البرمجة غرضية التوجه

Object Oriented Programming

Dr. REEMA AL-KAMHA

الهدف من المقرر

- تعريف الطالب بأهم مبادئ وتقنيات البرمجة الغرضية التوجه، وإلى تطبيق لغة الجافا لبناء برامج قابلة للتوسع والتطوير بطريقة مضبوطة ومنهجية باعتماد هذه التقنيات.
- تمكين الطالب من بناء تطبيقات متصلة بقواعد بيانات و ذات واجهات رسومية لنمذجة مسائل واقعية

مفردات مادة البرمجة غرضية التوجه
Object Oriented Programming

أساسيات لغة الجافا JAVA Essentials

- بنية برنامج بلغة جافا Structure of JAVA Program
- المتغيرات Variables
- المعاملات Operators
- الأنواع الأولية Primitive Types
- تعليمة الطباعة Console Output

بنى التحكم Control Structures

- العبارات الشرطية (if/else)
- البنى التكرارية (for, while, do/while)
- بنية الاختيار المتعدد (switch)
- عبارتي break و continue
- المعاملات المنطقية Logical Operators

الطرق Methods

- التحميل الزائد للطريقة Method Overloading

السلاسل Strings

المتجهات Arrays

الأغراض و الصفوف Objects and Classes

- المرجع (this The this Reference)
- مفهوم static

تمثيل علاقتي (Composition and Aggregation)

Object-Based Assembly of New Classes by Composition and Aggregation

مفردات مادة البرمجة غرضية التوجه
Object Oriented Programming

الوراثة Inheritance
- الكلمة المفتاحية super
- المؤثر final
- إعادة تعريف الطريقة Method Overriding

تعددية الأشكال Polymorphism

الصفوف المجردة Abstract Classes

الواجهات Interfaces

الاستثناءات Exceptions

المجموعات في الجافا Collections in JAVA

الملفات Files

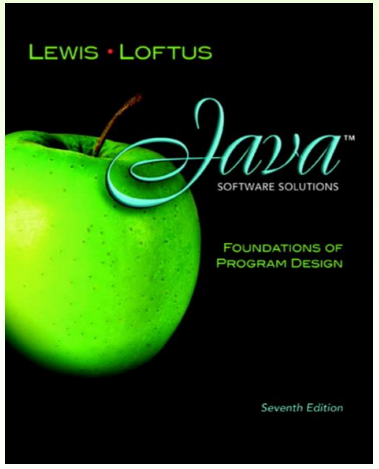
تصميم واجهات المستخدم Graphical User Interfaces

إنشاء قواعد بيانات في الجافا

Creating Database Applications Using JAVA Persistence JPA (JPA)

مراجع المادة

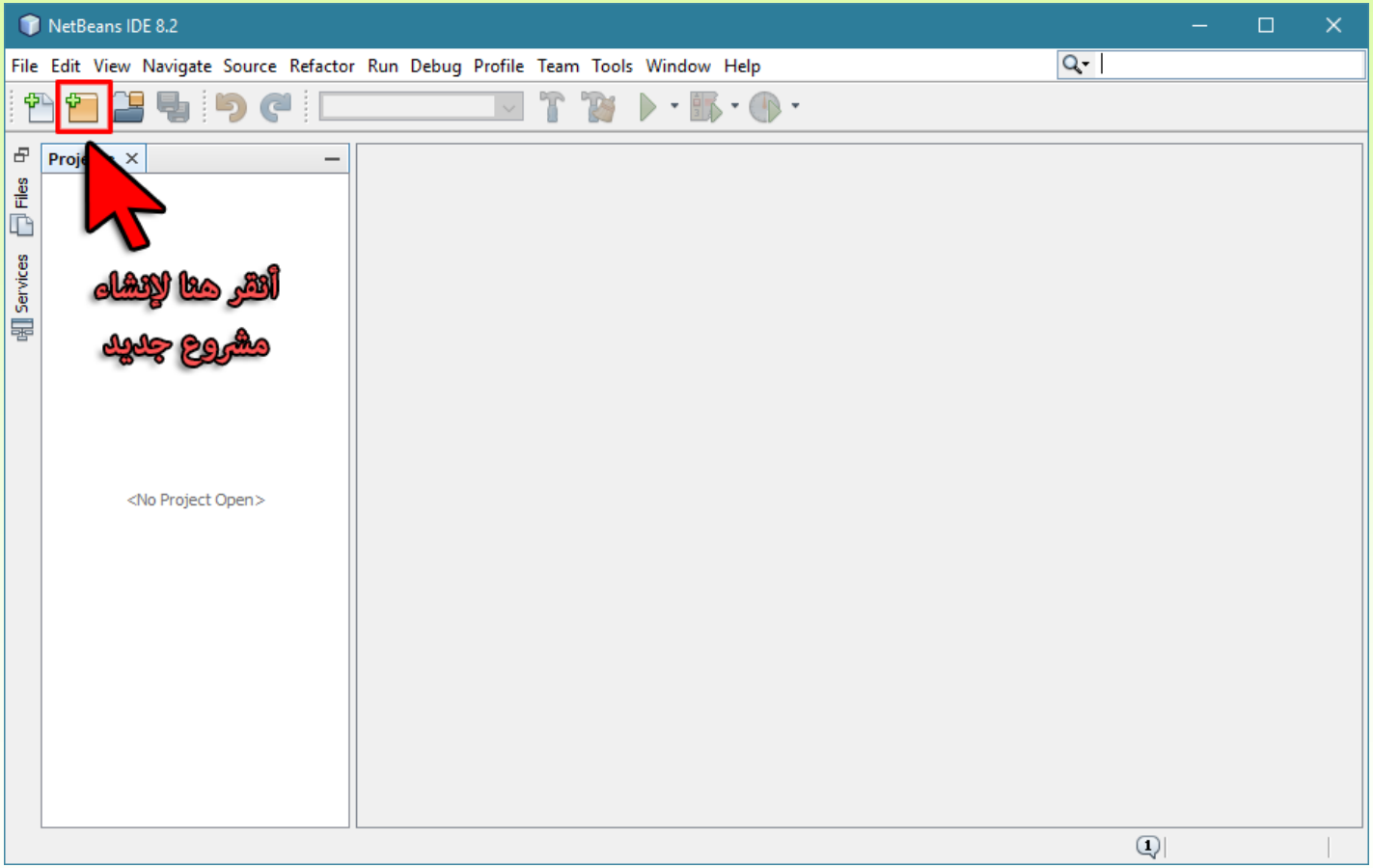
- https://harmash.com/java/java-overview/
- https://www.geeksforgeeks.org/java/
- Lewis & Loftus, Java Software Solutions, Foundations of Program Design,7th Edition
- Bruce Eckel, Thinking in java, Prentice Hall, 4th Edition



Java Essentials أساسيات لغة الجافا

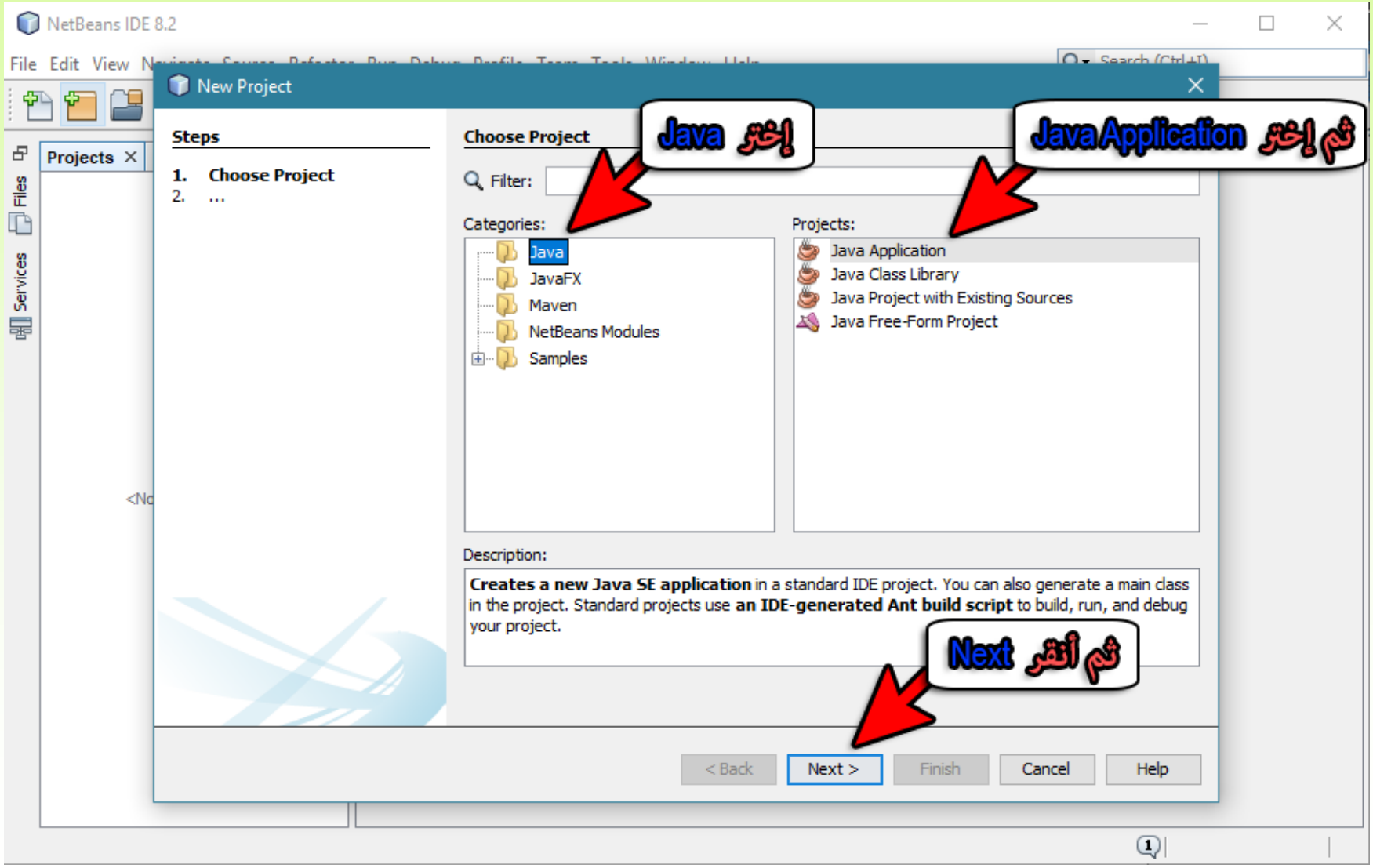
خطوات إنشاء مشروع جديد في الجافا باستخدام الـ NetBeans

1.أنقر على أيقونة **New Project**



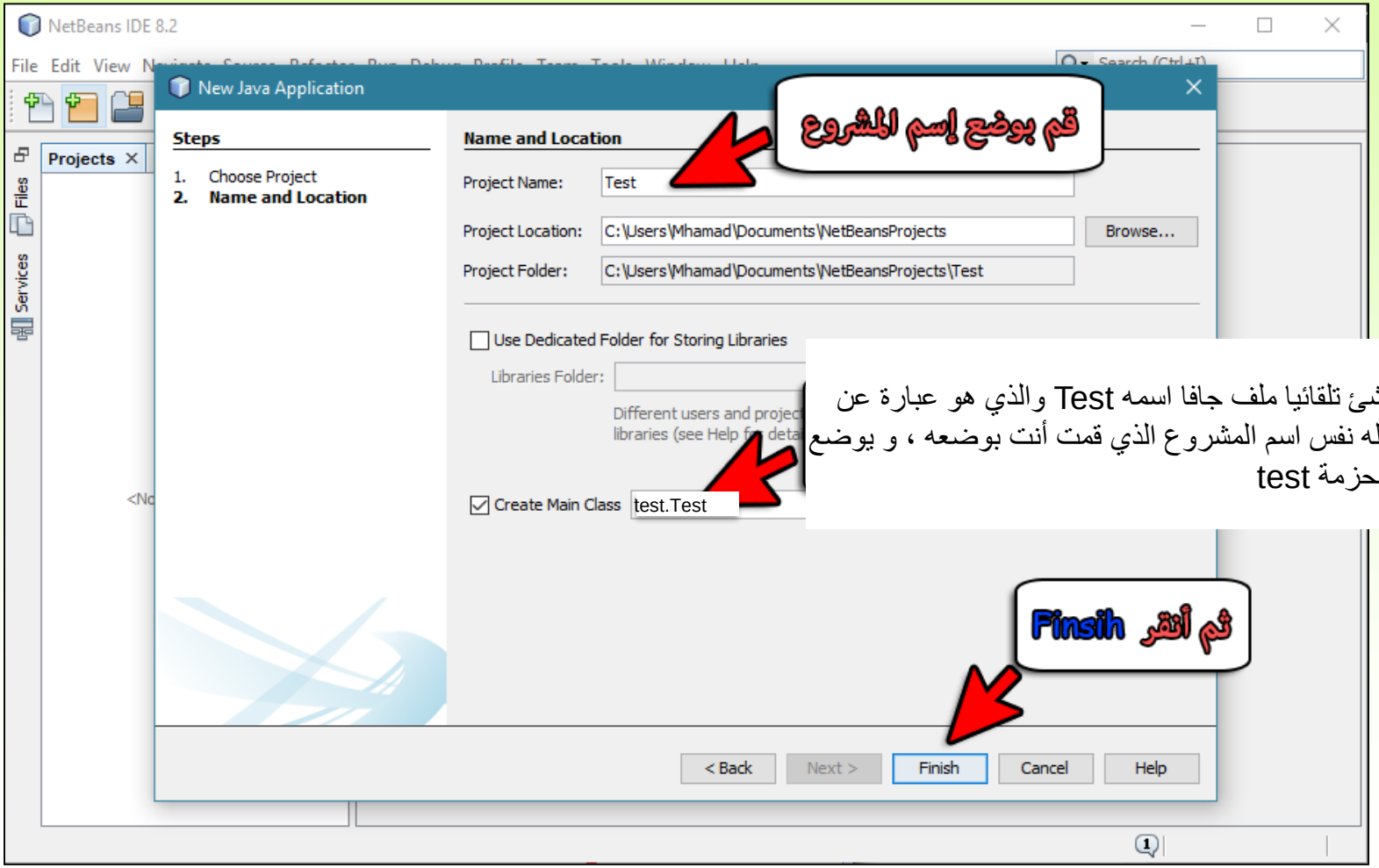
خطوات إنشاء مشروع جديد في الجافا باستخدام الـ NetBeans

2.ثم أنقر على مجلد **Java** ثم مجلد **Java Application** ثم أنقر على الزر **Next** كما في الصورة التالية.



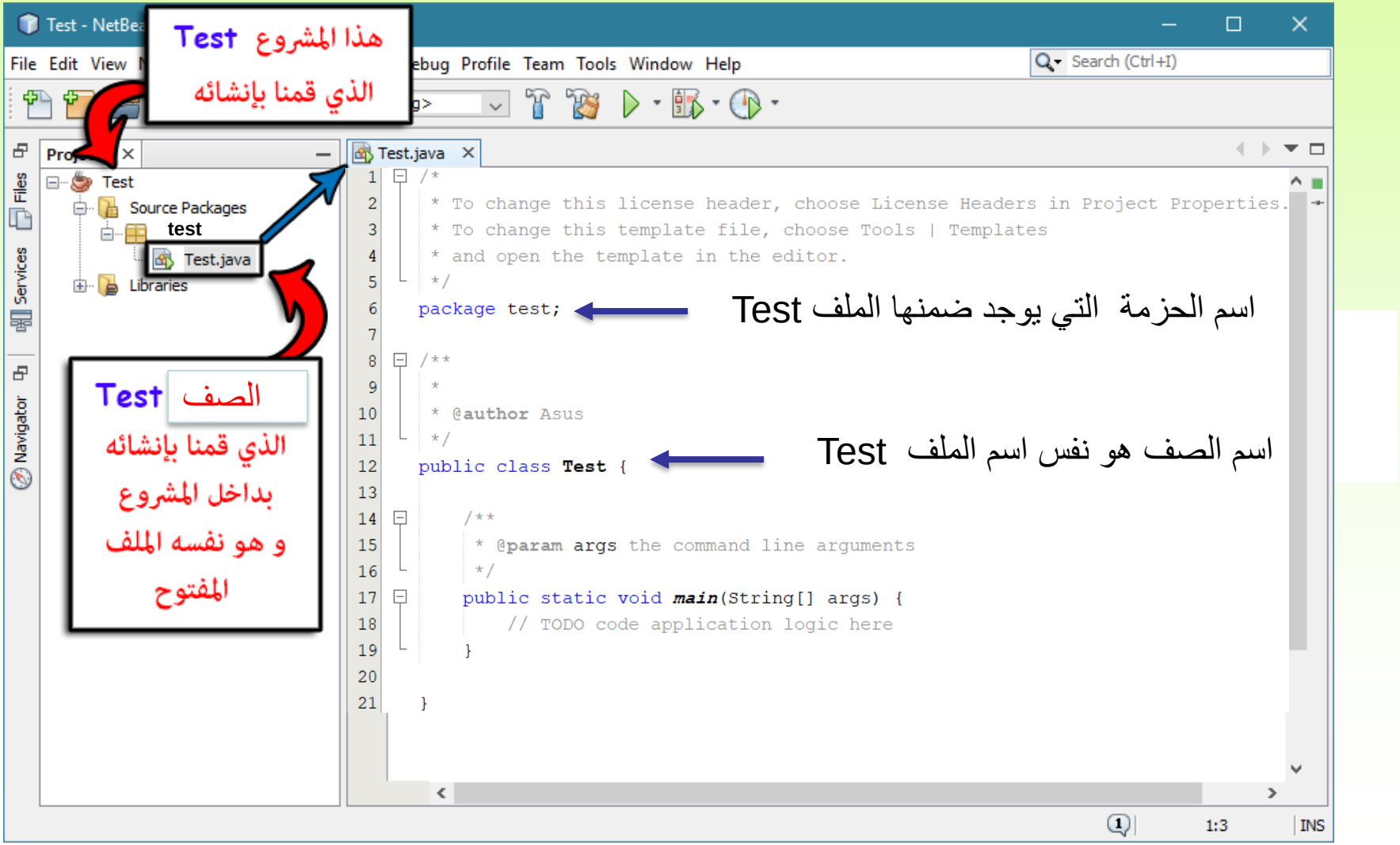
خطوات إنشاء مشروع جديد في الجافا باستخدام الـ NetBeans

3. الآن قم بوضع اسم للمشروع، ثم إضغط **Finish** كما في الصورة التالية.



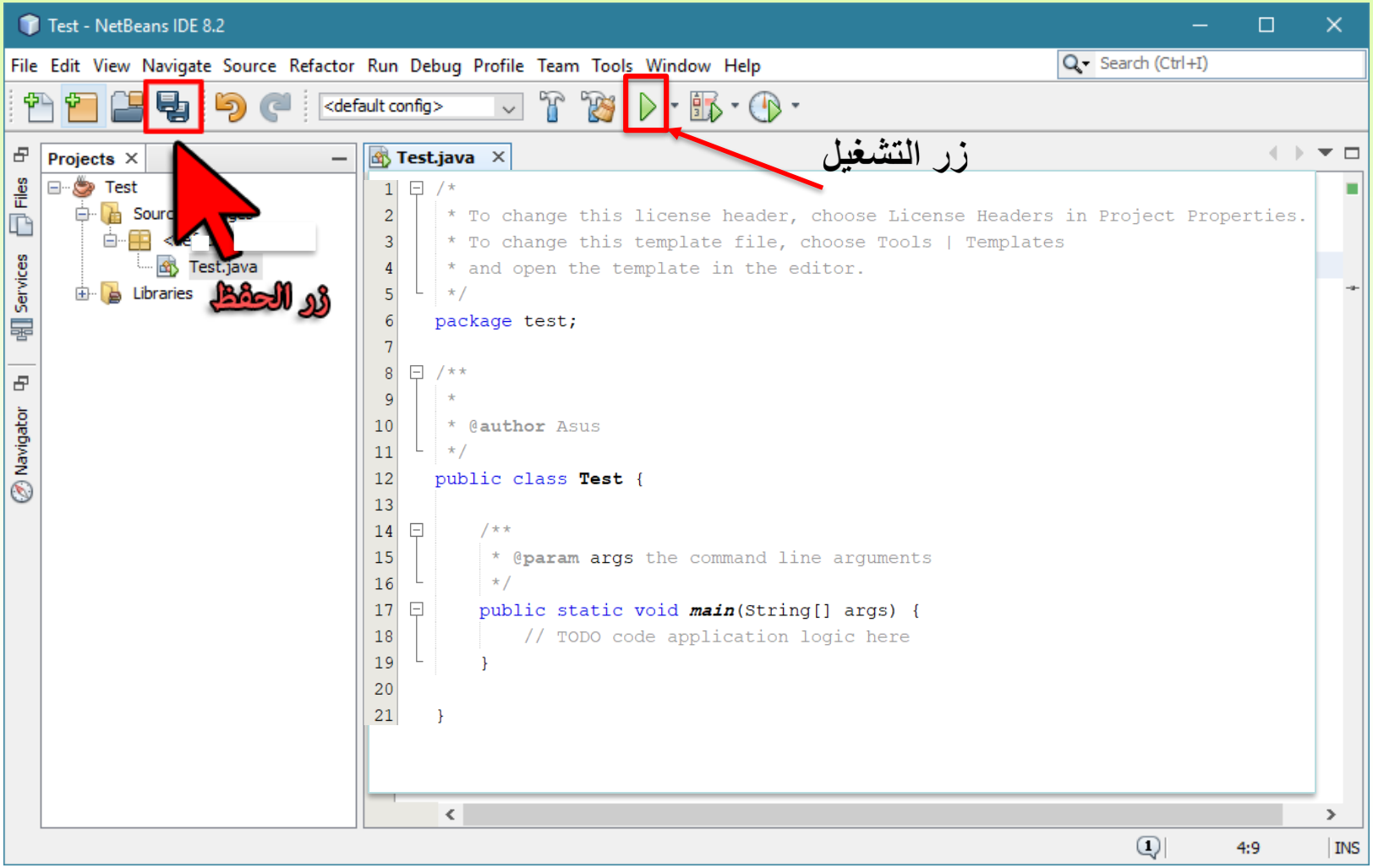
خطوات إنشاء مشروع جديد في الجافا باستخدام الـ NetBeans

4.لاحظ أن المشروع Test يحوي مجلد Source Package و مجلد Libraries
يضم مجلد Source Packages الحزمة test والتي تحوي بدورها جميع ملفات (صفوف) المشروع



خطوات إنشاء مشروع جديد في الجافا باستخدام الـ NetBeans

6. في الأخير إضغط على أيقونة **الحفظ** لتحفظ جميع التعديلات التي قمت بإجرائها، و أيقونة التشغيل لتنفيذ المشروع.

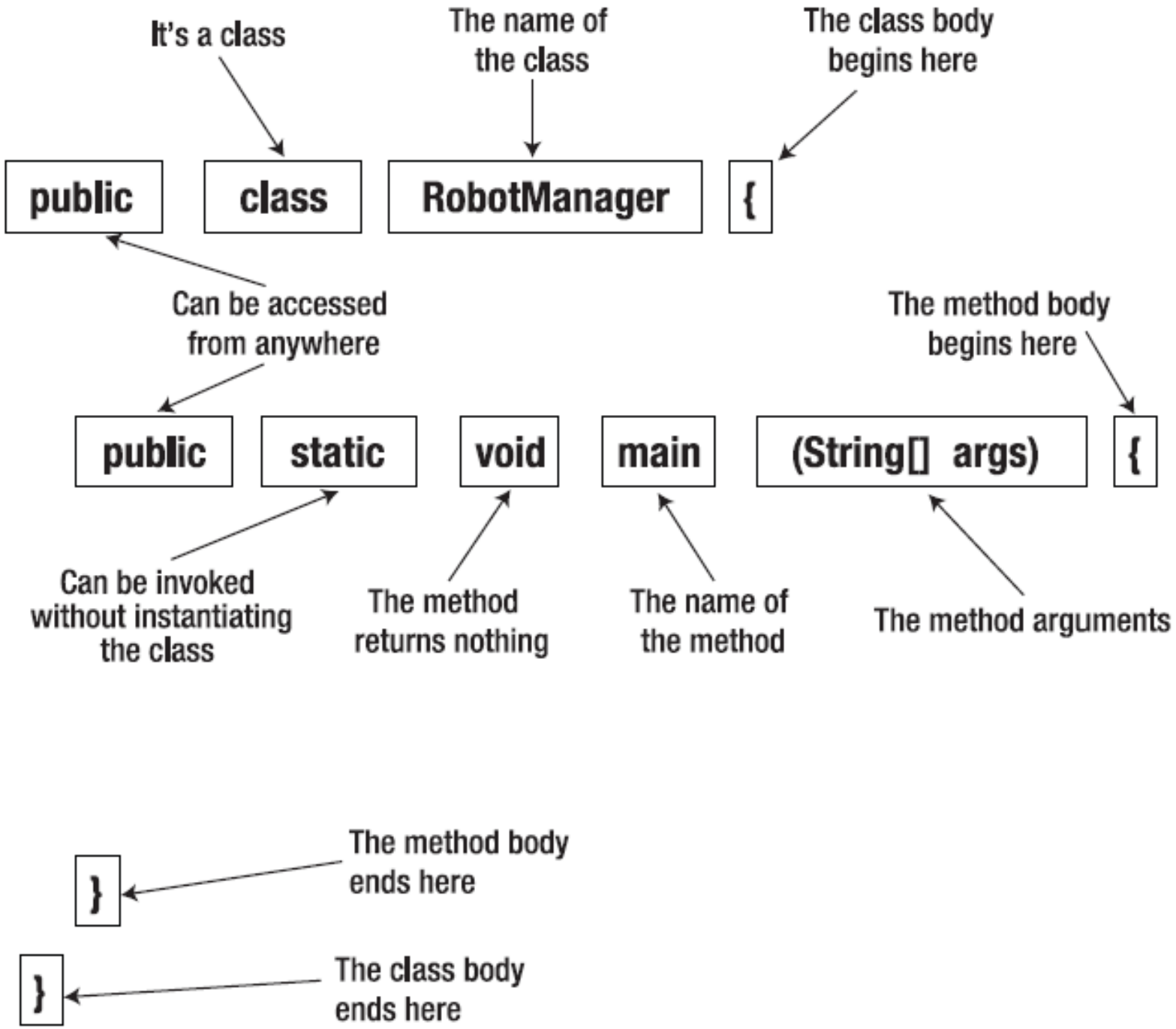


كتابة برنامج يطبع على الشاشة العبارة Welcome to java world

```
package test;
public class Test {
public static void main(String[] args) {
   System.out.print ("Welcome to java world");
    }
}
```

سنحصل على النتيجة التالية عند التشغيل. Welcome to java world

- **الجافا هي لغة برمجة غرضية التوجه**. هذا يعني أن كل مشروع في الجافا قد يتألف من ملف جافا واحد أو أكثر له الامتداد java و الملف في الوقت نفسه عبارة عن صف class. توضع هذه الملفات (الصفوف) ضمن حزمة واحدة و التي في مثالنا هنا test
- السطر package test; يدل على الحزمة الموجود ضمنها الملف Test
- في السطر public class Test **الكلمة class هي كلمة مفتاحية** يليها اسم الصف Test. أما كلمة public فتعني أن محدد الوصول لهذا الصف عام، أي يمكن رؤيته (الوصول له) من صفوف موجودة في حزم أخرى.
- قد يتألف الصف من طريقة واحدة أو أكثر
- السطر public static void main(String[] args)
  - الكلمة main تدل على أن الطريقة هي طريقة رئيسية في هذا الصف، أي يبدأ التنفيذ منها. وليس من الضروري أن يعرف الصف طريقة رئيسية، و في هذه الحالة يصبح غير قابل للتنفيذ لوحده ، و لكن يمكن استخدامه من خلال تمرير التحكم إليه من صف رئيسي.
  - الكلمة void تعني أن الطريقة لا ترجع أي قيمة
  - الكلمة static تعني أن الطريقة يمكن استخدامها دون الحاجة إلى إنشاء غرض من الصف
  - العبارة String[] args تعني أن وسيط الدالة اسمه args من النوع متجه array من نوع String


It's a class
The name of the class
The class body begins here
public
class
RobotManager
{
Can be accessed from anywhere
The method body begins here
public
static
void
main
(String[] args)
{
Can be invoked without instantiating the class
The method returns nothing
The name of the method
The method arguments
}
The method body ends here
}
The class body ends here

تعليمة الطباعة

الشكل العام لتعليمة الطباعة

```
System.out.print(params);
```

- تقوم بطباعة القيم params على الشاشة، حيث أن params هي مجموعة من القيم أو المتحولات التي نريد طباعتها مربوطة مع بعضها بإشارة "+".
- إن التعليمة System.out.print تطبع القيم و تبقي المؤشر في السطر نفسه
- لنقل المؤشر إلى السطر التالي بعد عملية الطباعة نستخدم التعليمة System.out.println
- **بعض الموجهات التي تتحكم بشكل الطباعة**

Escape Sequence	Meaning
\b	Backspace يحذف محرف واحد للخلف
\t	tab تفصل بين القيم المطبوعة بثمانية فراغات
\n	newline الانتقال لسطر جديد
\r	carriage return يحذف كل المحارف السابقة الموجودة على نفس السطر
\"	double quote
\'	single quote
\\	backslash

```
//********************************************************************
//  Countdown.java       Author: Lewis/Loftus
//
//  Demonstrates the difference between print and println.
//********************************************************************

public class Countdown
{
   //-----------------------------------------------------------------
   //  Prints two lines of output representing a rocket countdown.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      System.out.print ("Three... ");
      System.out.print ("Two... ");
      System.out.print ("One... ");
      System.out.print ("Zero... ");
      System.out.println ("Liftoff!");  // appears on first output line
      System.out.println ("Houston, we have a problem.");
   }
}
```

Output

```
Three... Two... One... Zero... Liftoff!
Houston, we have a problem.
```

```
public class Facts
{
   //-----------------------------------------------------------------
   //  Prints various facts.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      // Strings can be concatenated into one long string
      System.out.println ("We present the following facts for your "
                          + "extracurricular edification:");

      System.out.println ();

      // A string can contain numeric digits
      System.out.println ("Letters in the Hawaiian alphabet: 12");
      // A numeric value can be concatenated to a string
      System.out.println ("Dialing code for Antarctica: " + 672);
      System.out.println ("Year in which Leonardo da Vinci invented " +
                          "the parachute: " + 1515);
       System.out.println ("Speed of ketchup: " + 40 + " km per year");
     }
}
```

Output

```
We present the following facts for your extracurricular edification:

Letters in the Hawaiian alphabet: 12
Dialing code for Antarctica: 672
Year in which Leonardo da Vinci invented the parachute: 1515
Speed of ketchup: 40 km per year
```

```java
//********************************************************************
//  Addition.java       Author: Lewis/Loftus
//
//  Demonstrates the difference between the addition and string
//  concatenation operators.
//********************************************************************

public class Addition
{
   //-----------------------------------------------------------------
   //  Concatenates and adds two numbers and prints the results.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      System.out.println ("24 and 45 concatenated: " + 24 + 45);

      System.out.println ("24 and 45 added: " + (24 + 45));
   }
}
```

Output

```
24 and 45 concatenated: 2445
24 and 45 added: 69
```

Quick Check

What output is produced by the following?

```
System.out.println ("X: " + 25);
System.out.println ("Y: " + (15 + 50));
System.out.println ("Z: " + 300 + 50);
```

```
X: 25
Y: 65
Z: 30050
```

```
public class Roses
{
   //-----------------------------------------------------------------
   //  Prints a poem (of sorts) on multiple lines.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      System.out.println ("Roses are red,\n\tViolets are blue,\n" +
         "Sugar is sweet,\n\tBut I have \"commitment issues\",\n\t" +
         "So I'd rather just be friends\n\tAt this point in our " +
         "relationship.");
   }
}
```

Output

```
Roses are red,
      Violets are blue,
Sugar is sweet,
      But I have "commitment issues",
      So I'd rather just be friends
      At this point in our relationship.
```

Reserved Words الكلمات المحجوزة

- الكلمات المحجوزة تملك معنى خاصاً و تقوم بأعمال محددة في لغة الجافا. لذلك لا تسمح هذه اللغة باستخدام الكلمات المحجوزة كجزء من أسماء متحولات و غيرها.

abstract	else	interface	switch
assert	enum	long	synchronized
boolean	extends	native	this
break	false	new	throw
byte	final	null	throws
case	finally	package	transient
catch	float	private	true
char	for	protected	try
class	goto	public	void
const	if	return	volatile
continue	implements	short	while
default	import	static	
do	instanceof	strictfp	
double	int	super	

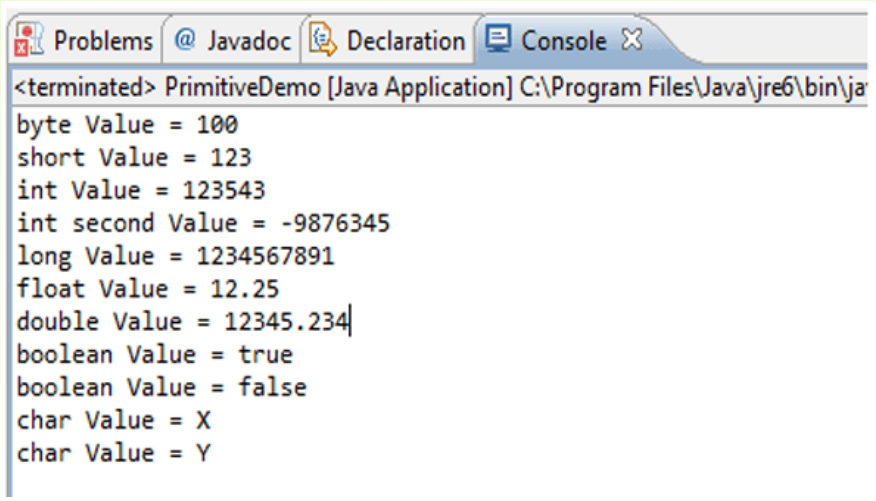
Primitive Types الأنواع البسيطة

- يوجد في الجافا ثمانية أنواع بسيطة

Data Type	Size in Bits	Range of Values	Signed/Unsigned
boolean	1	true or false	NA
byte	8	-2^7 to $2^7 - 1$	Signed
short	16	-2^{15} to $2^{16} - 1$	Signed
char	16	0 to $2^{15} - 1$	Unsigned
int	32	-2^{31} to $2^{31} - 1$	Signed
float	32	-2^{31} to $2^{31} - 1$	Signed
double	64	-2^{63} to $2^{63} - 1$	Signed
long	64	-2^{63} to $2^{63} - 1$	Signed

أمثلة

```
package primitive;
public class PrimitiveDemo {
public static void main(String[] args) {
  byte b =100;
  short s =123;
  int v = 123543;
  int calc = -9876345;
  long amountVal = 1234567891;
  float intrestRate = 12.25f;
  double sineVal = 12345.234;
  boolean flag = true;
  boolean val = false;
  char ch1 = 88; // code for X
  char ch2 = 'Y';
  System.out.println("byte Value = "+ b);
  System.out.println("short Value = "+ s);
  System.out.println("int Value = "+ v);
  System.out.println("int second Value = "+ calc);
  System.out.println("long Value = "+ amountVal);
  System.out.println("float Value = "+ intrestRate);
  System.out.println("double Value = "+ sineVal);
  System.out.println("boolean Value = "+ flag);
  System.out.println("boolean Value = "+ val);
  System.out.println("char Value = "+ ch1);
  System.out.println("char Value = "+ ch2);
}
}
```



النوع الشريطي String

- النوع الشريطي ليس نوعا بسيطا (هو صف class) إلا أنه و لكثرة استخدامه يمكن التعامل معه بنفس الطريقة التي نتعامل بها مع الأنواع البسيطة.
- String Name=“Hello World”;
- لاحظ أنه يجب وضع القيمة المسندة بين إشارتي تنصيص(" ")

Identifiers المعرفات

- هي أسماء تعطى **للمتحولات و الطرق و الصفوف**.
- يمكن تشكيل هذه الأسماء من حروف و أرقام و بعض الإشارات الخاصة.
- من الإشارات المسموحة هي: إشارة (_)، و إشارة الدولار ($).
- لا يمكن للمعرفات أن تبدأ برقم.
- الجافا حساسة لحالة الأحرف، فمثلا: Total, total, TOTAL هي معرفات مختلفة.
- لا يمكن استخدام الكلمات المحجوزة كمعرفات.

Quick Check

Which of the following are valid Java identifiers?

`grade`	Valid
`quizGrade`	Valid
`NetworkConnection`	Valid
`frame2`	Valid
`3rdTestScore`	Invalid – cannot begin with a digit
`MAXIMUM`	Valid
`MIN_CAPACITY`	Valid
`student#`	Invalid – cannot contain the '#' character
`Shelves1&2`	Invalid – cannot contain the '&' character

Variables المتحولات

- هو اسم لمكان في الذاكرة يمكن أن تخزن فيه بيانات
- يتم تحديد نوع لأي متحول نريد استخدامه، فمثلا:

int x; (x متحول من النوع الصحيح)

- يمكن التصريح عن عدة متحولات في عبارة واحدة

```
int count, temp, result;
```

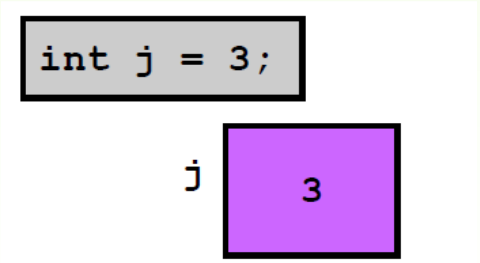
- يمكن وضع قيم ابتدائية للمتحولات أثناء التصريح عنها باستخدام عملية الاسناد (=)

```
int age = 30;
```

حيث يقوم مترجم لغة الجافا بحجز مكان في الذاكرة اسمه age يستطيع تخزين عدد صحيح، و يقوم بوضع القيمة 30 فيه.

- يمكن تغيير قيمة المتحول أثناء تنفيذ البرنامج، أي يمكن تغيير محتوى المتحول (القيمة المخزنة فيه) في أي مكان في البرنامج.

```
int x=10;
x=4;
```



```
//********************************************************************
//  PianoKeys.java       Author: Lewis/Loftus
//
//  Demonstrates the declaration, initialization, and use of an
//  integer variable.
//********************************************************************

public class PianoKeys
{
   //-----------------------------------------------------------------
   //  Prints the number of keys on a piano.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      int keys = 88;
      System.out.println ("A piano has " + keys + " keys.");
   }
}
```

Output

```
A piano has 88 keys.
```

الثوابت Constants

- الثابت هو اسم لمكان في الذاكرة يمكن أن يخزن قيمة من نوع معين، و تبقى هذه **القيمة ثابته (لا يمكن تغييرها) طالما أن البرنامج في حالة تنفيذ**.

- **الفرق بين المتحول و الثابت** هو أنه يمكن للمتحول أخذ قيم مختلفة أثناء تنفيذ البرنامج، بينما يأخذ الثابت قيمة واحدة تسند إليه أثناء التصريح عنه (قبل تنفيذ البرنامج) و لا يجوز تغييرها أثناء التنفيذ.

- يصرح عن الثابت باستخدام الكلمة المحجوزة final

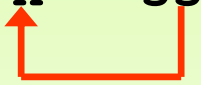
- تصرح العبارة التالية عن ثابت PI و تسند إليه القيمة المحددة

```
final double PI=3.14159;
```

Assignment الاسناد

- تمكن عملية الاسناد من إعطاء قيم للمتحولات أو الثوابت
- معامل الاسناد هو =
- يمكن اسناد قيمة 55 إلى متحول صحيح x بالشكل:

```
int x = 55;
```



ركز جيداً على المفاهيم التالية لكي تعرف كيف تصلح الأخطاء التي قد تتعرض لها عند كتابة الكود.

- عملية تعريف متغير بدون إعطائه قيمة تسمى **Declaration**
- عملية إعطاء قيمة لمتغير تم إنشاؤه سابقاً تسمى **Assigning**
- عملية تعريف متغير و إعطائه قيمة مباشرةً عند إنشائه تسمى **Initialisation**

```java
//********************************************************************
//  Geometry.java       Author: Lewis/Loftus
//
//  Demonstrates the use of an assignment statement to change the
//  value stored in a variable.
//********************************************************************

public class Geometry
{
   //-----------------------------------------------------------------
   //  Prints the number of sides of several geometric shapes.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      int sides = 7;  // declaration with initialization
      System.out.println ("A heptagon has " + sides + " sides.");

      sides = 10;  // assignment statement
      System.out.println ("A decagon has " + sides + " sides.");

      sides = 12;
      System.out.println ("A dodecagon has " + sides + " sides.");
   }
}
```

Output

```
A heptagon has 7 sides.
A decagon has 10 sides.
a dodecagon has 12 sides.
```

العوامل المستخدمة في العمليات الحسابية Arithmetic Operators

- تستخدم لتقييم بعض التعابير الرياضية، وهي الجمع و الطرح و الضرب و القسمة و الباقي

Addition	+
Subtraction	−
Multiplication	*
Division	/
Remainder	%

- يعتمد نوع القيمة المعادة على نوع المتحولات الداخلة في العملية. فمثلا إذا كان نوع المتحولين x و y صحيحا، فإن القيمة المعادة من التعبير x+y هي قيمة من النوع الصحيح.
- يكفي أن يكون نوع أحد المتحولين حقيقياً (double أو float) لتكون القيمة المعادة من النوع الحقيقي.

أمثلة

- إذا كان كلا العددين في عملية القسمة من النوع الصحيح، فإن القيمة المعادة تكون من النوع الصحيح.
- القيمة المعادة نتيجة القسمة التالية 5/2 هي 2
- يكفي أن يكون أحد العددين حقيقياً ليكون ناتج القسمة حقيقياً
- القيمة المعادة نتيجة القسمة التالية 5.0/2 هي 2.5

Quick Check

What are the results of the following expressions?

```
   12 / 2  =  6
12.0 / 2.0 =  6.0
   10 / 4  =  2
  10 / 4.0 =  2.5
   4 / 10  =  0
  4.0 / 10 =  0.4
   12 % 3  =  0
   10 % 3  =  1
   3 % 10  =  3
```

ادخال قيم المتحولات (المتغيرات) من لوحة المفاتيح

- لكي يتمكن المستخدم من إدخال قيمة لمتحول (لمتغير)عن طريق لوحة المفاتيح فإنه يجب:
  - استدعاء الصف Scanner و الموجود في الحزمة (المكتبة) : java.util وذلك عن طريق كتابة
    import java.util.Scanner; أو **import java.util.*;**
  - ومن ثم انشاء كائن (غرض) Object من الصف Scanner

```
Scanner variable_name=new Scanner(System.in);
```

اكتب برنامجا يطلب من المستخدم ادخال اسمه

```
import java.util.Scanner;  // Import the Scanner class
class MyClass {
  public static void main(String[] args) {
    Scanner myObj = new Scanner(System.in);  // Create a Scanner object
    System.out.println("Enter username");
    String userName = myObj.nextLine();  // Read user input
    System.out.println("Username is: " + userName);  // Output user input
  }
}
```

نتيجة التنفيذ

```
Enter username
Reema
Username is: Reema
```

اكتب برنامجا يطلب من المستخدم ادخال اسمه و عمره و راتبه

```
class MyClass {
  public static void main(String[] args) {
    Scanner myObj=new Scanner(System.in);
    System.out.println("Enter name, age and salary");
    String name=myObj.nextLine();
    int age=myObj.nextInt();
    double salary=myObj.nextDouble();
    System.out.println("Name: "+name);
    System.out.println("Age: "+age);
    System.out.println("Salary: "+salary);
   }
}
```

نتيجة التنفيذ

```
Enter name, age and salary
Sami
30
40000
Name: Sami
Age: 30
Salary: 40000.0
```

اكتب برنامجا يطلب من المستخدم ادخال عددين صحيحين و حساب جدائهما

```
import java.util.*;   // so that I can use Scanner

public class ScannerMultiply {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.print("Please type two numbers: ");
        int num1 = console.nextInt();
        int num2 = console.nextInt();
        int product = num1 * num2;
        System.out.println("The product is " + product);
    }
}
```

Output

```
Please type two numbers: 8 6
The product is 48
```

- The `Scanner` can read multiple values from one line.

بعض الطرق التي يوفرها الصف Scanner

Method	Description
`nextInt()`	reads an `int` from the user and returns it
`nextDouble()`	reads a `double` from the user
`next()`	reads a one-word `String` from the user
`nextLine()`	reads a one-*line* `String` from the user

العوامل المستخدمة في المقارنات Logical Comparison

- تقوم بمقارنة متغيرات (متحولات) و إعادة قيمة بوليانية (true أو false)
- انتبه للفرق بين (==) و (=)

Operator	Meaning	Example	Value
==	equals	`1 + 1 == 2`	`true`
!=	does not equal	`3.2 != 2.5`	`true`
<	less than	`10 < 5`	`false`
>	greater than	`10 > 5`	`true`
<=	less than or equal to	`126 <= 100`	`false`
>=	greater than or equal to	`5.0 >= 5.0`	`true`

العوامل التي تستخدم في وضع شروط منطقية Logical Operators

- تسمح بتشكيل شروط مركبة من شروط بسيطة، حيث تسمح بربط شرطين أو أكثر للحصول على شرط جديد مركب.

!	Logical NOT
&&	Logical AND
||	Logical OR

```
if (total < MAX+5 && !found)
                System.out.println ("Processing…");
```

Increment and Decrement Operators
معاملات الزيادة و النقصان

Expression	Operation	Example	Result
x++	Add one after use زيادة بعدية	int x = 10, y; y = x++;	x is 11 y is 10
++x	Add one before use زيادة قبلية	int x=10, y; y=++x	x is 11 y is 11
x--	Subtract one after use نقصان بعدي	int x=10, y; y=x--;	x is 9 y is 10
--x	Subtract one before use نقصان قبلي	int x=10, y; y=3*x+(--x);	x is 9 y is 39

Assignment Operators العمليات الحسابية المختصرة

- There are many assignment operators in Java, including the following:

Operator	Example	Equivalent To
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

العبارة

```
result /= (total-MIN) % num;
```

مكافئة لـ

```
result = result / ((total-MIN) % num);
```

Assignment Operators العمليات الحسابية المختصرة

Operator	Operation	Example	Effect
=	assignment	sum = 5;	sum =5;
+=	addition with assignment	sum += 5;	sum = sum +5;
-=	subtraction with assignment	sum -= 5;	sum = sum –5;
*=	multiplication with assignment	sum *=5;	sum = sum*5;
/=	division with assignment	sum /= 5;	sum = sum/5;

Comments التعليقات

- نصوص توضيحية يكتبها المبرمج ضمن البرنامج بهدف توضيح تعليمات هذا البرنامج
- يهمل المترجم محتويات التعليقات لأنها مجرد نصوص توضيحية
- أشكال التعليقات:

```
// this comment runs to the end of the line

/*  this comment runs to the terminating
    symbol, even across line breaks        */
```

التحويل بين الأنواع Data Conversion

- عندما يتم إسناد قيمة من نوع بيانات معين إلى متحول من نوع بيانات آخر، يقوم مترجم اللغة بإجراء تحويل للقيمة المسندة بحيث تناسب نوع المتحول المسند إليه
- تسمح لغة الجافا بإسناد قيمة **من نوع ضيق إلى متحول من نوع واسع، كإسناد قيمة صحيحة لمتحول حقيقي**
- تسمح لغة الجافا أيضا بإسناد قيمة **من نوع واسع إلى متحول من نوع ضيق، كإسناد قيمة حقيقية إلى متحول صحيح**
- **عملية إسناد قيمة من نوع ضيق إلى متحول من نوع أوسع** هي عملية بسيطة و **لا تحتاج إلى تحويل صريح** لأن النوع الواسع يستطيع تخزين القيمة من النوع الضيق دون أي ضياع للدقة
- يرافق **التحويل من نوع واسع إلى نوع** ضيق ضياع محتمل للدقة، و بالتالي **تشترط لغة الجافا القيام بهذا التحويل بشكل صريح** و إلا يصدر المترجم خطأ
- **لا يحتاج التحويل من نوع ضيق إلى نوع واسع إلى تحويل صريح، بينما يجب القيام بالتحويل الصريح في حالة التحويل من نوع واسع إلى نوع ضيق**

Data Conversion التحويل بين الأنواع

Widening Conversions التحويل من النوع الضيق إلى العريض

From	To
byte	short, int, long, float, or double
short	int, long, float, or double
char	int, long, float, or double
int	long, float, or double
long	float or double
float	double

Narrowing Conversions التحويل من النوع العريض إلى الضيق

From	To
byte	char
short	byte or char
char	byte or short
int	byte, short, or char
long	byte, short, char, or int
float	byte, short, char, int, or long
double	byte, short, char, int, long, or float

Example

```
int dollars = 20;
double money = dollars;
```

- **Only widening conversions can happen via assignment**
- Note that the value or type of `dollars` did not change

```
byte tiny = 1;
int small  =  tiny;
double big = small;
```

Example

- Both widening and narrowing conversions can be accomplished by explicitly casting a value
- To cast, the type is put in parentheses in front of the value being converted

```
int total = 50;
float result = (float) total / 6;
```

- Without the cast, the fractional part of the answer would be lost

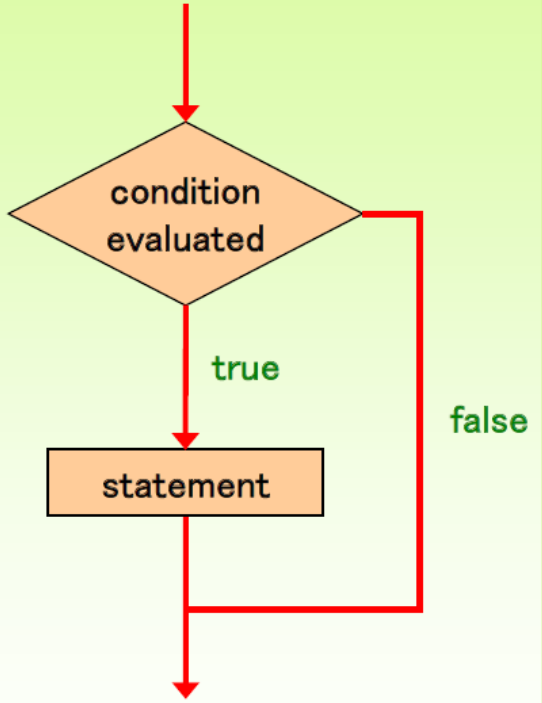
Boolean Expressions

- Specific expressions can be evaluated using truth tables

`total < MAX`	`found`	`!found`	`total < MAX && !found`
false	false	true	false
false	true	false	false
true	false	true	true
true	true	false	false

The if Statement

- الشكل العام للعبارة if



`if` is a Java reserved word

The *`condition`* must be a boolean expression. It must evaluate to either true or false.

```
if ( condition )
   statement;
```

If the *`condition`* is true, the *`statement`* is executed. If it is false, the *`statement`* is skipped.

- إذا كان الشرط condition محققا فإنه يتم تنفيذ التعليمة statement، و إلا يتم ينتقل التنفيذ إلى خارج العبارة if
- **إذا كانت statement أكثر من تعليمة واحدة فيجب وضعها ضمن قوسين.**

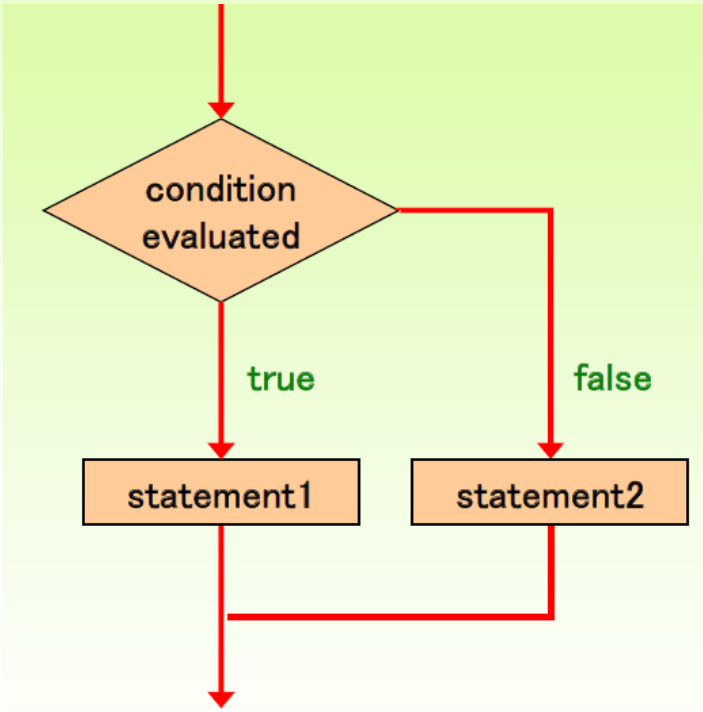
The if-else Statement

- An *else clause* can be added to an `if` statement to make an *if-else statement*

```
if ( condition )
   statement1;
else
   statement2;
```

- If the *condition* is true, *statement1* is executed; if the condition is false, *statement2* is executed
- One or the other will be executed, but not both

إذا كانت statement1 أو statement2 أكثر من تعليمة واحدة فيجب وضعها ضمن قوسين.



```
import java.util.Scanner;
public class MinOfThree
{
   //-----------------------------------------------------------------
   //  Reads three integers from the user and determines the smallest
   //  value.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      int num1, num2, num3, min;
      Scanner scan = new Scanner (System.in);
      System.out.println ("Enter three integers: ");
      num1 = scan.nextInt();
      num2 = scan.nextInt();
      num3 = scan.nextInt();
      if (num1 < num2)
         if (num1 < num3)
            min = num1;
         else
            min = num3;
      else
         if (num2 < num3)
            min = num2;
         else
            min = num3;
      System.out.println("Minimum value: " + min);
   }
}
```

Sample Run

```
Enter three integers:
84   69   90
Minimum value: 69
```

التعليمة الشرطية البسيطة

- لها الصيغة التالية:

condition **?** ***expression1*** **:** ***expression2***

- If the ***condition*** is true, ***expression1*** is evaluated; if it is false, ***expression2*** is evaluated

مثال:

```
larger = ((num1 > num2) ? num1 : num2);
```

If `num1` is greater than `num2`, then `num1` is assigned to `larger`; otherwise, `num2` is assigned to `larger`

اكتب برنامج بلغة جافا يقوم باختبار رقم مدخل من لوحة المفاتيح ويطبع رسالة تعلم المستخدم فيما إذا كان الرقم المدخل موجب أم سالب.

```
package positivenumber;
import java.util.*;
public class PositiveNumber {
    public static void main(String[] args) {
        System.out.println("enter a number");
        Scanner s=new Scanner(System.in);
        int x=s.nextInt();
        if(x>=0)
            System.out.println("The number "+x+ " is positve");
        else
            System.out.println("The number "+x+ " is negative");
    }
}
```

Sample Run

```
enter a number
-30
The number -30 is negative
```

اكتب برنامجا يمكن من ادخال ثلاث علامات، وحساب المعدل و كتابة التقدير

```
package average;
import java.util.Scanner;
public class Average {
public static void main(String[] args) {
       double quiz,mid,f,avg;
        char result=' ';
        Scanner s=new Scanner(System.in);
        System.out.println("enter the quiz mark");
        quiz=s.nextDouble();
        System.out.println("enter the mid mark");
        mid=s.nextDouble();
        System.out.println("enter the final mark");
        f=s.nextDouble();
        avg=(quiz+mid+f)/3;

        if((avg>=90)&&(avg<=100))
            result='A';
        else if ((avg>=70)&&(avg<90))
            result='B';
        else if ((avg>=50)&&(avg<70))
            result='C';
        else
            result='D';
        System.out.println("average is"+avg);
        System.out.println("result is"+result);
    }
}
```

الحلقة التكرارية for-loop

- تستخدم لتكرار تنفيذ مجموعة من التعليمات إما عددا محدد سلفا من المرات أو عددا من المرات حتى يتحقق شرط معين.
- الشكل العام للحلقة for:

The *initialization* is executed once before the loop begins

The *statement* is executed until the *condition* becomes false

```
for ( initialization ; condition ; increment )
    statement;
```

The *increment* portion is executed at the end of each iteration

إذا كانت statement أكثر من تعليمة واحدة فيجب وضعها ضمن قوسين.

الحلقة التكرارية for-loop

- يستخدم الجزء initialization لوضع قيم ابتدائية للمتحولات
- الجزء increment ينفذ عند نهاية كل تكرار
- condition هو شرط توقف الحلقة عن التكرار، و هو شرط منطقي يتم تقييمه في كل تكرار، فإذا كانت نتيجة التقييم true فإن الحلقة تستمر بالتكرار، و إلا يتوقف التكرار و نخرج خارج الحلقة.
- **تعمل الحلقة for كالتالي:** عندما يصل التحكم إلى تعليمة for يتم تنفيذ التعليمة initialization و condition. إذا كان الشرط condition صحيحا يتم تنفيذ جسم الحلقة statement وإلا فإن التحكم يخرج خارج جسم الحلقة. إذا تم تنفيذ جسم الحلقة يعود التنفيذ مجددا إلى الحلقة for و يتم تنفيذ التعليمة increment ثم يجري اختبار الشرط condition فإذا كان صحيحا ينفذ جسم الحلقة مرة أخرى و هكذا.

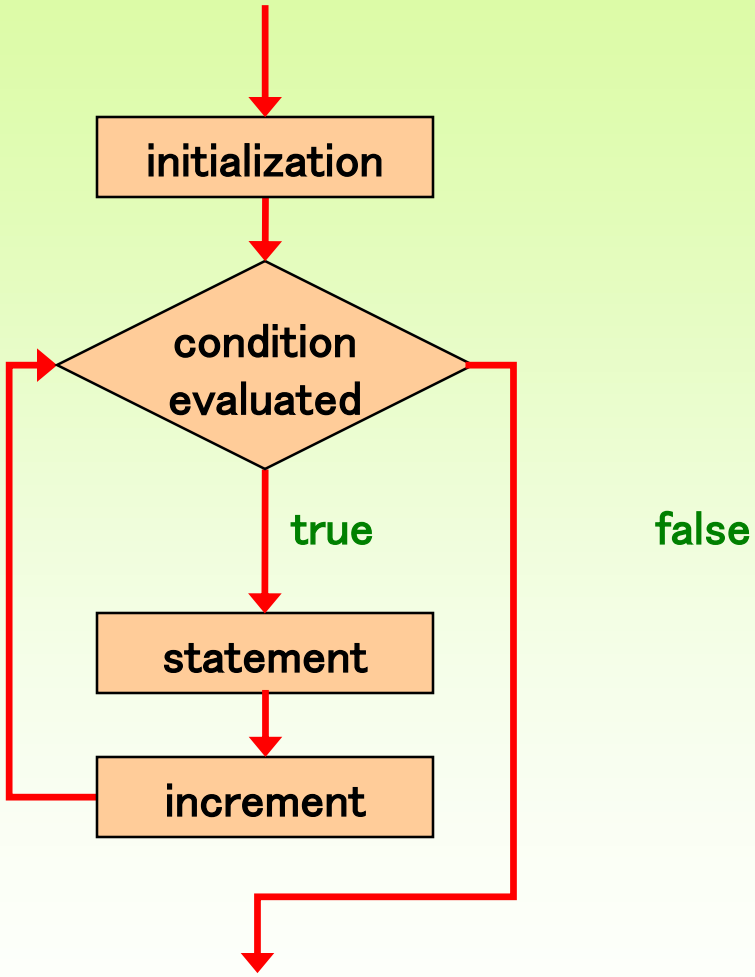
الحلقة التكرارية for-loop

- يمكن حذف أي من مكونات الحلقة initialization أو condition أو increment أو كل هذه المكونات.
- بشكل خاص، إذا تم حذف شرط التوقف condition فإن الحلقة تكرر جسمها عددا غير منته من المرات.

تكرر هذه الحلقة جسمها عددا غير منته من المرات لأنه لايوجد شرط توقف

```
int x=0;
for( ; ; )
   x+=1;
```

Logic of a for loop



ما هو خرج البرنامج التالي:

```
package test1;
public class Test1 {
    public static void main(String[] args) {
       for (int count=1; count <= 5; count++)
            System.out.println(count);
    }
}
```

الناتج

```
1
2
3
4
5
```

ما هو خرج البرنامج التالي:

```
package test;
public class Test
 {
    public static void main(String[] args) {
        for (int num=30; num > 0; num -= 5)
            System.out.println (num);
    }
}
```

الناتج

```
30
25
20
15
10
5
```

اكتب برنامجا يطبع الخرج التالي

Output

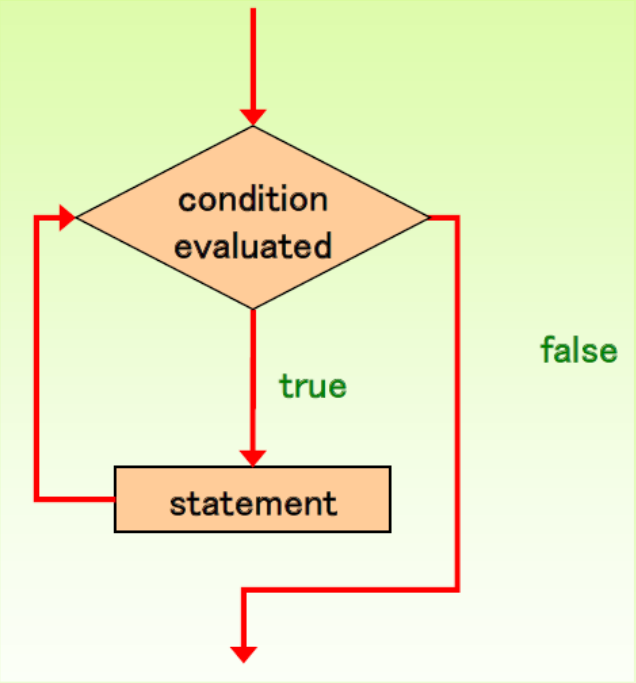
```
*
**
***
****
*****
******
*******
********
*********
**********
```

```
package stars;
public class Stars
{
 public static void main (String[] args)
   {
      final int MAX_ROWS = 10;

      for (int row = 1; row <= MAX_ROWS; row++)
      {
         for (int star = 1; star <= row; star++)
            System.out.print ("*");

         System.out.println();
      }
   }
}
```

الحلقة while



- الشكل العام للحلقة while

```
while ( condition )
  statement;
```

- تسمح الحلقة while تكرار التعليمة statement عددا غير محددا سلفا من المرات طالما أن شرط التنفيذ condition محققا و يتوقف التنفيذ عندما يصبح الشرط condition خاطئا.
- يتم في الحلقة while فحص الشرط condition قبل تنفيذ التعليمة statement
- إذا كان الشرط صحيحا يتم تنفيذ التعليمة statement و يستمر التنفيذ طالما الشرط صحيح
- أما إذا كان الشرط خاطئا فإن التعليمة statement لا ينفذ أبدا و ينتقل التنفيذ إلى ما بعد جسم الحلقة
- ملاحظة: إذا كانت statement أكثر من تعليمة واحدة فيجب وضعها ضمن قوسين.

ما هو خرج البرنامج التالي:

```
package positivenumber;
public class PositiveNumber {
    public static void main(String[] args) {
     int count = 1;
     while (count <= 5){
      System.out.println (count);
      count++;
     }
    }
}
```

الناتج

```
1
2
3
4
5
```

اكتب برنامجا يمكن المستخدم من الاستمرار في ادخال أعداد صحيحة طالما أنها غير مساوية 999 وحساب العدد الأصغر

```
package findmin;
import java.util.*;
public class FindMin {
    public static void main(String[] args) {
     System.out.println("Enter a number= ");
     Scanner s=new Scanner(System.in);
     int num=s.nextInt();
     int min=num;
     while(num!=999)
     {
      System.out.println("Enter a number= ");
      num=s.nextInt();
      if(num<min)
         min=num;
     }
     System.out.println("min= "+min);
    }
}
```

Boundary Conditions are Tricky

```
int count = 1;
while ( count <= 3 ) {
  System.out.println( "count is:" + count );
  count ++;
}
```

This loop will be executed 3 times. The output is:
count is: 1
count is: 2
count is: 3

```
int count = 1;
while ( count < 3 ) {
  System.out.println( "count is:" + count );
  count ++;
}
```

This loop will be executed 2 times. The output is:
count is: 1
count is: 2

Counting downwards by two

```
int count = 6;                                  // count is initialized
while ( count >= 0 ) {                          // count is tested
  System.out.println( "count is:" + count );
  count -= 2;                 // count is changed by 2
}
System.out.println( "Done counting by two's." );
```

This loop will be executed 4 times. The output is:
count is: 6
count is: 4
count is: 2
count is: 0
Done counting by two's.

Infinite Loops

- An example of an infinite loop (مثال على حلقة غير منتهية):

```
int count = 1;
while (count <= 25)
{
    System.out.println (count);
    count-= 1;
}
```

- This loop will continue executing until interrupted (Control-C) or until an underflow error occurs

الحلقة do-while

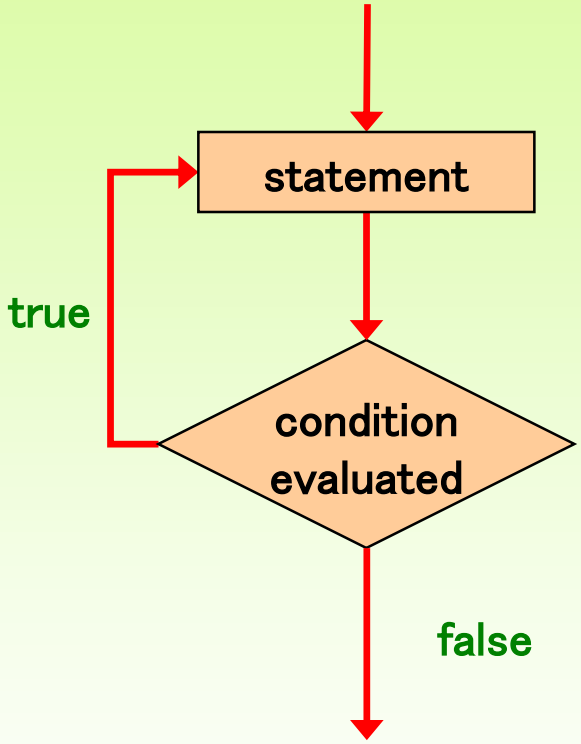
- الشكل العام للحلقة do-while

```
do
  statement;
while (condition);
```

- يتم تنفيذ التعليمة statement أولا ثم يتم اختبار الشرط condition، فإذا كان محققا يكرر تنفيذ التعليمة statement ويستمر التكرار حتى يصبح الشرط condition خاطئا. و عندما يصبح الشرط خاطئا ينتقل التنفيذ إلى ما بعد جسم الحلقة.

- ملاحظة: إذا كانت statement أكثر من تعليمة واحدة فيجب وضعها ضمن قوسين.

Logic of a do Loop



Example

```
package test2;
public class Test2 {
    public static void main(String[] args) {
     int count = 0;
     do
     {
      count++;
      System.out.println (count);
     } while (count < 5);
    }
}
```

الناتج

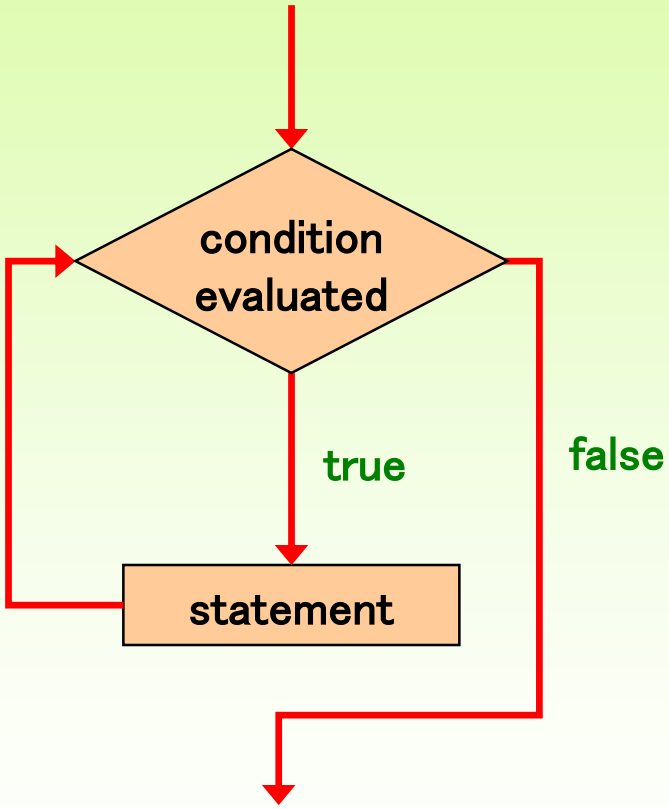
```
1
2
3
4
5
```

اكتب برنامجا يمكن المستخدم من الاستمرار في ادخال عدد صحيح طالما أنه أكبر من الصفر

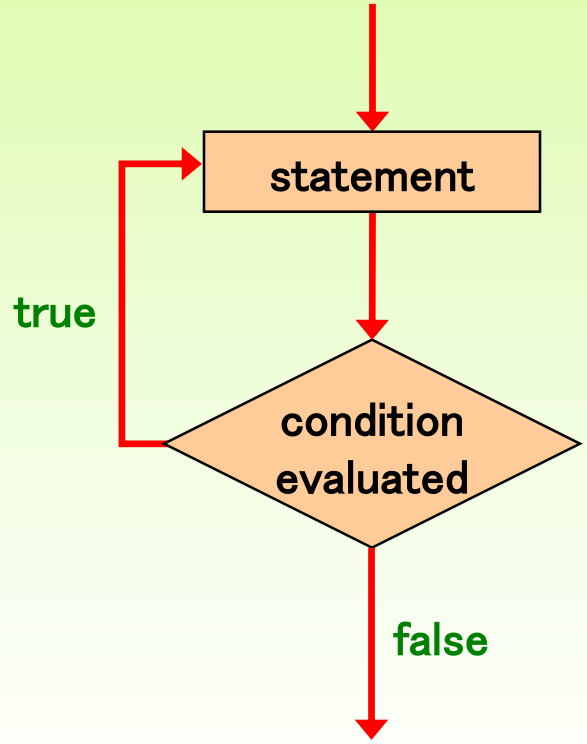
```
package exampleofdo;
import java.util.*;
public class ExampleofDo {
    public static void main(String[] args) {
    int num;
    do
    {
       System.out.println ("Enter a number= ");
       Scanner s=new Scanner(System.in);
       num=s.nextInt();
    }
    while(num>0);
    }
}
```

Comparing while and do

The while Loop



The do Loop



التعليمة break

- هي عبارة عن تعليمة قطع (توقف) و هي تستخدم في الحلقات التكرارية أو في تعليمة switch
- **إذا وجدت و نفذت ضمن حلقة فإنها تقطعها و تخرج منها إلى أول تعليمة تلي الحلقة**
- **إذا وجدت في تعليمة switch فإنها تفيد في الخروج منها و الذهاب إلى أول تعليمة تليها**

```
public class Main {
    public static void main(String[] args) {
       //  هنا قمنا بإنشاء حلقة for تتألف من 10 دورات. في كل دورة تطبع قيمة العداد المستخدم فيها
      for( int i=1; i<=10; i++ )
      {
         // في كل دورة سيتم فحص قيمة العداد و بمجرد أن تصبح تساوي 6 سيتم إيقاف الحلقة نهائياً
         if( i == 6 )
            break;
        System.out.println( i );
      }
    }
}
```

نتيجة التنفيذ:

1
2
3
4
5

التعليمة **break** جعلت الحلقة تتوقف عندما أصبحت قيمة العداد **i** تساوي **6**.

اكتب برنامجا يمكن المستخدم من ادخال عدد صحيح و التحقق فيما إذا كان عددا أوليا أم لا

```
package prime;
import java.util.*;
public class Prime {
    public static void main(String[] args) {
    System.out.println("Enter a number=");
    Scanner s=new Scanner(System.in);
    int num=s.nextInt();
    boolean prime=true;
    for(int i=2; i<num;i++)
        if(num%i==0)
        {
            prime=false;
            break;
        }
    if(prime)
      System.out.println(num + "is a prime number");
    else
      System.out.println(num + "is not a prime number");

    }
}
```

التعليمة continue

- تعليمة تحكم توضع ضمن الحلقات التكرارية
- تقوم بتوقيف التكرار الحالي و تنقل التحكم إلى بداية الحلقة للقيام بتكرار جديد إذا كان شرط الحلقة يسمح بذلك

```
public class Main {
   public static void main(String[] args) {
       //   هنا قمنا بإنشاء حلقة for تتألف من 10 دورات. في كل دورة تطبع قيمة العداد المستخدم فيها
       for( int i=1; i<=10; i++ )
       {
// في كل دورة سيتم فحص قيمة العداد و عندما تصبح تساوي 3 سيتم الإنتقال إلى الدورة التالية في الحلقة بدون تنفيذ أمر الطباعة الموضوع بعدها
           if( i == 3 )
               continue;
           System.out.println( i );
       }
    }
}
```

نتيجة التنفيذ:

1
2
4
5
6
7
8
9
10

التعليمة **continue** جعلت الحلقة تتجاوز الدورة الثالثة، لذلك لم تطبع الرقم **3** لأنها لم تنفذ أمر الطباعة في الدورة الثالثة.

التعليمة switch-case

- تمكن هذه التعليمة من توجيه التحكم (التنفيذ) إلى مقطع برمجي محدد و تنفيذه
- يتم توجيه التحكم بناء على قيمة معينة يتم من خلالها تحديد المقطع البرمجي المطلوب تنفيذه

```
// switch statement
switch(expression)
{
  case value1 :
     Statements
     break; // break is optional

   case value2 :
     Statements
     break; // break is optional

   default :
     Statements
}
```

يجب أن يكون المتحول expression من أحد الأنواع التالية: byte, short, int, char, String

اكتب برنامجا يمكن المستخدم من ادخال رقم يوم من أيام الأسبوع و يعطي اسم اليوم

```
import java.util.Scanner;
public class Days {
public static void main(String[] args) {
        // TODO code application logic here
        System.out.println("Enter day number");
        Scanner s=new Scanner(System.in);
        int day=s.nextInt();
        String name="";
        switch(day){
            case 1:name=“Sunday";break;
            case 2:name=“Monday";break;
            case 3:name=“Tuesday";break;
            case 4:name=“Wednesday";break;
            case 5:name=“Thursday";break;
            case 6:name=“Friday";break;
            case 7:name=“Saturday";break;
            default: System.out.println("enter a number from 1 to 7");
         }
        System.out.println(name);
    }
}
```

```
import java.util.Scanner;
public class GradeReport
{
   //-----------------------------------------
   //  Reads a grade from the user and prints co
   //-----------------------------------------
   public static void main (String[] args)
   {
      int grade, category;
      Scanner scan = new Scanner (System.in);
      System.out.print ("Enter a numeric grade (0 to 100): ");
      grade = scan.nextInt();
      category = grade / 10;
      System.out.print ("That grade is ");
      switch (category)
      {
         case 10:
            System.out.println ("a perfect score. Well done.");
            break;
         case 9:
            System.out.println ("well above average. Excellent.");
            break;
         case 8:
            System.out.println ("above average. Nice job.");
            break;
         case 7:
            System.out.println ("average.");
            break;
         case 6:
            System.out.println ("below average. You should see the");
            System.out.println ("instructor to clarify the material "
                              + "presented in class.");
            break;
         default:
            System.out.println ("not passing.");
      }
   }
```

Sample Run

```
Enter a numeric grade (0 to 100): 91
That grade is well above average.
Excellent.
```

اكتب برنامجا يطلب من المستخدم أن يحزر رقم بين الواحد و المائة يولده الحاسوب عشوائيا

```
package guessnumber;
import java.util.Scanner;
public class GuessNumber {
    public static void main(String[] args) {
        int num,guess,tries=0;
        Scanner s=new Scanner(System.in);
        num=(int)(Math.random()*100)+1;
        do{
            System.out.println("Guess my number");
            guess=s.nextInt();
            if(guess>num)
            {
                System.out.println("Too high try a smaller number!");
                tries++;
            }
            else if(guess<num)
            {
                System.out.println("too small,Try a bigger number!");
                tries++;
            }
            else
            {
                System.out.println("congrats,you win!");
                System.out.println("you tried: "+tries);
            }
        }while(guess!=num);
    }
}
```