جَامعة المَنارة
MANARA UNIVERSITY

كلية الهندسة المعلوماتية

برمجة 3

Java Programming

ا. د. علي عمران سليمان

محاضرات الأسبوع الخامس

الفصل الأول 2024-2025

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();`

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

Secure Random-Number Generation 2

- يمكن التحكم بنطاق الأعداد العشوائية المنتجة من خلال إرسال قيمة صحيحة للطريقة nextInt() مابين القوسين.
- مثلاً للتعبيرعن الشعارH بـ0 والنقش T بـ1 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 الصفروإلى العدد السابق للعدد المرسل.

```
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. ;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 المحجوزله 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; .

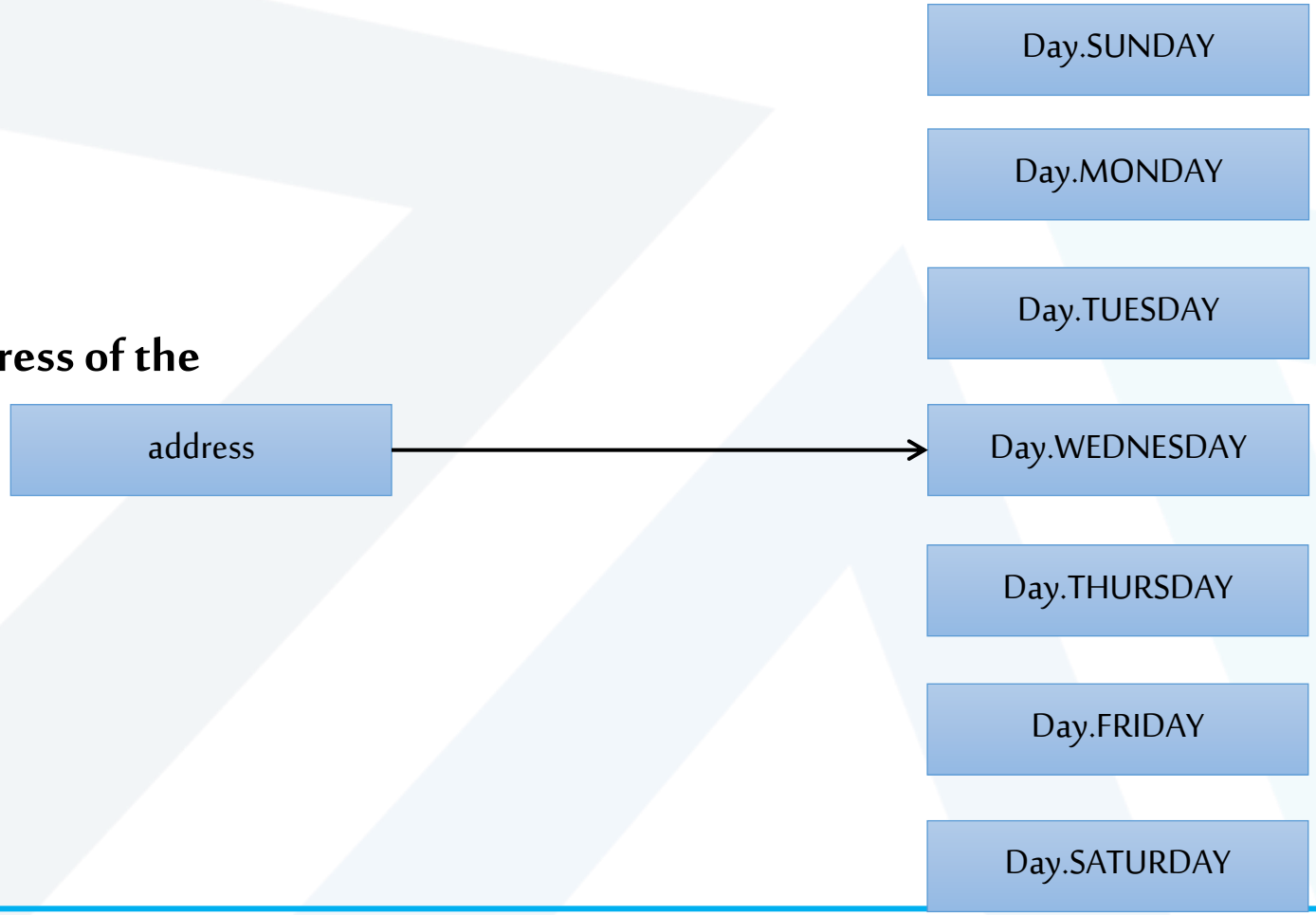
Enumerated Type s1

إن Enumهوصنف 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 التعداديه 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 متغيرين 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.

انتهت محاضرات الأسبوع الخامس