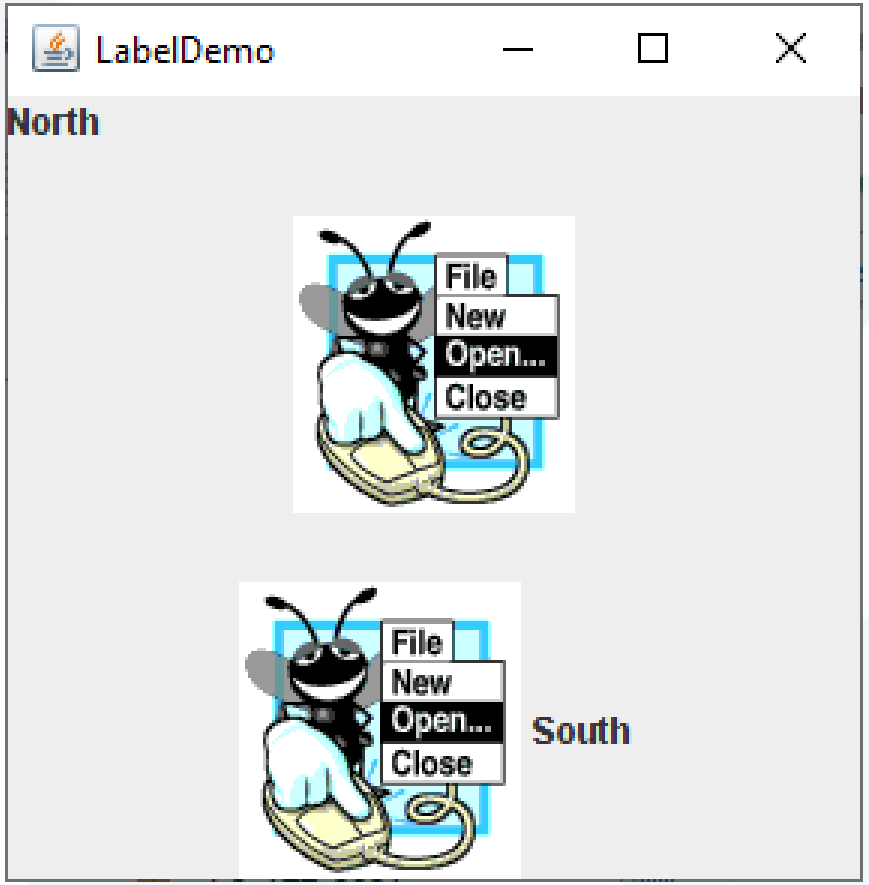


Fig. 9.13 | JLabel with text and with images

الصنف TestDraw.java

JLabel with text and with images.

9.12 GUI &Graphics

- ✓ يتلقى باني الصنف ImageIcon سلسلة تحدد المسار إلى الصورة.
- ✓ في حال تلقى اسم الملف فقط، تفترض Java أنه موجود في نفس الدليل مثل الصنف .LabelDemo.
- ✓ يمكن للصنف ImageIcon تحميل الصور بتنسيقات صورGIFو JPEGو PNG.
- ✓ تستدعي الطريقة setText لإضافة تسمية أو لتغيير النص الذي تعرضه التسمية. يمكن استدعاء الأسلوب setText على أي JLabel لتغيير نصه.
- ✓ يعرض هذا JLabel كلاً من الرمز والنص.
- ✓ من خلال استدعاء نسخة محملة بشكل زائد من طريقة إضافة تأخذ بارامترين. الأول هي المكون الذي نريد إرفاقه ، والثاني هو المنطقة التي يجب وضعها فيه.
- ✓ يحتوي كل إطار JFrameعلى تخطيط مرتبط يساعد JFrame في وضع مكونات واجهة المستخدم الرسومية المرفقة به.
- ✓ يُعرف التخطيط الافتراضي لـ JFrame باسم BorderLayout وله خمس مناطق

Five regions - North (top), South (bottom), East (right side), West (left side) and Center.

showInputDialog
showMessageDialog

12.3 Simple GUI-Based Input / Output with JOptionPane

- // display result in a JOptionPane message dialog

```
JOptionPane.showMessageDialog(null, "The sum is " + sum, "Sum of Two Integers",
JOptionPane.PLAIN_MESSAGE);
```

- قد تتضمن طريقة العرض الساكنه showMessageDialog من الصنف JOptionPane أربع متغيرات.
- الأول يحدد مكان ظهور مربع الحوار إذا غاب أو كان null ستكون في الوسط.
- الثاني الرسالة التي يجب عرضها - في هذه الحالة ، نتيجة تسلسل السلسلة "المجموع هو" وقيمة المجموع.
- الثالث السلسلة التي يجب أن تظهر في شريط العنوان أعلى مربع الحوار وهنا **"Sum of Two In tegers** .
- الرابع الايقونه التي ستظهر على يسار مربع الحوار وإن كانت PLAIN_MESSAGE لن تظهر ايقونه.
- توجد أربع ايقونات هي QUESTION_MESSAGE، INFORMATION_MESSAGE، WARNING_MESSAGE ، ERROR_MESSAGE والحالة PLAIN_MESSAGE بدون ايقونه.

showMessageDialog

JOptionPane static constants for message dialogs

Message dialog type	Icon	Description
ERROR_MESSAGE		Indicates an error.
INFORMATION_MESSAGE		Indicates an informational message.
WARNING_MESSAGE		Warns of a potential problem.
QUESTION_MESSAGE		Poses a question. This dialog normally requires a response, such as clicking a Yes or a No button.
PLAIN_MESSAGE	no icon	A dialog that contains a message, but no icon

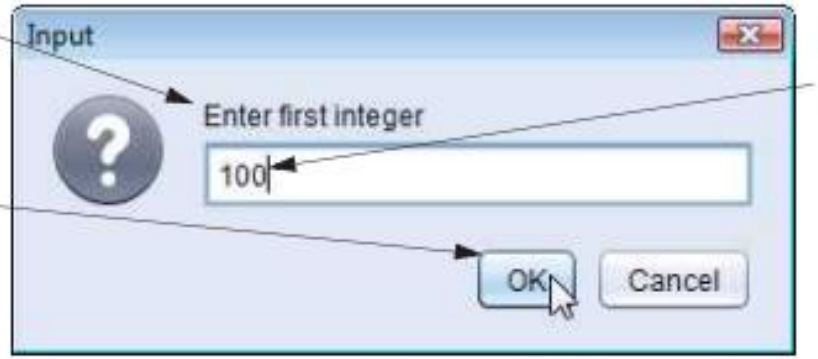
Fig. 12.3 | JOptionPane static constants for message dialogs.

program that uses JOptionPane for input and output

(a) Input dialog displayed by lines 10–11

Prompt to the user

When the user clicks **OK**, `showInputDialog` returns to the program the `100` typed by the user as a `String`; the program must convert the `String` to an `int`



Text field in which the user types a value

(b) Input dialog displayed by lines 12–13

(c) Message dialog displayed by lines 22–23—When the user clicks **OK**, the message dialog is dismissed (removed from the screen)

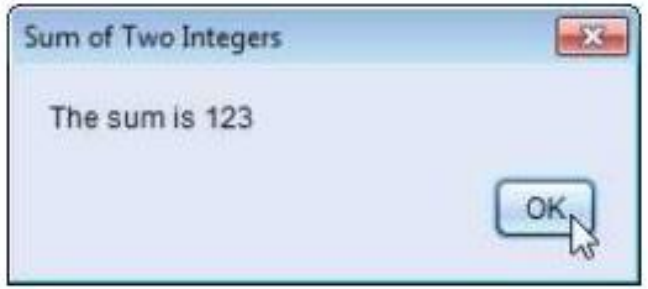
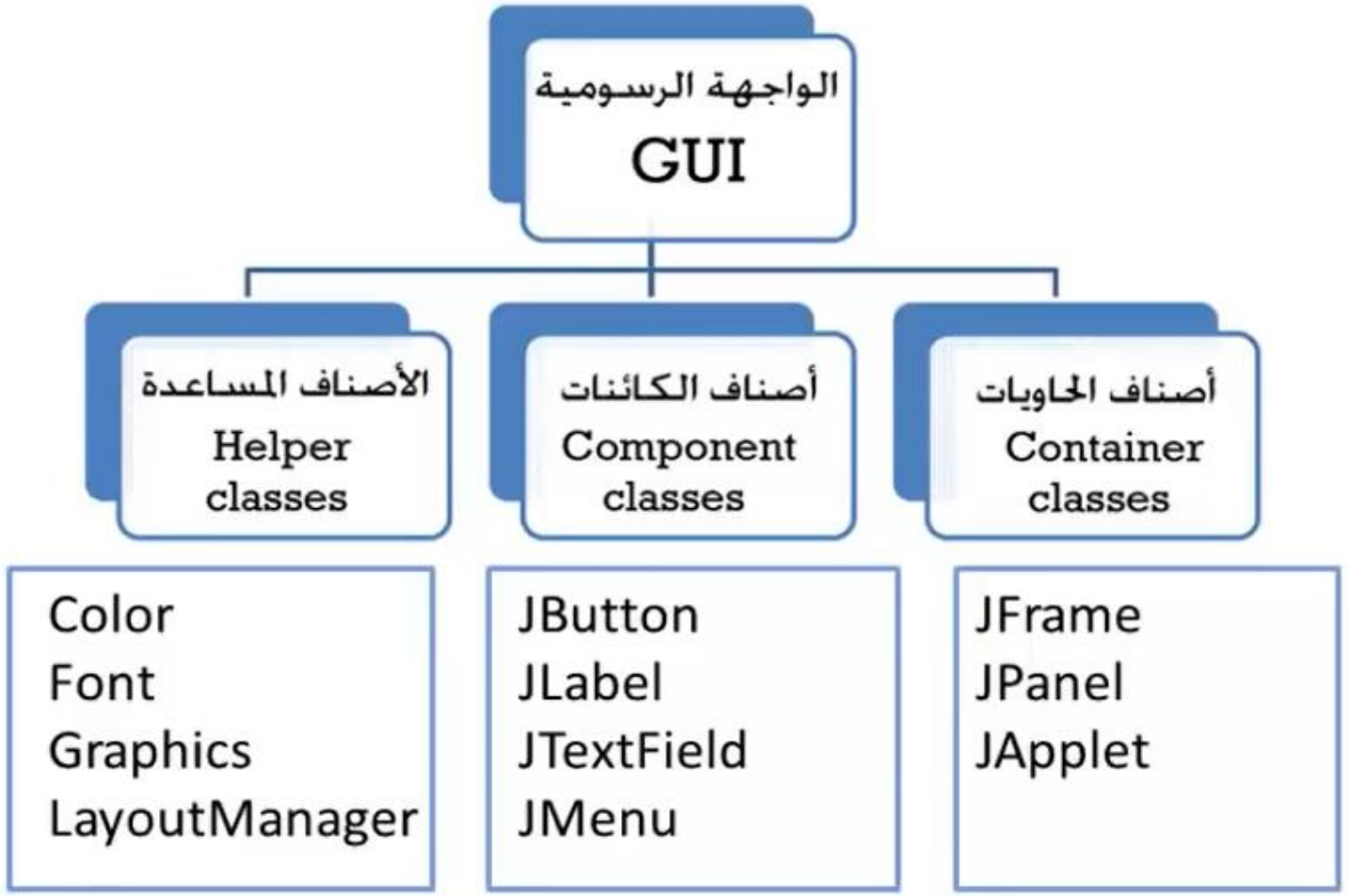


Fig. 12.2 | Addition program that uses `JOptionPane` for input and output. (Part 2 of 2.)

GUI Classes


الواجهة الرسومية
GUI
الأصناف المساعدة
Helper classes
أصناف الكائنات
Component classes
أصناف الحاويات
Container classes
Color
Font
Graphics
LayoutManager
JButton
JLabel
JTextField
JMenu
JFrame
JPanel
JApplet

Some basic Swing GUI components

Component	Description
JLabel	Displays uneditable text and/or icons.
JTextField	Typically receives input from the user.
JButton	Triggers an event when clicked with the mouse.
JCheckBox	Specifies an option that can be selected or not selected.
JComboBox	A drop-down list of items from which the user can make a selection.
JList	A list of items from which the user can make a selection by clicking on any one of them. Multiple elements can be selected.
JPanel	An area in which components can be placed and organized.

Fig. 12.4 | Some basic Swing GUI components.

12.5 Displaying Text and Images Using Labels

- JFrameهو صنف فرعي غير مباشرة من الصنف java.awt.Window الذي يوفر السمات والسلوكيات الأساسية للنافذة مثل شريط العنوان في الأعلى، وأزرار لتصغير النافذة وتكبيرها وإغلاقها.
- تتكون GUI النموذجية من العديد من المكونات. غالبًا ما يقدم مصممو GUI نصًا يوضح الغرض من كل منها.
- نص التسمية يتم إنشاؤه باستخدام JLabel - صنف فرعية من JComponent.
- نادرًا ما تغير التطبيقات محتويات التسميات بعد إنشائها.
- لكل عنصر GUI العديد من الميزات أكثر مما يمكننا تغطيته في الأمثلة الخاصة بنا.
- لمعرفة التفاصيل الكاملة، قم بزيارة صفحته في التوثيق عبر الإنترنت.
- docs.oracle.com/javase/7/docs/api/javax/swing/JLabel.html.

12.5 Displaying Text and Images Using Labels

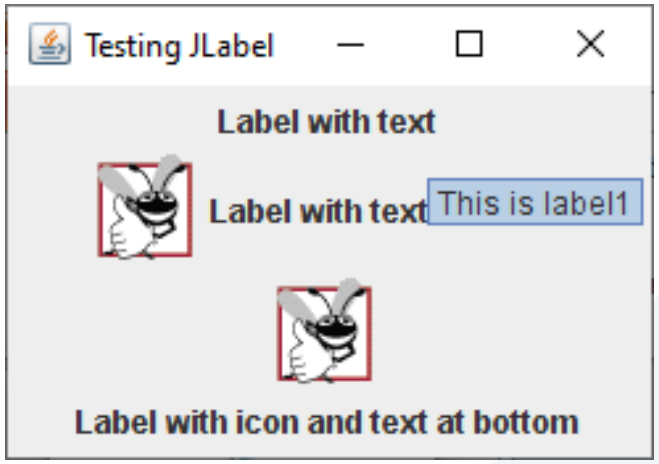
```
package ch12GUI;
//Fig. 12.6: LabelFrame.java     JLabels with text and icons.
import java.awt.FlowLayout; // specifies how components are arranged
import javax.swing.JFrame;  // provides basic window features
import javax.swing.JLabel;  // displays text and/or images
import javax.swing.SwingConstants; // common constants used with Swing
import javax.swing.Icon;    // interface used to manipulate images
import javax.swing.ImageIcon;       // loads images
public class LabelFrame extends JFrame
{private final JLabel label1; // JLabel with just text
private final JLabel label2;  // JLabel constructed with text and icon
private final JLabel label3;  // JLabel with added text and icon
// LabelFrame constructor adds JLabels to JFrame
public LabelFrame()
{    super("Testing JLabel");
   setLayout(new FlowLayout());    // set frame layout
```

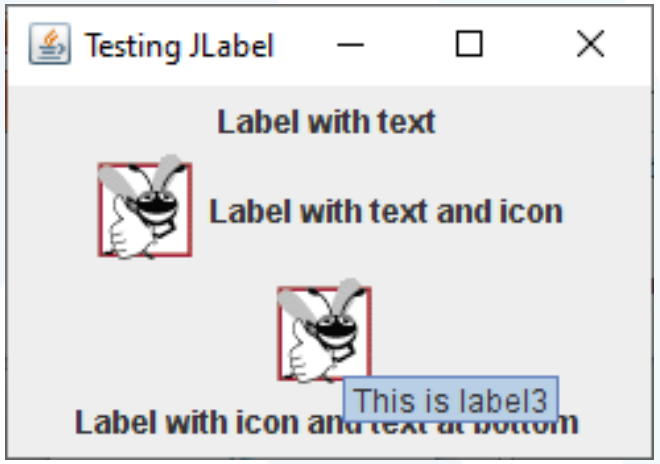
12.5 Displaying Text and Images Using Labels

```
        // JLabel constructor with a string argument
        label1 = new JLabel("Label with text");
        label1.setToolTipText("This is label1");
        add(label1); // add label1 to JFrame
        // JLabel constructor with string, Icon and alignment arguments
        Icon bug = new ImageIcon(getClass().getResource("bug1.png"));
        label2 = new JLabel("Label with text and icon", bug, SwingConstants.LEFT);
        label2.setToolTipText("This is label2");
        add(label2); // add label2 to JFrame
        label3 = new JLabel(); // JLabel constructor no arguments
        label3.setText("Label with icon and text at bottom");
        label3.setIcon(bug); // add icon to JLabel
        label3.setHorizontalTextPosition(SwingConstants.CENTER);
        label3.setVerticalTextPosition(SwingConstants.BOTTOM);
        label3.setToolTipText("This is label3");
        add(label3); // add label3 to JFrame
    }
} // end class LabelFrame
```

12.5 Displaying Text and Images Using Labels

```
 package ch12GUI;
//Fig. 12.7: LabelTest.java   Testing LabelFrame.
import javax.swing.JFrame;
public class LabelTest
{public static void main(String[] args)
{   LabelFrame labelFrame = new LabelFrame();
    labelFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    labelFrame.setSize(260, 180);
    labelFrame.setVisible(true); }
} // end class LabelTest
```





- تحميل مصدر الصورة في السطر، يستدعي التعبير getClass().getResource("bug1.png") المنهج getClass() (الموروثة بشكل غير مباشر من الصنف Object) لاسترداد مرجع إلى كائن Class الذي يمثل تعريف صنف LabelFrame. يتم بعد ذلك استخدام هذا المرجع لاستدعاء getResource التابع لمنهج Class، والذي يُرجع موقع الصورة كعنوان URL. يستخدم منشئ ImageIcon عنوان URL لتحديد موقع الصورة، ثم يقوم بتحميلها .
- تقوم JVM بتحميل تصريحات الأصناف في الذاكرة باستخدام مُحمل الصنف. يعرف مُحمل الصنف مكان وجود كل صنف يقوم بتحميلها على القرص. تستخدم طريقة getResource مُحمل صنف كائن الصنف لتحديد موقع المورد، مثل ملف صورة. في هذا المثال، يتم تخزين ملف الصورة في نفس الموقع مثل ملف LabelFrame.class. تعمل التقنيات الموضحة هنا على تمكين التطبيق من تحميل ملفات الصور من المواقع المرتبطة بموقع ملف الصنف .
- تحميل واجهة موارد الصور SwingConstants (packagejavax.swing) تعلن عن مجموعة من الثوابت الصحيحة الشائعة (مثل SwingConstants.LEFT وSwingConstants.CENTER وSwingConstants.RIGHT) التي يتم استخدامها مع العديد من مكونات Swing. افتراضيًا، يظهر النص على يمين الصورة عندما تحتوي التسمية على نص وصورة معًا. يمكن ضبط المحاذاة الأفقية والرأسية لـ JLabel باستخدام الأساليب setHorizontalAlignment وsetVerticalAlignment، على التوالي. يحدد السطر نص تلميح الأداة للتسمية 2، ويضيف السطر التسمية 2 إلى JFrame.

يتم قراءة البيانات من لوحة المفاتيح بصيغة أسكي إذا كانت القراءة بايت واحد وبصيغة سلسلة إذا كانت القراءة بعدة بايتات وعلى المبرمج تحويل من سلسلة نصية إلى أرقام عددية int باستخدام الدوال الخاصة بالسلاسل للتعامل معها كأعداد

```
package Rectangle7_10_2023;
import java.io.*;
public class Excepy {
    public static void main(String args[])throws IOException
      {// American Standard Code for Information Interchange
 int b; b=System.in.read();
//for(int i='a'; i<='z';i++)
System.out.println("ASCII "+b);
   }
 }
```

انتهت محاضرة الأسبوع السادس

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

Drawing Arcs


```
// Fig. 6.11: DrawSmiley.java    // Demonstrates filled shapes.
 import java.awt.Color;
import java.awt.Graphics;
import javax.swing.JPanel;
public class DrawSmiley extends JPanel
{    public void paintComponent(Graphics g)
    {    super.paintComponent(g);          // draw the face
        g.setColor(Color.YELLOW);       g.fillOval(10, 10, 200, 200);
        // draw the eyes
        g.setColor(Color.BLACK);
        g.fillOval(55, 65, 30, 30);     g.fillOval(135, 65, 30, 30);
        // draw the mouth
        g.fillOval(50, 110, 120, 60);
        // "touch up" the mouth into a smile
        g.setColor(Color.YELLOW);
        g.fillRect(50, 110, 120, 30);   g.fillOval(50, 120, 120, 40);
    }
} // end class DrawSmiley
```

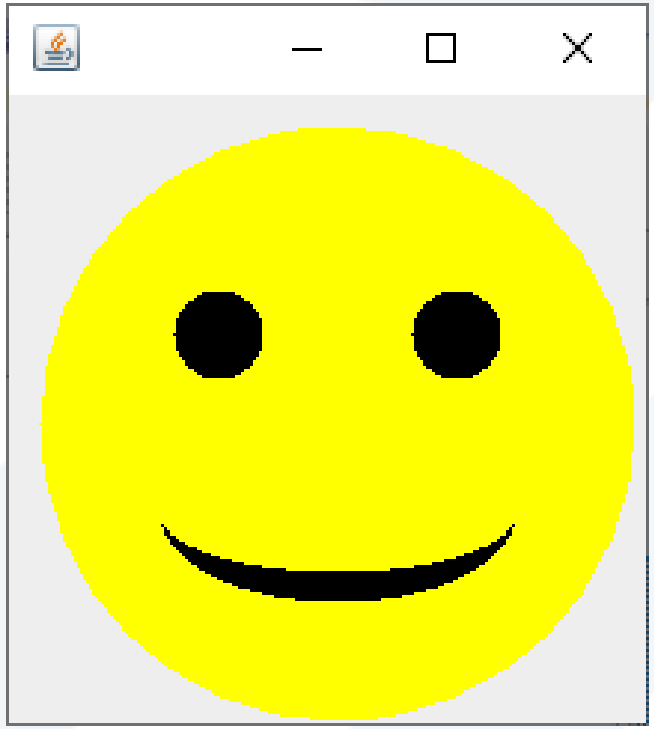
Drawing Arcs


```
// Fig. 6.12: DrawSmileyTest.java
// Test application that displays a smiley face.
import javax.swing.JFrame;

public class DrawSmileyTest
{
   public static void main(String[] args)
   {
      DrawSmiley panel = new DrawSmiley();
      JFrame application = new JFrame();

      application.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      application.add(panel);
      application.setSize(230, 250);
      application.setVisible(true);
   }
} // end class DrawSmileyTest
```

colors and filled shapes

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

Graphics methods fillRect and fillOval draw filled rectangles and ovals.



Creating Simple Drawings

5.25 GUI &Graphics

- ✓ رسم الأقواس في Java يشبه رسم الأشكال البيضاوية - القوس هو مجرد جزء من الشكل البيضاوي.
- ✓ منهج الرسومات `fillArc` من الصنف `Graphics` يرسم قوسًا ممتلئًا.
- ✓ يتطلب المنهج `fillArc` ستة معاملات.
  - الأربعة الأولى تمثل المستطيل المحيط الذي سيرسم فيه القوس.
  - المعامل الخامس هو زاوية البداية ، مع عدم وجود معامل تشير إلى الصفر على محور x.
  - والسادس يحدد مقدار المسح ، أو مقدار القوس المراد تغطيته.
    - يتم قياس زاوية البداية والمسح بالدرجات.
    - المسح الموجب يرسم القوس بعكس اتجاه عقارب الساعة والسالب معها.
- ✓ تتطلب طريقة `drawArc` نفس المعلمات مثل `fillArc`، ولكنها ترسم حافة القوس بدلاً من تعبئتها.
- ✓ منهج setBackground يغير لون الخلفية لمكون واجهة المستخدم الرسومية.

Drawing Arcs

7.25 GUI &Graphics

```
1  // Fig. 7.25: DrawRainbow.java
2  // Demonstrates using colors in an array.
3  import java.awt.Color;
4  import java.awt.Graphics;
5  import javax.swing.JPanel;
6
7  public class DrawRainbow extends JPanel
8  {
9     // define indigo and violet
10    private final static Color VIOLET = new Color( 128, 0, 128 );
11    private final static Color INDIGO = new Color( 75, 0, 130 );
12
13    // colors to use in the rainbow, starting from the innermost
14    // The two white entries result in an empty arc in the center
15    private Color[] colors =
16       { Color.WHITE, Color.WHITE, VIOLET, INDIGO, Color.BLUE,
17         Color.GREEN, Color.YELLOW, Color.ORANGE, Color.RED };
18
19    // constructor
20    public DrawRainbow()
21    {
22       setBackground( Color.WHITE ); // set the background to white
23    } // end DrawRainbow constructor
24
```

Fig. 7.25 | Drawing a rainbow using arcs and an array of colors. (Part 1 of 2.)

Drawing Arcs


```
25      // draws a rainbow using concentric arcs
26      public void paintComponent( Graphics g )
27      {
28         super.paintComponent( g );
29
30         int radius = 20; // radius of an arc
31
32         // draw the rainbow near the bottom-center
33         int centerX = getWidth() / 2;
34         int centerY = getHeight() - 10;
35
36         // draws filled arcs starting with the outermost
37         for ( int counter = colors.length; counter > 0; counter-- )
38         {
39            // set the color for the current arc
40            g.setColor( colors[ counter - 1 ] );
41
42            // fill the arc from 0 to 180 degrees
43            g.fillArc( centerX - counter * radius,
44               centerY - counter * radius,
45               counter * radius * 2, counter * radius * 2, 0, 180 );
46         } // end for
47      } // end method paintComponent
48   } // end class DrawRainbow
```

Fig. 7.25 | Drawing a rainbow using arcs and an array of colors. (Part 2 of 2.)

Drawing Arcs


```
1  // Fig. 7.26: DrawRainbowTest.java
2  // Test application to display a rainbow.
3  import javax.swing.JFrame;
4
5  public class DrawRainbowTest
6  {
7     public static void main( String[] args )
8     {
9        DrawRainbow panel = new DrawRainbow();
10       JFrame application = new JFrame();
11
12       application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
13       application.add( panel );
14       application.setSize( 400, 250 );
15       application.setVisible( true );
16    } // end main
17 } // end class DrawRainbowTest
```

Fig. 7.26 | Creating `JFrame` to display a rainbow. (Part 1 of 2.)

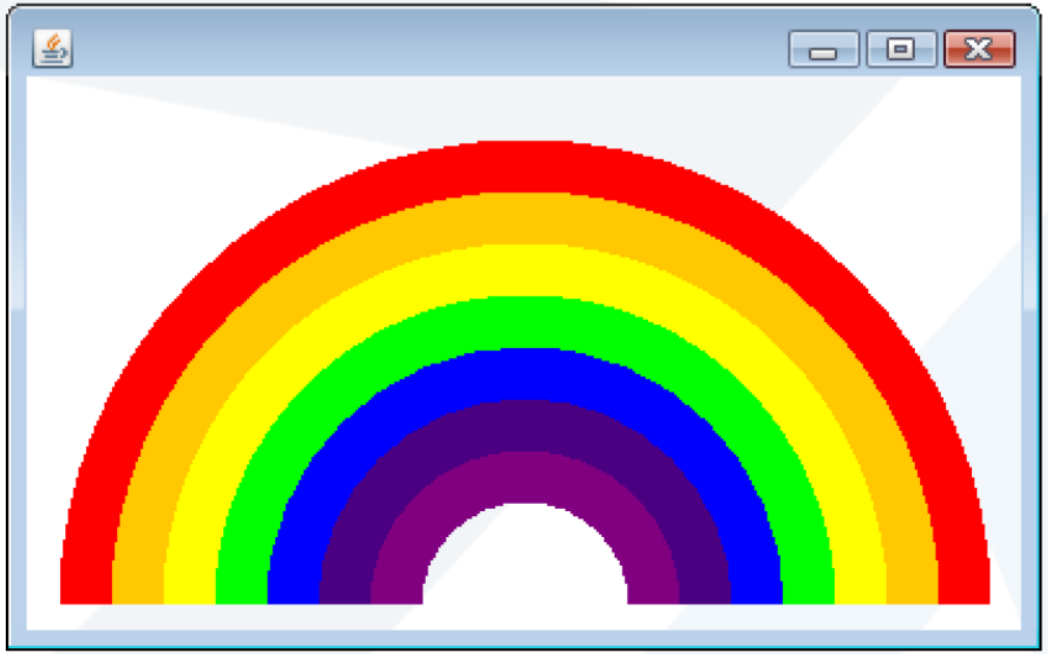


Fig. 7.26 | Creating `JFrame` to display a rainbow. (Part 2 of 2.)

7.26 GUI &Graphics

السطران يعلنان ويتنشئان لونين جديدين

```
final Color VIOLET = new Color( 128, 0, 128 );
final Color INDIGO = new Color( 75, 0, 130 );
```

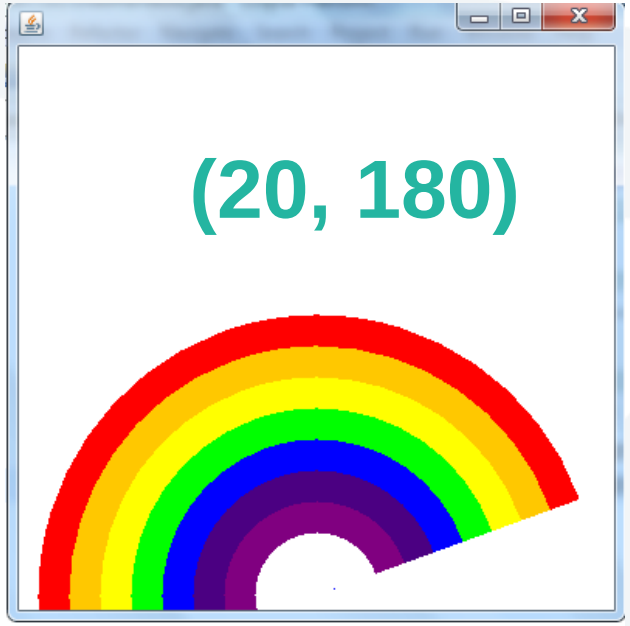
- أن ألوان قوس قزح هي **الأحمر والبرتقالي والأصفر والأخضر والأزرق والنيلي والبنفسجي.**
- تحتوي Java فقط على ثوابت سابقة التعريف للألوان الخمسة الأولى.
- الاسطر تهيئة مصفوفة مع ألوان قوس قزح، بدءاً من الأقواس الأعمق أولاً يبدأ المصفوفة بعنوين Color.WHITE، والتي، كما سترى قريبًا، هي لرسم الأقواس الفارغة في مركز قوس قزح.

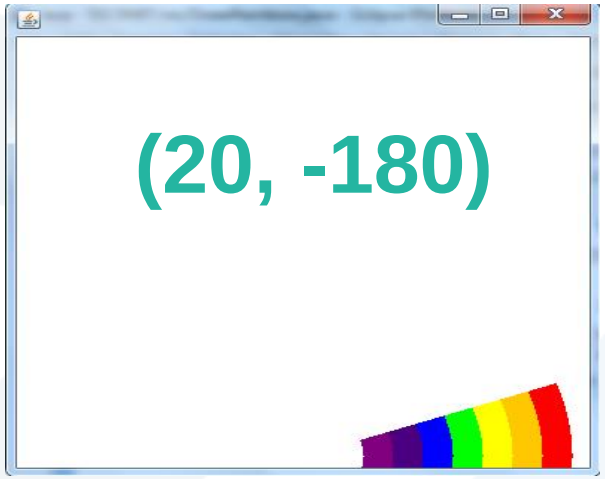
```
private Color colors[] = { Color.WHITE, Color.WHITE, VIOLET, INDIGO,
Color.BLUE, Color.GREEN, Color.YELLOW, Color.ORANGE, Color.RED };
```

- يمكن تهيئة متغيرات الحالة عند إعلانها. يحتوي المُنشئ على عبارة واحدة تستدعي طريقة setBackground (الذي تم توارثه من الفئة JPanel) بالمتغير Color.WHITE. تأخذ مجموعة setBackground وسيطة لون واحدة وتجعل الخلفية للمكون من ذلك اللون.
- الطريقة paintComponent تعلن عن متغير يعبر عن نصف قطر الدائرة التي سترسم، أي يحدد سمك كل قوس. تحدد المتغيرات المحلية centerX وcenterY موقع نقطة الوسط على قاعدة قوس قزح.(منتصف X وعلى بعد 10 من كامل h).
- تستخدم الحلقة counter للتحكم والعد العكسي من نهاية المصفوفة، مع رسم أكبر قوس أولاً ووضع كل قوس أصغر متتابع أعلى القوس السابق. * يعين السطر 40 اللون لرسم القوس الحالي من المصفوفة. السبب في أن لدينا مداخل Color.WHITEفي بداية المصفوفة هي إنشاء قوس فارغ في المركز. خلاف ذلك، فإن مركز قوس قزح يكون مجرد دائرة نصف دائرة البنفسجي الصلبة.

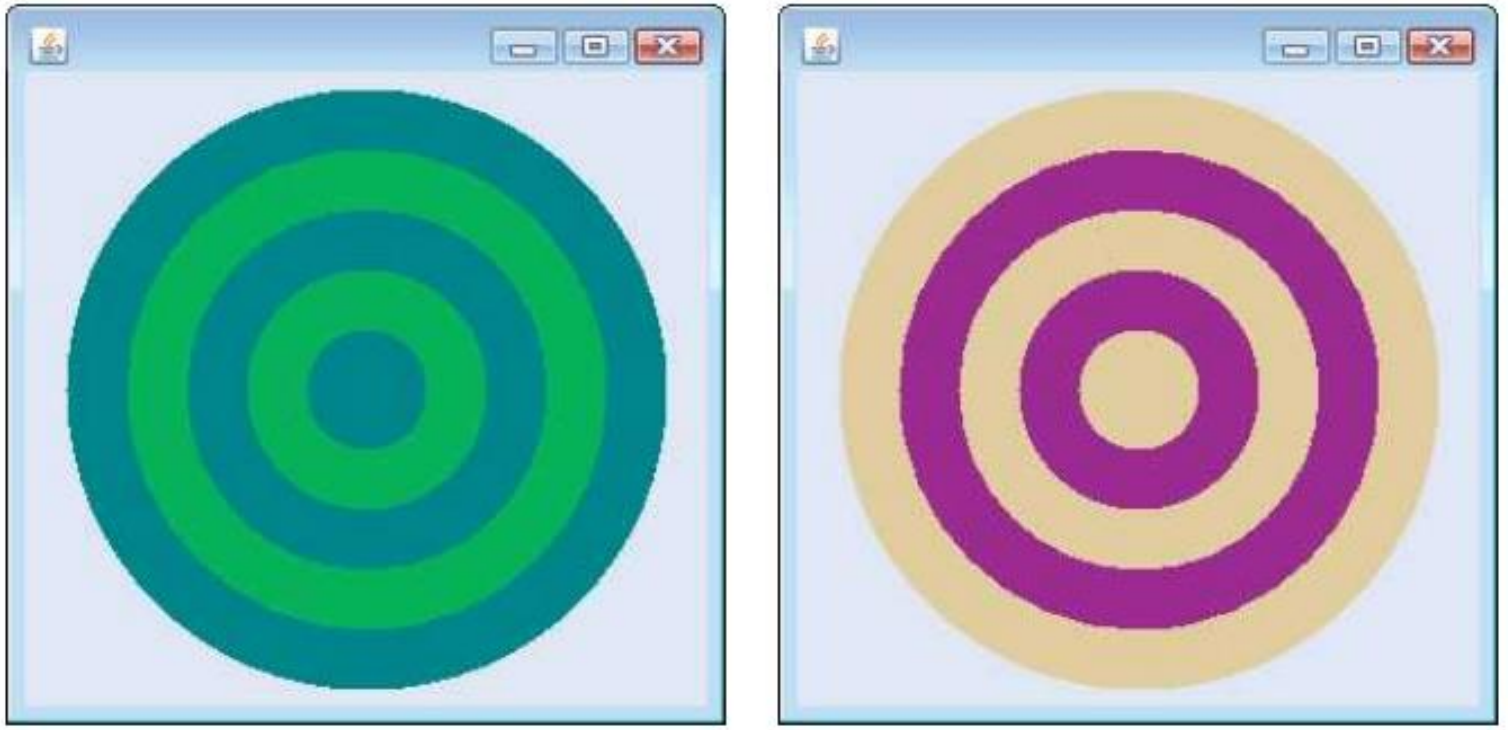
[ملاحظة: يمكنك تغييرألوان فردية وعدد الإدخالات في المصفوفة لإنشاء تصميمات جديدة.]

7.26 GUI &Graphics





7.26 GUI &Graphics



ليكن المطلوب رسم الشكل السابق

Drawing Arcs

جامعة المنارة
MANARA UNIVERSITY

DrawRainbow21 1

```
// Fig. 5.26: Shapes2.java
 // Demonstrates drawing different Shapes2.
 import java.awt.Color;
import java.awt.Graphics;
 import javax.swing.JPanel;
 import java.security.SecureRandom;
 //import java.util.Random; // program uses class Random
  public class DrawRainbow21 extends JPanel
  {
   // Define indigo and violet
   final Color VIOLET = new Color( 128, 0, 128 );
   final Color INDIGO = new Color( 75, 0, 130 );
   int x1,y1,w,h;
   // colors to use in the rainbow, starting from the innermost
   // The two white entries result in an empty arc in the center
   private Color colors[] =
      { Color.WHITE, Color.WHITE, VIOLET, INDIGO, Color.BLUE,
        Color.GREEN, Color.YELLOW, Color.ORANGE, Color.RED };
```

Drawing Arcs

DrawRainbow21 2

```
// constructor
   private int choice; // user's choice of which shape to draw

// constructor sets the user's choice

public DrawRainbow21(int userChoice)
   {choice = userChoice;
      setBackground( Color.WHITE ); // set the background to white
   } // end DrawRainbow21 constructor

   // draws a rainbow using concentric circles
   public void paintComponent( Graphics g )
   {      super.paintComponent( g );

// draws a cascade of Shapes21 starting from the top-left corner///
      SecureRandom randomNumbers = new SecureRandom();
// SecureRandom number generator
```

Drawing Arcs

DrawRainbow21 3

```
// pick the shape based on the user's choice
switch ( choice )
{case 1: // draw DrawRainbow2
  int radius = 40;// radius of an arch
     // draw the rainbow near the bottom-center
      int centerX = getWidth() / 2;
      int centerY = getHeight() - 10;

      // draws filled arcs starting with the outermost
      for ( int counter = colors.length; counter > 0; counter-- )
      {  // set the color for the current arc
         g.setColor( colors[ counter - 1 ] );

         // fill the arc from 0 to 180 degrees
         g.fillArc( centerX - counter * radius,
            centerY - counter * radius,
            counter * radius * 2, counter * radius * 2, 45, 90 );
```

Drawing Arcs

DrawRainbow21 4

```
 // set the color for the current arc
          /*g.setColor( colors[ counter - 1 ] );
          g.fillOval( 10 + counter * 10, 10 + counter * 10,
      240 - counter * 20, 240 - counter * 20 );}*/
          } break;
case 2: // set the color for the current arc
  for ( int counter = 8; counter > 0; counter-- )
  {  g.setColor( colors[ counter - 1 ] );
       g.fillOval(130-counter * 10,130-counter *10,counter * 20,counter * 20 );
  //g.fillOval(30+counter *10,30+counter *10,200-counter *20,200-counter *20);
  /*g.fillOval(10+counter*10,10+counter*10,240-counter*20,240-counter*20); */
  x1=130 - counter *10; y1=130 - counter * 10; w= counter * 20; h=counter * 20;
            System.out.println("x1=" +x1+ " y1= "+y1+ " w= "+w+" h= "+h);
     }  break;
case 3: // stores each random integer generated shapesType,shapesColor,(x1,y1)
begin ,widnth ,height
```

Drawing Arcs

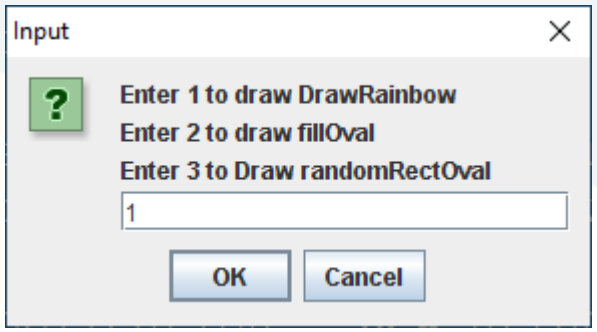

```
for ( int counter = 10; counter > 0; counter-- )
   {   int shapesType, shapesColor, x1, y1, x2, y2;
    shapesType = 1 + randomNumbers.nextInt( 2 );
    x1 = 1 + randomNumbers.nextInt( 200 ); y1 = 1 + randomNumbers.nextInt( 200 );
    w = 1 + randomNumbers.nextInt( 188 );  h = 1 + randomNumbers.nextInt( 188);
          System.out.println("x1=" +x1+ " y1= "+y1+ " w= "+w+" h= "+h);
  //if((x1+x2)>400) x2=380;  if((y1+y2)>400) y2=370;
          if(shapesType==1)
 {  // draw rectangles   and set the color for the current rectangles
  shapesColor = 2 + randomNumbers.nextInt( 6 );
   g.setColor( colors[ shapesColor ] ); g.fillRect( x1, y1, w, h ); }
 else { // draw ovals and set the color for the current ovals
 shapesColor = 2 + randomNumbers.nextInt( 6 );
 g.setColor( colors[ shapesColor ] );  g.fillOval( x1, y1, w, h );}  //break;
          } } // end for
}// end method paintComponent
   }// end class DrawRainbow21
```

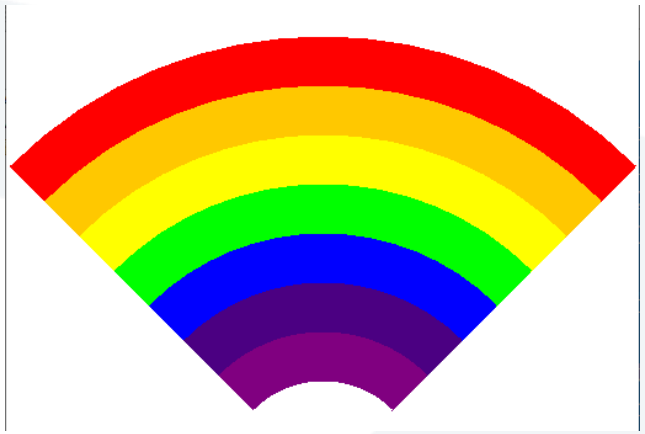
Drawing Arcs

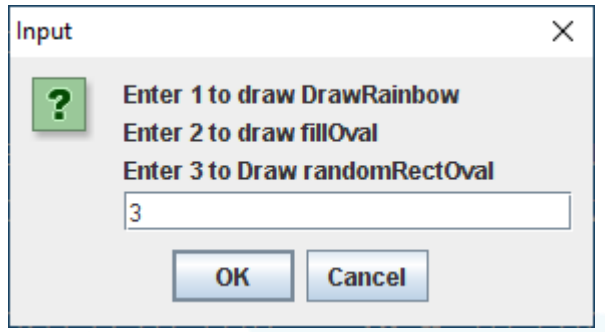
DrawRainbow21Test

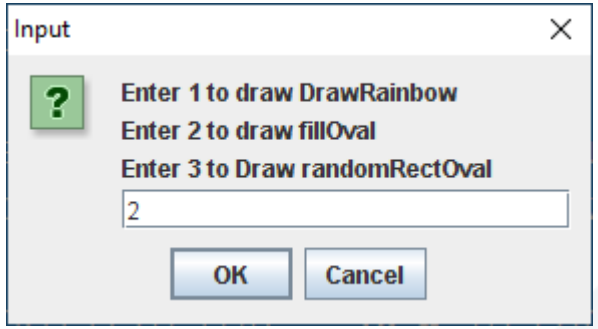
```
public class DrawRainbow21Test
 {  public static void main( String[] args )
 {  // obtain user's choice
 String input = JOptionPane.showInputDialog(
 "Enter 1 to draw DrawRainbow \n" +
 "Enter 2 to draw fillOval\n"+
 "Enter 3 to Draw randomRectOval\n");
 int choice = Integer.parseInt( input ); // convert input to int
 // create the panel with the user's input
 DrawRainbow21 panel = new DrawRainbow21( choice );
 JFrame application = new JFrame(); // creates a new JFrame
 application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
 application.setSize( 400, 400 ); // set the desired size
 application.add( panel ); // add the panel to the frame
 application.setVisible( true ); // show the frame
 } // end main
 } // end class DrawRainbow1Test
```

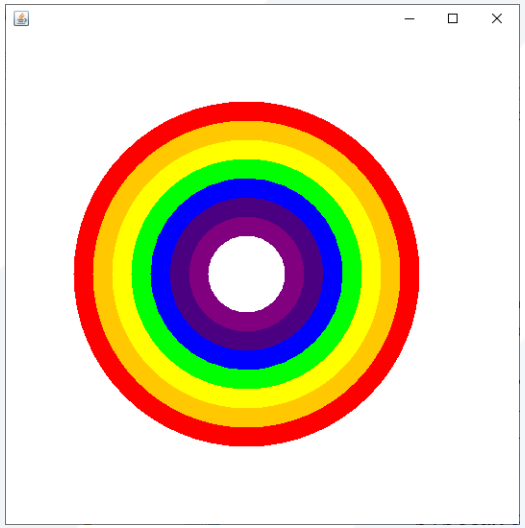
Drawing Arcs

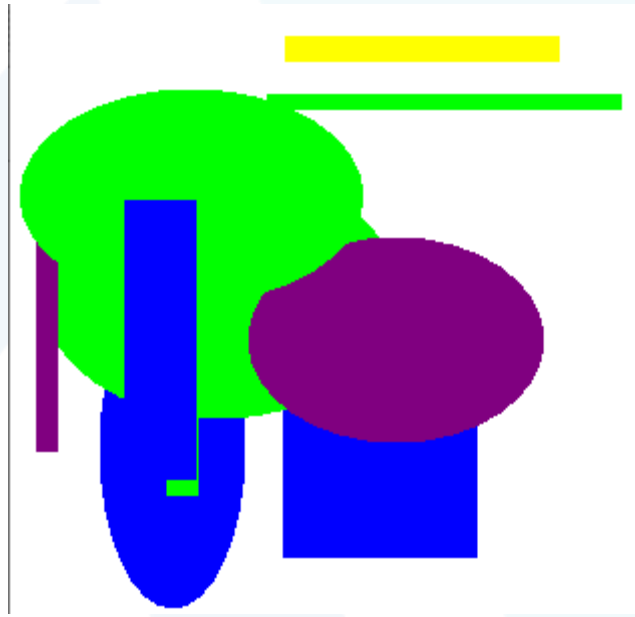




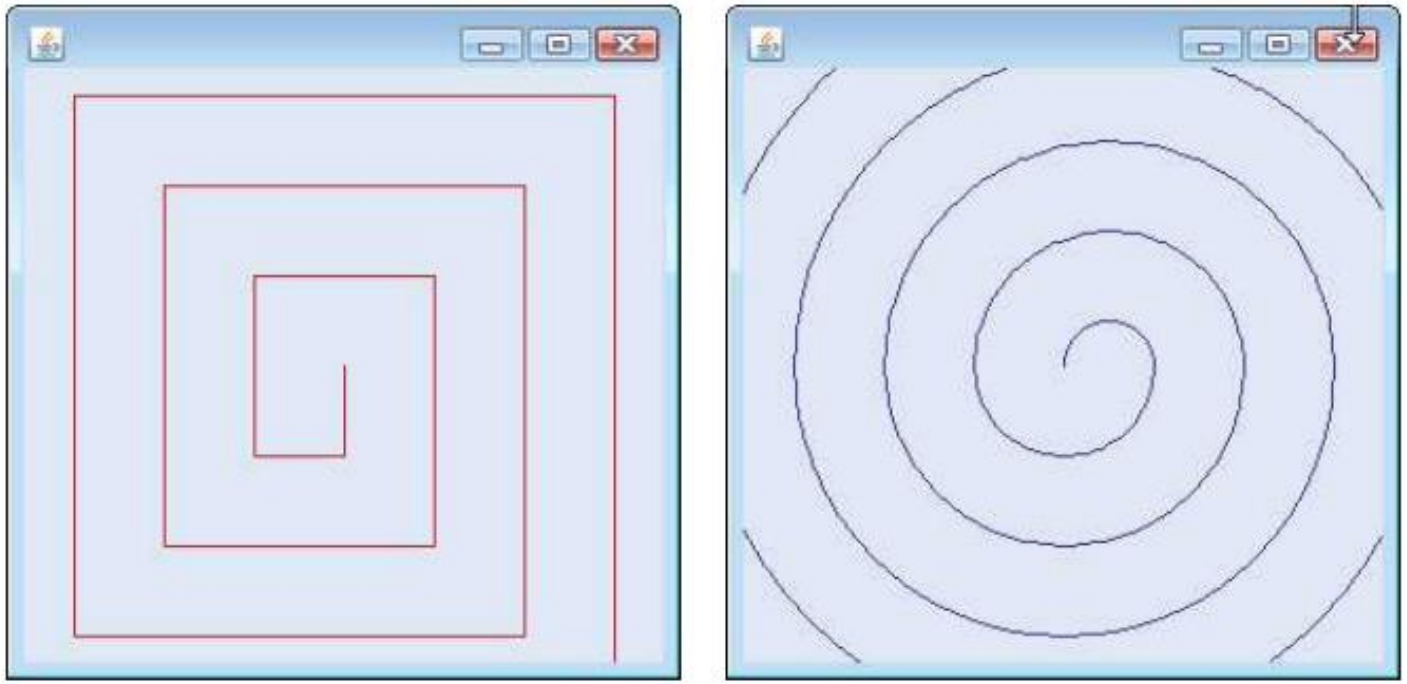








Drawing Arcs



ليكن المطلوب رسم الشكل السابق