جَامعة المَنارة
MANARA UNIVERSITY

كلية الهندسة المعلوماتية

برمجة 3

Java Programming

ا. د. علي عمران سليمان

محاضرات الأسبوع الخامس

الفصل الثاني 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).

- 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 ومتوفر في 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 ScalingFactorعامل 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. ;(5)int number = 2 + 3 * randomNumbers.nextInt

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

Enumerated Type s1

- عند تعريف متغير من نوع معين سيتقبل هذا النوع فقط 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.

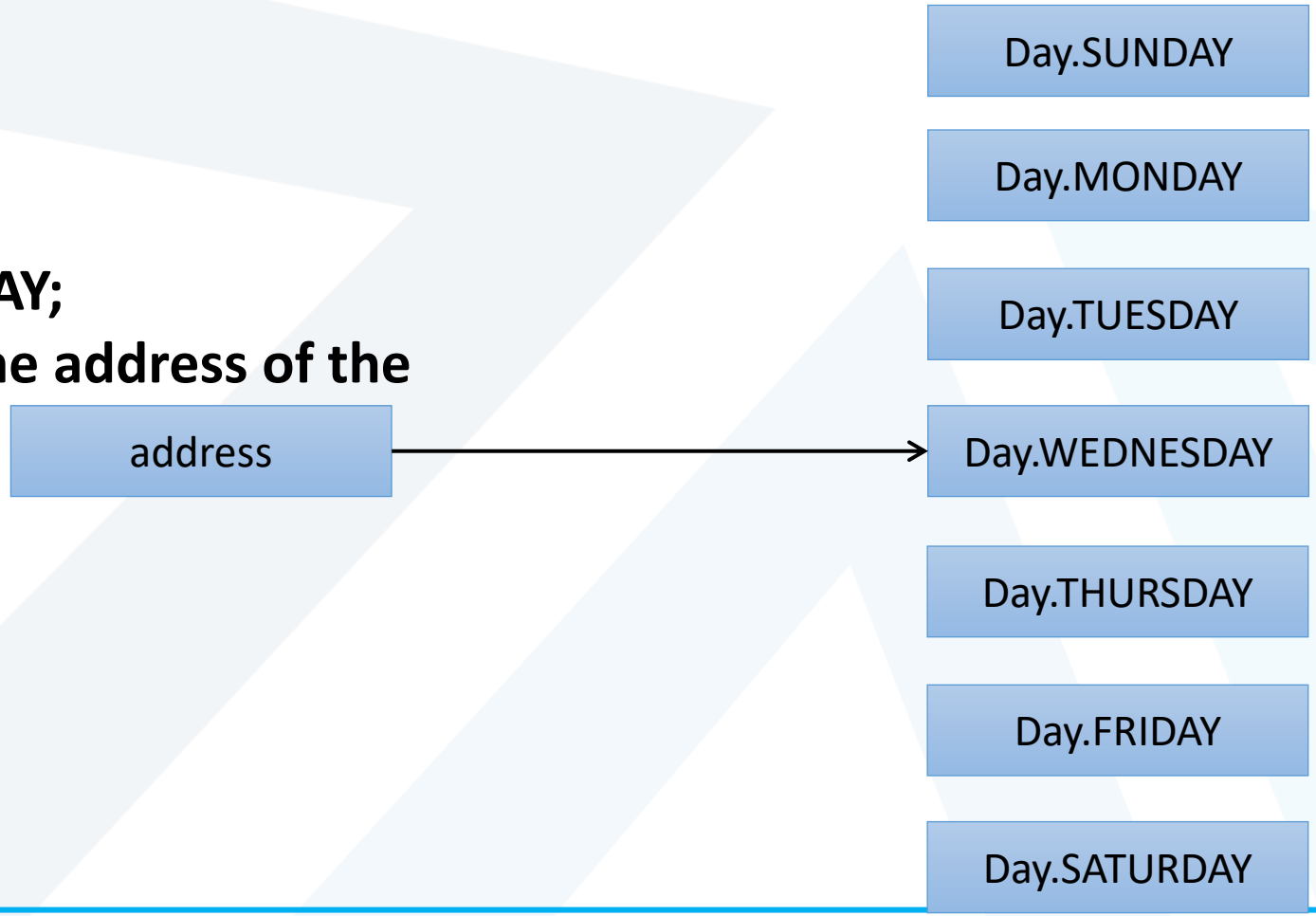
- **Enum** هو صنف خاص وأن **WorkDay** ليس سوى مرجعاً للكائن **Day.WEDNESDAY;** .

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 النواع 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.

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
```

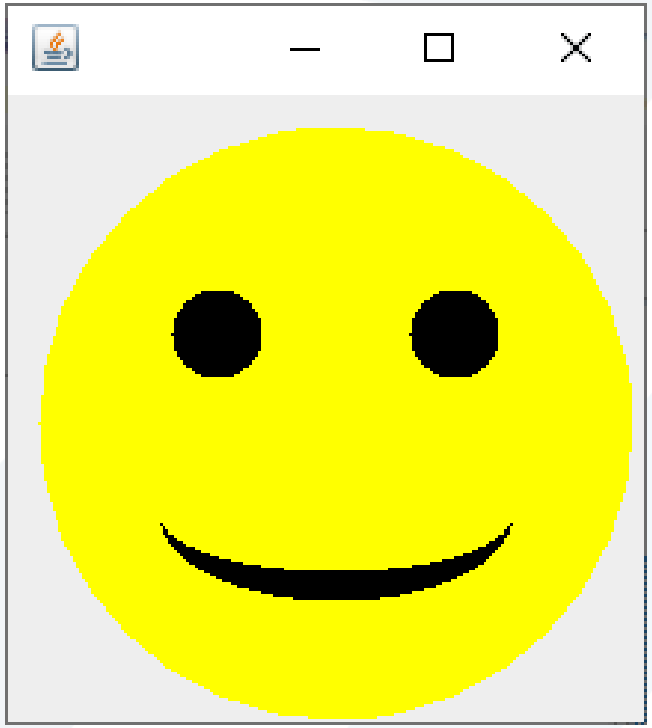
Drawing Arcs


```
// 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


```
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.)

Drawing Arcs

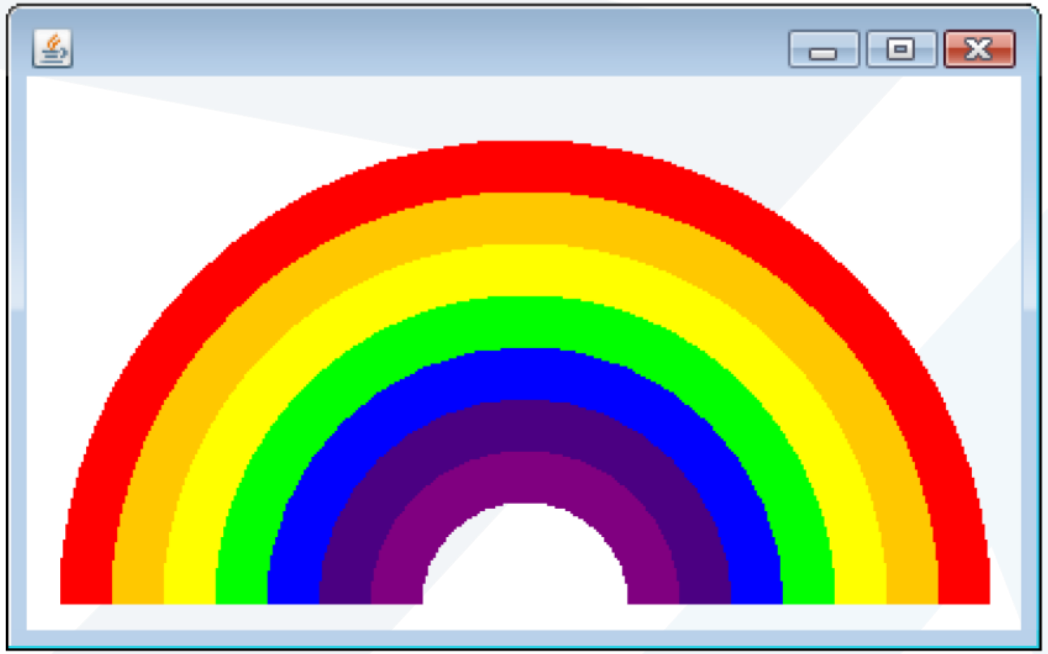


Fig. 7.26 | Creating `JFrame` to display a rainbow. (Part 2 of 2.)

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 );
```

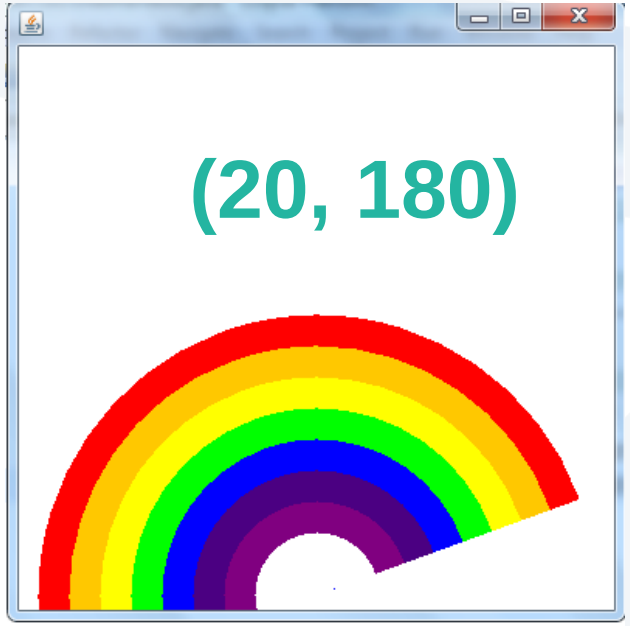
- أن ألوان قوس قزح هي **الأحمر والبرتقالي والأصفر والأخضر والأزرق والنيلي والبنفسجي.**
- تحتوي Java فقط على ثوابت سابقة التعريف للألوان الخمسة الأولى.
- الاسطر تهيئة مصفوفة مع ألوان قوس قزح، بدءاً من الأقواس الأعمق أولاً يبدأ المصفوفة بعنوين Color.WHITE، والتي، كما سترى قريبًا، هي لرسم الأقواس الفارغة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

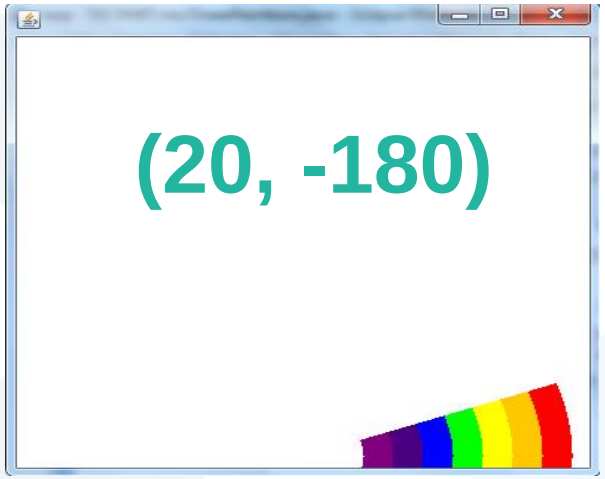
```
private Color colors[] = { Color.WHITE, Color.WHITE, VIOLET, INDIGO,
Color.BLUE, Color.GREEN, Color.YELLOW, Color.ORANGE, Color.RED };
```

- 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.

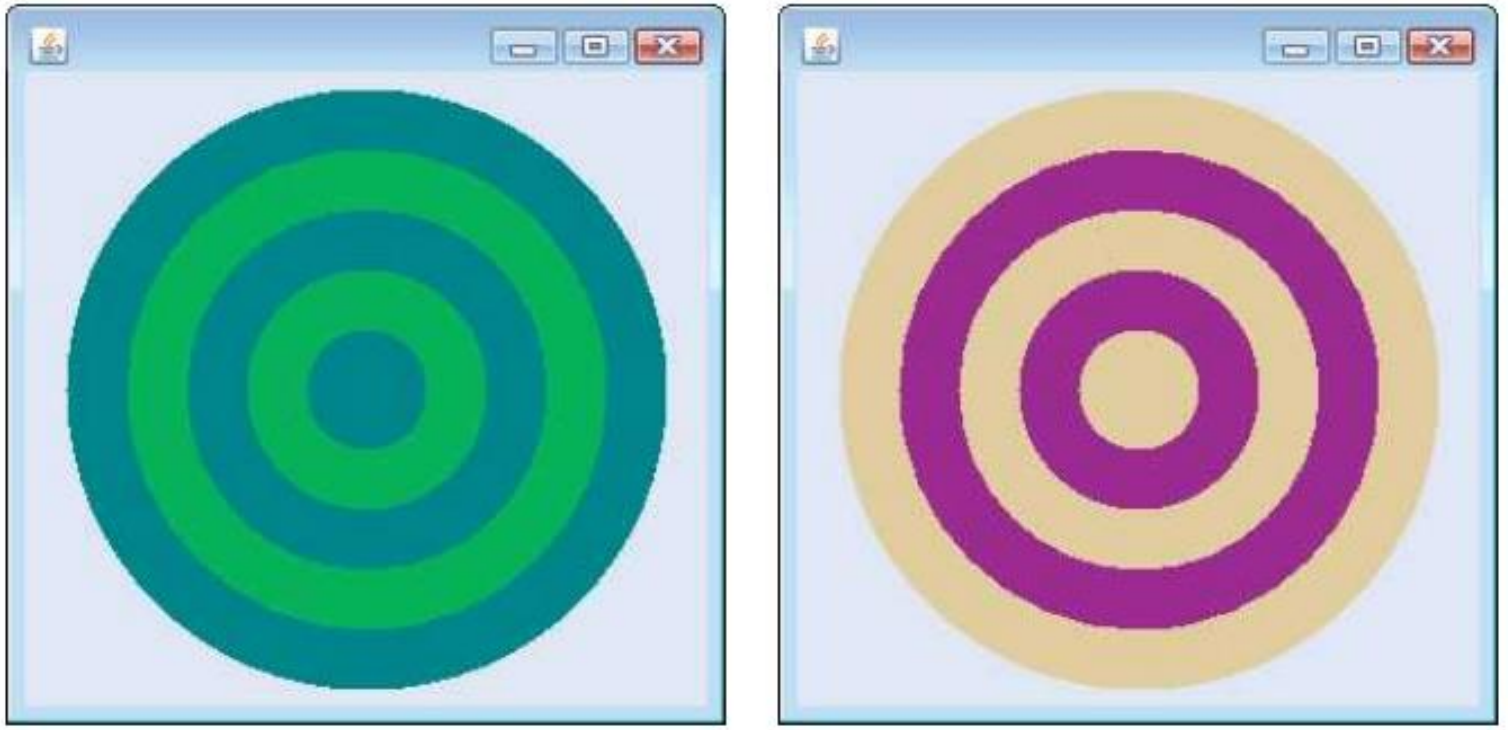
[ملاحظة: يمكنك تغييرألوان فردية وعدد الإدخالات في المصفوفة لإنشاء تصميمات جديدة.]

7.26 GUI &Graphics





7.26 GUI &Graphics



ليكن المطلوب رسم الشكل السابق

Drawing Arcs

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 };
```

Drawing Arcs

جامعة المنارة
MANARA UNIVERSITY

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

جَامعة المَنارة
MANARA UNIVERSITY

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

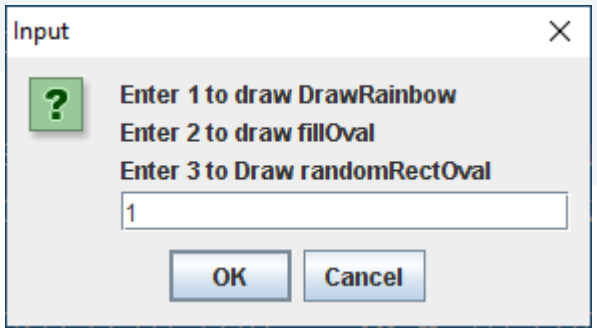
DrawRainbow21Test

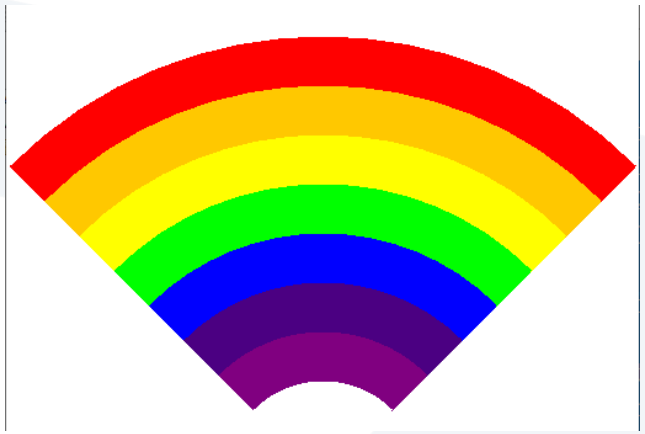
```
public class DrawRainbow21Test
 {  public static void main( String[] args )
 {  // obtain user's choice
 String input = JOptionPane.showInputDialog(
 "Enter 1 to draw DrawRainbow \n" +
 "Enter 2 to draw fillOval\n"+
 "Enter 3 to Draw randomRectOval\n");
 int choice = Integer.parseInt( input ); // 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
```

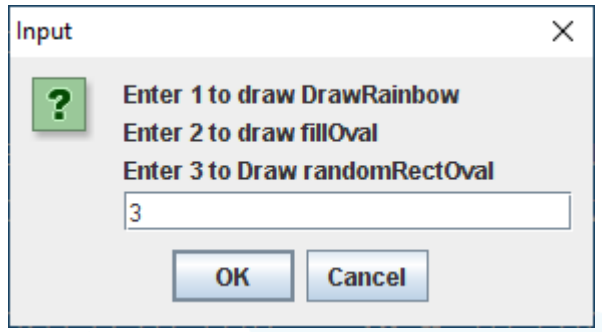
Drawing Arcs

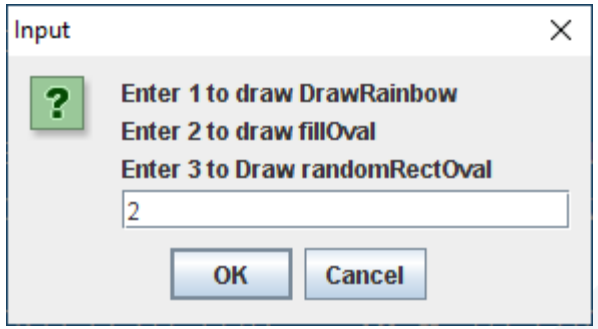
جَامعة المَنارة
MANARA UNIVERSITY

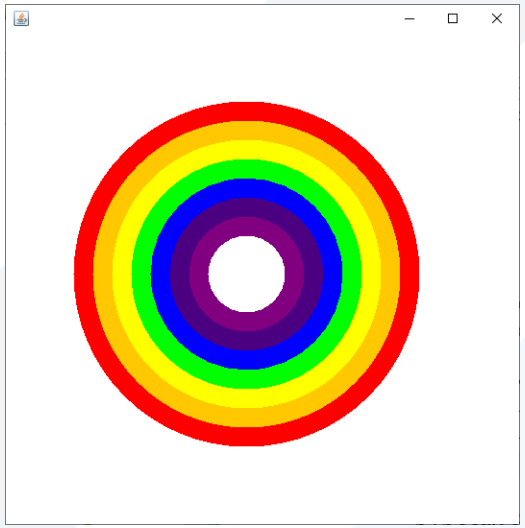
7.25 GUI &Graphics

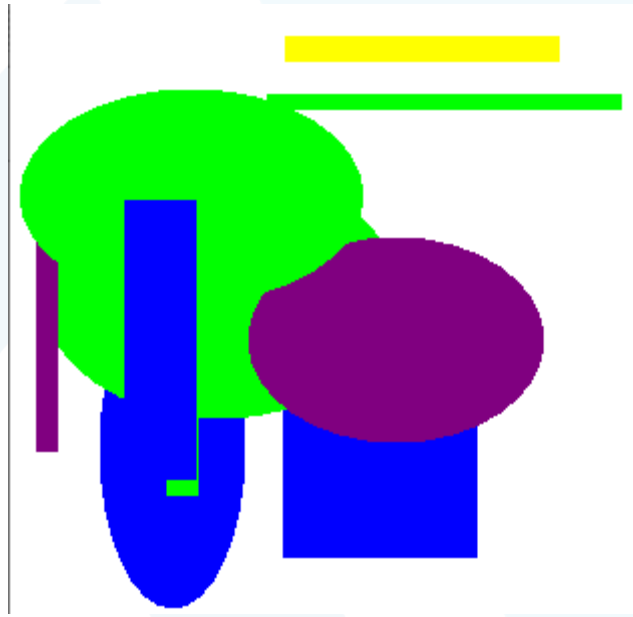




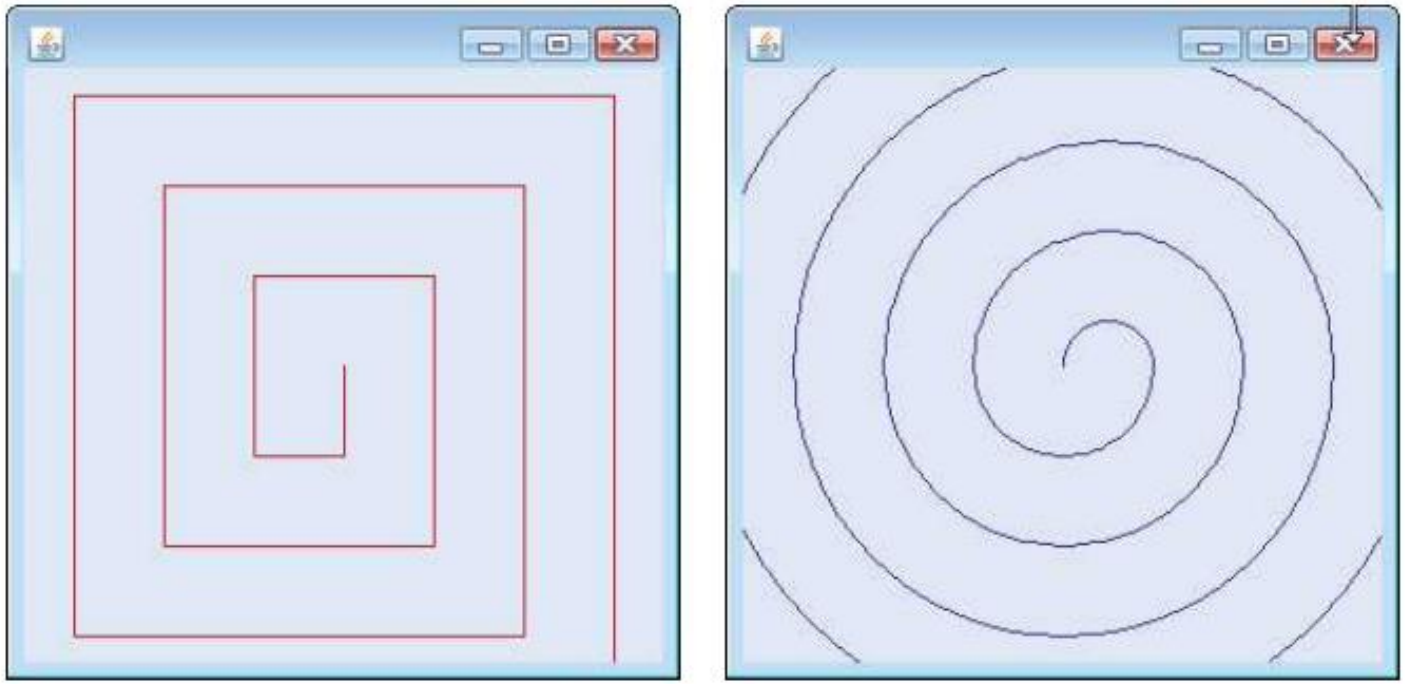








Drawing Arcs



ليكن المطلوب رسم الشكل السابق