الدرس الأول - مقدمة

المحاور

- ما هي البرمجة؟
- لماذا نتعلم البرمجة؟
- ماذا تعني لغة البرمجة؟
- كتابة أول برنامج بلغة جافا

ما هي البرمجة

تعرف البرمجة بأنها عملية كتابة تعليمات وتوجيه جهاز الحاسوب للقيام بمهمة معينة، وهناك عدة لغات برمجة يتتشابه في المنطق والأساس وتختلف في الخصائص والبناء مما يجعل لكل لغة استخدامات معينة تتميز بها.

لماذا نتعلم البرمجة

1. للحصول على دخل مادي مرتفع
2. مهارة التفكير المنطقي وحل المشكلات
3. توفير الوقت والجهد
4. تحويل الأفكار الإبداعية إلى مشاريع إبداعية
5. مواكبة التطور التكنولوجي
6. تنوع المجالات التي تدخل فيها البرمجة

ماذا تعني لغة البرمجة

لغة البرمجة هي عبارة عن مجموعة من الأوامر تكتب وفق قواعد محددة، ثم تُحول إلى لغة يفهمها الحاسوب بواسطة برنامج يدعى ال(compiler).

تطورت لغات البرمجة على عدة مراحل إذ كانت البرامج تُكتب في البداية بلغة الآلة (0و1) مباشرة ثم تطورت إلى لغة تشبه إلى حد ما لغة الإنسان وهي ما عرف بلغة التجميع ثم تطورت لتصبح قريبة جدا من لغة الإنسان وهي ما عرف باللغات عالية المستوى أو لغات الجيل الثالث.

لغة جافا:

لغة جافا هي لغة برمجة عالية المستوى طورتها شركة (Sun Microsystems) ثم اشترتها شركة (oracle) وهي لغة برمجية كائنية التوجه تدخل في العديد من المجالات.

بعض مجالات لغة جافا:

- تطبيقات الويب.
- تطبيقات الهاتف المحمول.
- تطوير تطبيقات سطح المكتب.
- برمجة الآلات والروبوتات.
- معالجات النصوص وجداول البيانات.

كتابة أول برنامج بلغة جافا

أول برنامج سنتعلم كتابته هو برنامج الطباعة وهو مشتهر باسم Hello world.

كل ما عليك هو إنشاء ملف وتسميته Main.java ثم نسخ الكود التالي:

```
public class Main {
  public static void main(String[] args) {
    System.out.println("Hello World!");
  }
}
```

جملة الطباعة في الكود السابق هي:

```
    System.out.println("Hello World!");
```

وعند الضغط على run سيتم طباعة الجملة على الconsole:

```
Hello World!
```

طبعا يمكنك إضافة جمل طباعة كما شئت وتغيير ما بين علامات التنصيص، كما في البرنامج التالي:

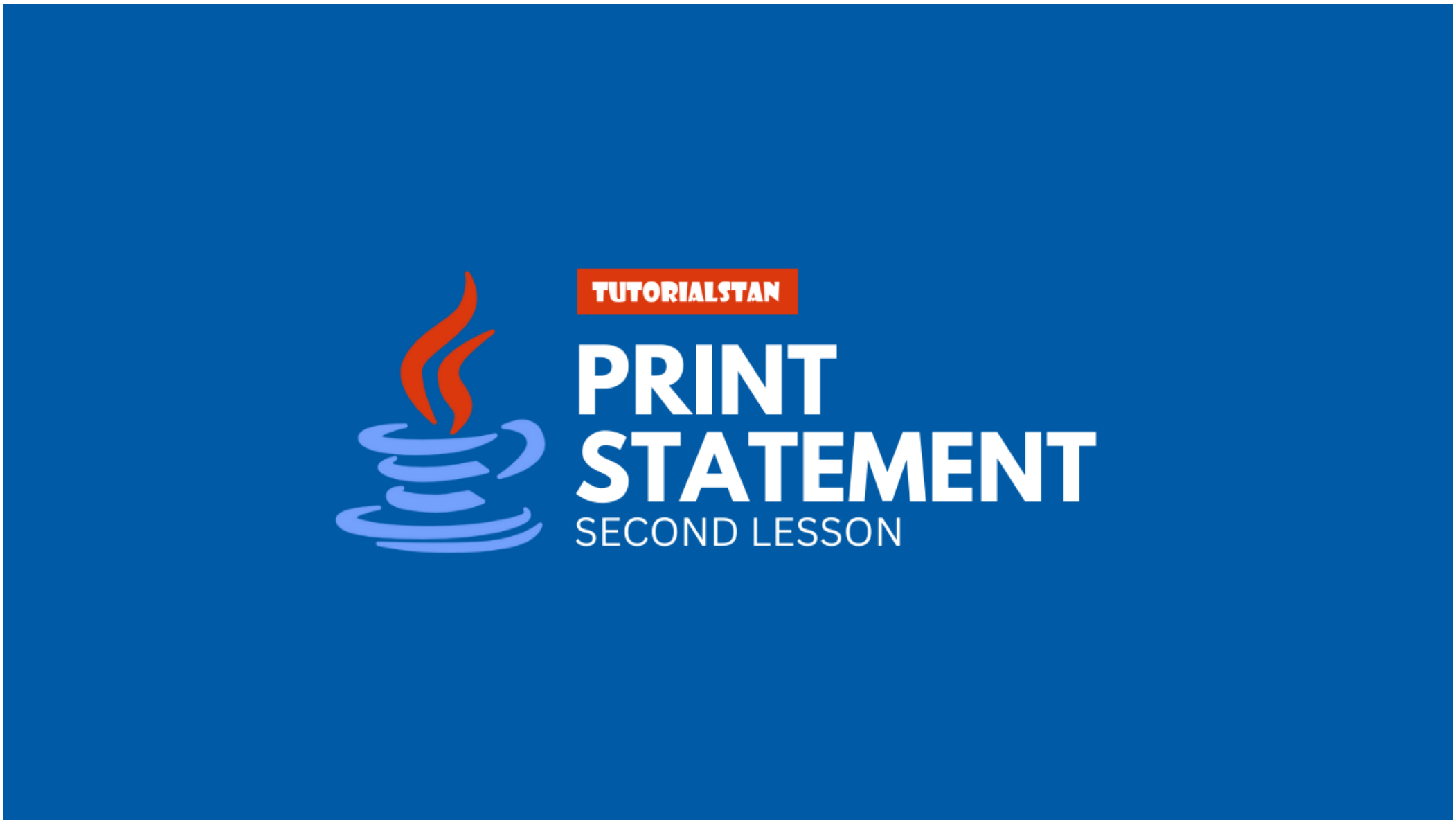
```
public class Main {
  public static void main(String[] args) {
    System.out.println("Hello Developers");
    System.out.println("This is my first program!");
    System.out.println("Coding is Fun :)");
  }
}
```

سيقوم البرنامج بطباعة الآتي:

```
Hello Developers
This is my first program!
Coding is Fun :)
```

وهيك رسميا بنكون خلصنا الدرس الأول وبدأت مسيرة أفخم مبرمج في العالم :)

الدرس الثاني - جملة الطباعة



المحاور

- طباعة النصوص
- طباعة الأرقام
- طباعة العمليات
- التعليقات

طباعة النصوص

تم تغطية هذا الجزء في الدرس الماضي لكن سنتعرف اليوم على الفرق بين جملة print و println.

```
System.out.println("Hi!");
System.out.println("My Name is Suhaib");
```

وعند الضغط على run سيتم طباعة الجملة على الconsole:

```
Hi!
My Name is Suhaib
```

أما في حال استعملنا جملة print سيكون الكود:
أما في حال استعملنا جملة print سيكون الكود:

```
    System.out.print("Hi! ");
    System.out.print("My Name is Suhaib");
```

سيقوم البرنامج بطباعة الآتي:

```
Hi! My Name is Suhaib
```

طباعة الأرقام

لو جربنا طباعة رقم من خلال كتابة نفس الأمر:

```
System.out.println("5");
```

النتيجة:

النتيجة: 5

لكن في الحقيقة الطريقة الصحيحة لطباعة الأرقام هي كتابتها بدون علامات التنصيص

```
System.out.println(5);
```

النتيجة:

النتيجة: 5

جميل، ما الفرق بين الطريقة الأولى والثانية؟ في الحقيقة في أمر الطباعة الأول عامل الرقم كنص أما في الجملة عامله كرقم، ما الفرق؟ سنرى في القسم التالي

العمليات

هل تعلم أنك يمكنك اجراء عمليات حسابية داخل جملة الطباعة! لا تصدقني؟ هيا لنجرب

```
System.out.println(5 + 2);
```

النتيجة:

النتيجة: 7

ماذا لو أجرينا العملية الحسابية داخل علامات التنصيص؟

```
System.out.println("5 + 2");
```

النتيجة:

النتيجة: 5 + 2

أظن الصورة الآن اتضحت لك في الجملة الأولى عامل ما داخل الأقواس كأرقام فأجرى عملية الجمع، أما في الجملة الثانية عاملها كنص بسبب وجود علامات التنصيص. يمكنك تجريب العمليات الحسابية الأخرى كالطرح والضرب والقسمة، أترك هذا الأمر لك.

هناك نوع آخر من العمليات وهو العمليات المنطقية مثل أكبر من > أو أصغر من < أو المساواة == أو أكبر أو يساوي >= أو أصغر أو يساوي <= وهذه العمليات تكون نتيجتها إما صح أو خطأ (true,false)

```
System.out.println(5 > 2);
System.out.println(5 < 2);
System.out.println(5 >= 5);
System.out.println(5 <= 8);
System.out.println(5 == 8);
```

النتيجة:

النتيجة: true false true true false

التعليقات

التعليقات (comments) هي جزء من الكود لا يتم تنفيذه ولا يؤثر على طريقة عمل البرنامج، عادة ما نستخدمه لتوضيح عمل الكود وهي مهمة جدا في البرامج الكبيرة خصوصا تلك التي يشرف عليها أكثر من مبرمج. يتم كتابة التعليق من خلال كتابة // قبل السطر

```
//هذا التعليق لن ينفذ في البرنامج ولن يؤثر على سيره
```

```
// هذا الكود سيطبع إذا ما كانت ال5 أكبر من ال2
System.out.println(5 > 2);

// هذا الكود سيطبع اسمي (:
System.out.print("My Name is Suhaib");
```

النتيجة:

النتيجة: true My Name is Suhaib

الدرس الثاني - المتغيرات

المحاور

- ما هي المتغيرات
- طباعة الأرقام
- طباعة العمليات
- التعليقات
- حجم المتغيرات

ما هو المتغير

المتغير: المتغير هو عنصر يستخدم في البرمجة لتخزين قيمة واستخدامها فيما بعد في البرنامج. وكل متغير له نوع معين ومساحة معينة ويكون تعريف المتغير في الجافا من خلال تحديد نوع المتغير ثم تسمية المتغير ثم اعطاء المتغير قيمة كالتالي:

```
type var = value;
```

مثلا لتعريف رقم في الجافا نستخدم النوع int فإذا سمينا المتغير n وكانت قيمته تساوي 5 سيكون الكود كالتالي:

```
int n = 5;
```

في حال أردنا طباعة المتغير كل ما علينا هو استخدام نفس جملة الطباعة التي تعلمناها في الدرس الماضي:

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

سيقوم البرنامج بطباعة الآتي:

```
5
```

طباعة الأرقام

لو جربنا طباعة رقم من خلال كتابة نفس الأمر:

```
System.out.println("5");
```

النتيجة:

```
5
```

لكن في الحقيقة الطريقة الصحيحة لطباعة الأرقام هي كتابتها بدون علامات التنصيص

```
System.out.println(5);
```

النتيجة:

```
5
```

جميل، ما الفرق بين الطريقة الأولى والثانية؟ في الحقيقة في أمر الطباعة الأول عامل الرقم كنص أما في الجملة عامله كرقم، ما الفرق؟ سنرى في القسم التالي

العمليات

هل تعلم أنك يمكنك اجراء عمليات حسابية داخل جملة الطباعة! لا تصدقني؟ هيا لنجرب

```
System.out.println(5 + 2);
```

النتيجة:

```
7
```

ماذا لو أجرينا العملية الحسابية داخل علامات التنصيص؟

```
System.out.println("5 + 2");
```

النتيجة:

```
5 + 2
```

أظن الصورة الآن اتضحت لك في الجملة الأولى عامل ما داخل الأقواس كأرقام فأجرى عملية الجمع، أما في الجملة الثانية عاملها كنص بسبب وجود علامات التنصيص.
يمكنك تجريب العمليات الحسابية الأخرى كالطرح والضرب والقسمة، أترك هذا الأمر لك.

هناك نوع آخر من العمليات وهو العمليات المنطقية مثل أكبر من > أو أصغر من < أو المساواة == أو أكبر أو يساوي >= أو أصغر أو يساوي <= وهذه العمليات تكون نتيجتها إما صح أو خطأ (true,false)

```
System.out.println(5 > 2);
System.out.println(5 < 2);
System.out.println(5 >= 5);
System.out.println(5 <= 8);
System.out.println(5 == 8);
```

النتيجة:

```
true
false
true
true
false
```

التعليقات

التعليقات (comments) هي جزء من الكود لا يتم تنفيذه ولا يؤثر على طريقة عمل البرنامج، عادة ما نستخدمه لتوضيح عمل الكود وهي مهمة جدا في البرامج الكبيرة خصوصا تلك التي يشرف عليها أكثر من مبرمج. يتم كتابة التعليق من خلال كتابة // قبل السطر

```
//هذا التعليق لن ينفذ في البرنامج ولن يؤثر على سيره

// هذا الكود سيطبع إذا ما كانت ال5 أكبر من ال2
System.out.println(5 > 2);

// هذا الكود سيطبع اسمي :)
System.out.print("My Name is Suhaib");
```

النتيجة:

```
true
My Name is Suhaib
```

حجم المتغيرات

Data Type	Description	Value Range
byte	8-bit integer	-128 to 127
short	16-bit integer	-32,768 to 32,767
int	32-bit integer	-2,147,483,648 to 2,147,483,647
long	64-bit integer	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	32-bit floating point	3.4028235E38 to 1.4E-45
double	64-bit floating point	1.7976931348623157E308 to 4.9E-324
char	16-bit Unicode character	0 to 65,535
boolean	true or false	true or false

الدرس الرابع - المتغيرات (تكملة)

المحاور

- أنواع المتغيرات النصية
- String
- تحويل النوع (type casting)
- Math

أنواع المتغيرات النصية

يوجد نوعان من المتغيرات النصية في لغة الجافا:

1- *char*: يستخدام هذا النوع لتخزين حرف واحد فقط، ويمكن تعريف المتغير من هذا النوع باستخدام الكلمة المحجوزة "char" وتحديد اسم المتغير، وتعيين الحرف المطلوب بين علامتي التنصيص المفردة، كالتالي:

2- *String*: يستخدام هذا النوع لتخزين النصوص، ويمكن تعريف المتغير من هذا النوع باستخدام الكلمة المحجوزة "String" وتحديد اسم المتغير، كالتالي:

```
String s = "Hello World!";
```

```
char c = 'A';
```

Strings

يمكننا معرفة عدد من المعلومات عن المتغرات النصية وإجراء بعض العمليات عليها من خلال الدوال (Methods) الجاهزة مثل:
1- طول النص: length()
2- تحويل النص إلى الأحرف إلى upper case: toUpperCase()
3- تحويل النص إلى الأحرف إلى lower case: toLowerCase()
4- إيجاد موقع الcharفي الString: locate()

```
String txt = "Hi";
System.out.println(txt.length());
```

```
System.out.println(txt.toUpperCase());
System.out.println(txt.toLowerCase());
System.out.println(txt.locate('i'));
```

النتيجة:

```
2
HI
hi
1
```

Type Casting

Type Casting: هو تحويل نوع المتغير لنوع آخر وله نوعان:

1- توسيع المتغير (Widening Casting): وهو تحويل المتغير من نوع أصغر حجما إلى نوع أكبر حجما، مثال:

```
int a = 5;
double b=a;
```

كما لاحظنا لم نحتج لتغيير أي شيء على الكود إلا تعريف المتغير وإعطاؤه قيمة المتغير الأول، والسبب أننا حولنا من متغير نوعه int حجمه bit-32 إل متغير نوعه double حجمه bit-64.

2- تضييق المتغير (Narrowing Casting): وهو تحويل المتغير من نوع أكبر حجما إلى نوع أصغر حجما، مثال:

```
double a=5.2;
int b = (int) a;
```

في هذه الحالة نحتاج أن نعرف العملية يدويا.

Math

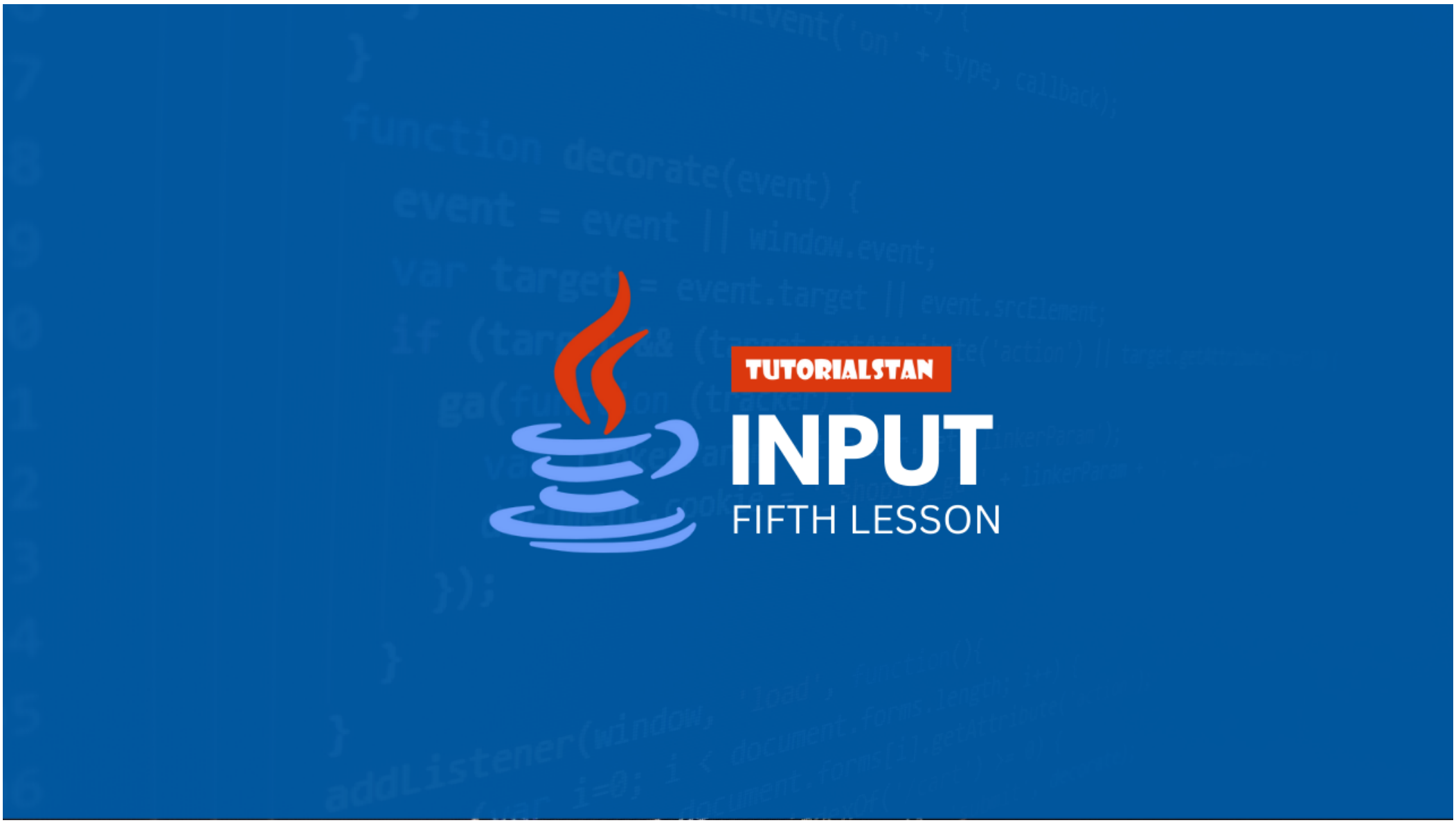
Math هو كلاس يحتوي على عدد من الدوال المهمة التي تجري بعض العمليات الرياضية مثل:

1- Math.max(a,b): إيجاد القيمة الأكبر بين العددين a,b
2- Math.min(a,b): إيجاد القيمة الأصغر بين العددين a,b
3- Math.sqrt(a): إيجاد الجذر للرقم a
4- Math.abs(a): إيجاد القيمة المطلقة للرقم a
5- Math.round(a): تقريب العدد a لأقرب عدد صحيح
6- Math.random(): عدد عشوائي من 0 إلى 1.0
أمثلة:

```
int a= 3;
int b= 10;

System.out.println(Math.max(a,b));
System.out.println(Math.min(a,b));
System.out.println(Math.sqrt(4));
System.out.println(Math.abs(-5));
```

الدرس الخامس - الإدخال



المحاور

- المكتبة البرمجية
- إدراج مكتبة util
- كتابة برنامج لقراءة النصوص
- كتابة برنامج لقراءة الأرقام

المكتبة البرمجية

المكتبة البرمجية: (حسب ويكيبيديا) هي مجموعة من البرامج الفرعية تستخدم لتطوير البرمجيات. تحتوي المكتبات كود «مُساعد» وبيانات توفر خدمات للبرامج المستقلة.
يعني بنقدر نعتبرها مجموعات كودات وكلاسات جاهزة عشان تسهل علينا الشغل بدل ما نكتب الكود من أول وجديد (:
خلينا نتفق انه في بعض المصطلحات بدنا نعرفها حاليا بس راح نفهمها لقدام زي الClass مثلا.

إدراج مكتبة util

المكتبة التي ستساعدنا لقراءة المعلومات من المستخدم اسمها util، ويمكن إدراجها في الكود كالآتي

```
//هنا يتم ادراج المكتبة ونستخدم أمر import
import java.util.*;
//هنا يكون البرنامج الخاص بنا
class Main {
  public static void main(String[] args) {
    //الكود الخاص بك
  }
}
```

الكلاس التي سنستخدمه اسمه Scanner، لذا يمكن استدعاؤه لوحده من المكتبة:

```
//هنا يتم ادراج المكتبة ونستخدم أمر import
import java.util.Scanner;
//هنا يكون البرنامج الخاص بنا
class Main {
  public static void main(String[] args) {
```

```
    الكود الخاص بك//
  }
}
```

في كلا الحالتين الكود سيعمل لكن عندما نستخدم ال* نكون استدعينا جميع الكلاسات في المكتبة وسواء استخدمنا أي الطريقتين فلن يتأثر أداء البرنامج (يعني اعمل اللي بدك اياه بالفترة الحالية).

كتابة برنامج لقراءة النصوص

أولا بعد أن قمنا بإدراج المكتبة سنقوم بتعريف object من الclass Scanner ويتم تعريفه كما نعرف المتغير تماما بهذه الطريقة:

```
//هنا يتم ادراج المكتبة ونستخدم أمر import
import java.util.Scanner;
class Main {
  public static void main(String[] args) {
    // نعرف متغير اسمه reader لنستخدمه في القراءة من المستخدم
    // ويكون نوعه الكلاس Scanner
    // (مش مطلوب منا نفهم اشي حاليا) قيمة المتغير دائما كما هي في الكود
    Scanner reader= new Scanner(System.in);
  }
}
```

الآن سنقوم بتعريف متغيرين من نوع String يقومان بقراءة الاسم الأول واسم العائلة للمستخدم ثم طباعة اسمه كاملا:

```
//هنا يتم ادراج المكتبة ونستخدم أمر import
import java.util.Scanner;
class Main {
  public static void main(String[] args) {
    // نعرف متغير اسمه reader لنستخدمه في القراءة من المستخدم
    Scanner reader= new Scanner(System.in);
    // نطلب من المستحدم كتابة اسمه الأول من خلال جملة طباعة عادية
    System.out.println("Your First Name:");
    // نعرف متغير من نوع String لقراءة اسم المستخدم
    String firstName= reader.nextLine();
    // نطلب من المستحدم كتابة اسم العائلة من خلال جملة طباعة عادية
    System.out.println("Your Last Name:");
    // نعرف متغير من نوع String لقراءة اسم العائلة
    String lastName= reader.nextLine();
    // نرحب به باسمه الكامل
    System.out.println("Welcome "+ firstName +" "+ lastName);
  }
}
```

كتابة برنامج لقراءة الأرقام

لقراءة رقم كل ما علينا هو استبدال المتغير النصي بتغير من نوع مناسب واستبدال قيمة المتغير بالقيمة المناسبة، انظر الأمثلة التالية لقيمة المتغير حسب نوعه:

1- int: reader.nextInt();
2- long: reader.nextLong();
3- short: reader.nextShort();
4- double: reader.nextDouble();
5- float: reader.nextFloat();
6- byte: reader.nextByte();
7- boolean: reader.nextBoolean();

أمثلة:

```
int a= reader.nextInt();
double b= reader.nextDouble();
boolean t= reader.nextBoolean();
```

Project (1)

المحاور

- تدريبات
- Homework

تدريبات

1. اكتب الكود للمعادلات التالية:

- $$5 + y\left(3x + \frac{(x+y)+5}{2+x}\right)$$

- $$\frac{x^2 + y^3 \cdot |z|}{3\sqrt{x} + \sqrt[3]{y}}$$

- جد قيم x في المعادلة التربيعية

2. اكتب برنامج يأخذ معلومات السيارة ثم يعرضها مع مراعاة أخذ المعلومات التالية:
    - اسم صاحب السيارة
    - الشركة المصنعة
    - نوع السيارة
    - موديل السيارة
    - لون السيارة

- عدد الكيلوات التي قطعتها السيارة
- فحص أداء السيارة (نسبة مئوية)
- سعر السيارة الجديدة من هذا النوع
- سعر السيارة

Homework

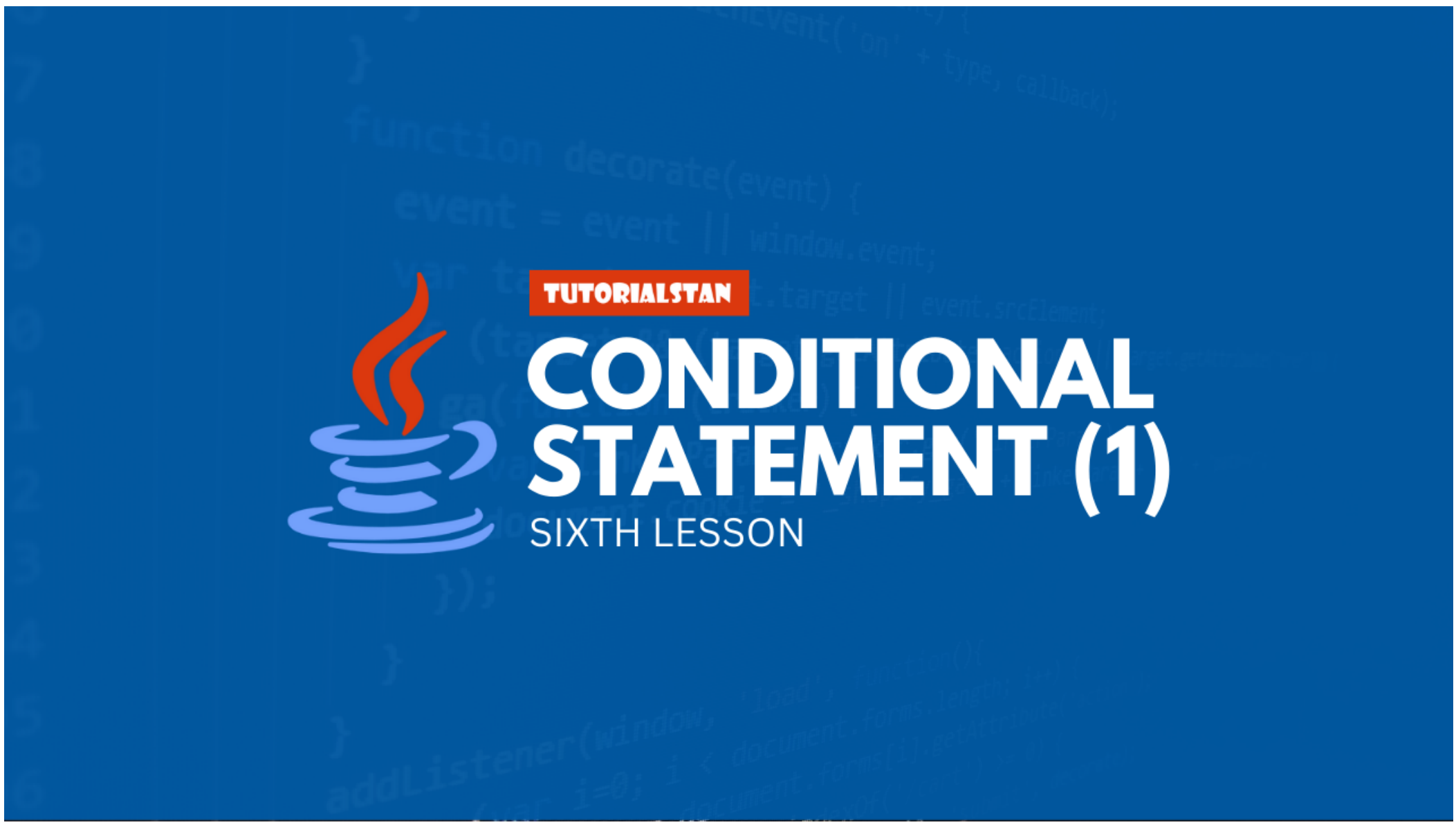
عندما يدخل المريض إلى الطوارئ في المشفى أول خطوة هي أخذ المعلومات والقراءات الحيوية للمريض، في هذا البرنامج سنقوم بعمل برنامج لأخذ المعلومات وهي كالتالي:

- اسم المريض
- عنوان السكن
- العمر
- هل يعاني من السكري
- هل يعاني من الضغط
- درجة الحرارة
- ضغط الدم
- نسبة الأكسجين في الدم
- الطول
- الوزن
- مؤشر BMI (حساب)

ملاحظات:

- يجب أخذ المعلومات جميعها كمدخلات إلا مؤشر BMI مع اختيار النوع المناسب للمتغير ثم عرضها بشكل متناسق من خلال جمل الطباعة
- مؤشر BMI له معادلة خاصة ابحث عنها وهي تعتمد على الطول والوزن لذا لا داعي لأخذها كمدخل.

الدرس السادس - الجملة الشرطية (الجزء الأول)



المحاور

- العمليات المنطقية (مراجعة وتفصل)
- جملة if-else

العمليات المنطقية (مراجعة وتفصل)

1. عمليات المقارنة

تكلمنا عن عمليات المقارنة في الدرس الدرس الثاني وهي أكبر من > أو أصغر من < أو المساواة == أو أكبر أو يساوي >= أو أصغر أو يساوي <= وهذه العمليات تكون نتيجتها إما صح أو خطأ (true,false)

2. المتغير boolean

تكلمنا عنه في درس الvariables وهو عبارة عن نوع متغير قيمته 0 أو 1 ويمكن أن تكون قيمته عملية منطقية بما أن ناتجها يكون 0 أو 1

```
int a=5;
int b=7;
boolean isTrue= a>b;
```

في حال أردنا أن ندمج أكثر من عملية منطقية، كيف يمكننا عمل ذلك؟

3. البوابات المنطقية

هي عمليات تستخدم للربط بين العبابرات المنطقية، وسنغطي في هذه الدورة ثلاث بوابات رئيسية:

1. يوابة And (و):
   - تستخدم هذه البوابة إذا أردنا أن تتحقق عبارتان منطقيتان معا ويرمز لها بالرمز &&
   - مثلا إذا أردنا أن نعلم إذا كان المتغير a أكبر من b وفي نفس الوقت نريد b أن تكون أكبر من صفر (عدد موجب)

```
int a=5;
int b=7;
boolean isTrue= a>b && b>0;
```

في هذا المثال المتغير b أكبر من صفر لكن المتغير a أصغر من b لذا نتيجة العملية False.

يعني في البوابة And يكون الجواب True إذا تحققت (جميع) العبارات في الجملة

- جدول الحقيقة (Truth Table) للبوابة And:

Input		Output
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

على فرض أن A وB عبارة عن عبارتين منطقيتين مع العلم أن 1 دائما تمثل القيمة True و0 تمثل False

2. بوابة OR (أو):
   - تستخدم هذه البوابة إذا أردنا أن تتحقق واحدة من العبارتين المنطقيتين أو كليهما ويرمز لها بالرمز ||
   - مثلا إذا أردنا أن نعلم إذا كان أحد المتغيرين a أو b سالبا (أقل من صفر)

```
int a=5;
int b=-7;
boolean isTrue= a<0 || b<0;
```

في هذه العبارة المتغير a أكبر من صفر لكن المتغير b أقل من صفر لذا سيكون الناتج True

يعني في البوابة OR يكون الجواب True إذا تحققت (أحد) العبارات في الجملة

Input		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

- جدول الحقيقة (Truth Table) للبوابة OR:

3. بوابة NOT
   - هي أبسط البوابات وتقوم بعكس نتيجة العملية ويرمز لها بالرمز !
   - بكل بساطة إذا كان عندنا عبارة أو متغير قيمته True ستحوله إلى False والعكس صحيح

```
boolean a= true;
boolean b= 5 < 4; //false النتيجة
boolean c= !a; //false النتيجة
boolean d= !b; //true النتيجة
```

Input	Output
A	Y
0	1
1	0

- جدول الحقيقة (Truth Table) للبوابة OR:

4. الأولويات في العمليات المنطقية

i. الأقواس
ii. العمليات الحسابية
iii. عمليات المقارنة
iv. بوابة Not
v. بوابة And
vi. بوابة Or مثال:

A=5, B=3, C=-2 Find:

1. A+C > B && B == A - 2 || C < 0
2. !(3A+2 > B || 5-B > 3 && !(-C/3 > .53))

```
1. A+C > B && B == A - 2 || C < 0
   5+(-2)>3 && 3 == 5-2 || -2 < 0
   3>3 && 3 == 3 || -2 < 0
   False && True || True
   False || True
   True

2. !(3A+2 > B || 5-B > 3 && !(-C/3 > .53))
   !(3*5+2 > 3 || 5-3 > 3 && !(-(-2)/3 > .53))
   !(True || False && False)
   !(True || False)
   !(True)
   False
```

جملة if-else

في بعض الأحيان قد لا يكون للبرنامج الخاص مسار واحد مباشر، قد نضطر بناء على شروط معينة تغيير مسار البرنامج بناء على متغيرات معينة.

مثلا أردنا عمل برنامج لمعرفة إذا ما كان الطالب قد نجح في الامتحان أو لا فإذا نجح في الامتحان يطبع البرنامج: مبارك لقد نجحت ,وإذا رسب يطبع له: حظا أوفر.

في مثل هذه الحالات نحتاج ما يعرف باسم الجملة الشرطية وهي بسيطة جدا:

```
if (condition) {
  في حال تحقق الشرط نفذ هذا الكود الموجود بين القوسين //
}
```

مكان condition نضع جملة منطقية تعبر عن الشرط الذي نريد تحقيقه.

وفي حال لم يتحقق الشرط يتم تنفيذ الكود الموجود بين قوسي else:

```
if (condition) {
  في حال تحقق الشرط نفذ هذا الكود الموجود بين القوسين //
}else{
  في حال لم يتحقق الشرط نفذ هذا الكود الموجود بين القوسين //
}
```

مثلا لو أردنا أن يدخل المستخدم عمره وفي حال كان أكبر من 18 سنة نطبع له جملة: You have passed the age requirement أما في حال كان أقل من 18 سنة نطبع you are under the legal age

```
import java.util.Scanner;

class Main {
  public static void main(String[] args) {

      Scanner reader= new Scanner(System.in);

      System.out.println("Enter your age:");
      int age= reader.nextInt();

      if (age >= 18) {
        System.out.println("You have passed the age requirement");
      }else{
        System.out.println("You are under the legal age");
      }
    }
}
```

الدرس السابع - الجملة الشرطية (الجزء الثاني)

المحاور

- else if Statement
- switch Statement

else if Statement

في الدرس الماضي تكلمنا عن جملة if وقلنا في حال كان لدينا شرطين متعاكسين نستخدم else، طيب إذا كان عندنا أكثر من شرطين متعاكسين، مثلا لو كان عندنا برنامج يأخذ علامة الطالب ثم يصنفها إلى ممتاز، جيد جدا، جيد، راسب كيف يمكننا كتابة أكثر من شرط؟

في الحقيقة هناك جملة يمكن إضافتها قبل else لإضافة شروط جديدة، وهي else if، وتكتب كالتاي:

```
if (condition1) {
  // في حال تحقق الشرط نفذ هذا الكود الموجود بين القوسين
}else if (condition2){
  // في حال لم يتحقق الشرط الأول وتحقق هذا الشرط نفذ هذا الكود الموجود بين القوسين
}
else{
  // في حال لم تتحقق جميع الشروط نفذ هذا الكود الموجود بين القوسين
}
```

لتنفيذ مثال تصنيف العلامات الذي ذكرناه في البداية:

```
import java.util.Scanner;

class Main {
  public static void main(String[] args) {

      Scanner reader= new Scanner(System.in);

      System.out.println("Enter your Mark:");
      double mark= reader.nextDouble();

      if (mark >= 90) {
        System.out.println("Excelent");
      }else if(mark >= 75){
        System.out.println("Very Good");
      }else if(mark >= 50){
        System.out.println("Good");
```

```
      }else{
        System.out.println("Failed");
      }
    }
}
```

switch Statement

لو ضربنا مثالا أنك صاحب شركة لها 7 فروع تريد عمل برنامج يضع العميل اسم المدينة التي فيها الفرع والبرنامج يعطيه العنوان الكامل ومعلومات التواصل وفي حال وضع العميل اسم مدينة أخرى يطبع البرنامج رسالة بأسماء المدن التي يوجد فيها فروع للمدينة، كيف ستبدأ بهذا البرنامج؟

لا بد أنك فكرت في الif else صحيح؟ إذا لم لا نجرب:

أولا: نقوم بطلب ادخال اسم المدينة ونضعه في متغير نصي اسمه city (للاختصار لن أكتب الكود، لا بد أنك صرت محترفا في هذه الخطوة)

ثانيا: نقوم بكتابة كود الطباعة على فرض أن أسماء المدن التي توجد فيها الفروع هي city1, city2, ...city7 والعناوين الكاملة Address1, Address2, ...Address7:

```
if (city == "city1") {

  System.out.println("Address: Address1");
  System.out.println("phone: phone1");

}else if(city == "city2"){

  System.out.println("Address: Address2");
  System.out.println("phone: phone2");

}
else if(city == "city3"){

  System.out.println("Address: Address3");
  System.out.println("phone: phone3");

}
else if(city == "city4"){

  System.out.println("Address: Address4");
  System.out.println("phone: phone4");

}
else if(city == "city5"){

  System.out.println("Address: Address5");
  System.out.println("phone: phone5");

}
else if(city == "city6"){

  System.out.println("Address: Address6");
  System.out.println("phone: phone6");

}
else if(city == "city7"){

  System.out.println("Address: Address7");
  System.out.println("phone: phone7");

}
else{
  System.out.println("there is no branches in "+ city);
  System.out.println("please choose one of these cities: city1, city2, city3, city4, city5, city6, city7");

}
```

الآن لدينا برنامج يقوم بهذه المهمة على أكمل وجه :)

لكن ماذا لو أخبرتك أن لدينا طريقة أسهل وكود أبسط لكتابة نفس البرنامج وهي جملة switch:

```
switch (city) {
  case "city1":
      System.out.println("Address: Address1");
      System.out.println("Phone: phone1");
```

```
        break;
    case "city2":
        System.out.println("Address: Address2");
        System.out.println("Phone: phone2");
        break;
    case "city3":
        System.out.println("Address: Address3");
        System.out.println("Phone: phone3");
        break;
    case "city4":
        System.out.println("Address: Address4");
        System.out.println("Phone: phone4");
        break;
    case "city5":
        System.out.println("Address: Address5");
        System.out.println("Phone: phone5");
        break;
    case "city6":
        System.out.println("Address: Address6");
        System.out.println("Phone: phone6");
        break;
    case "city7":
        System.out.println("Address: Address7");
        System.out.println("Phone: phone7");
        break;
    default:
        System.out.println("There are no branches in " + city);
        System.out.println("Please choose one of these cities: city1, city2, city3, city4, city5, city6, city7");
}
```

- في جملة switch نضع بين القوسين المتغير الذي نريد أن نختبر قيمته ثم نضع كل قيمة في جملة case
- نفصل بين كل جملة case والأخرى بالأمر break
- الجملة المكتوبة في default تستخدم بدلا من جملة else
- جملة default لا داعي لإنهائها بbreak إذا كانت آخر خيار
- جملة else if استخدامها أوسع إذ أنها تمكنك من اختيار نوع المقارنة (أكبر من، أصغر من، and, or...)
- جملة switch أسهل وأقصر عندما تكون المقارنة بين عدد من القيم المعروفة

الدرس الثامن - جمل التكرار

المحاور

- while loop
- do while loop
- for loop

while loop

في بعض الأحيان قد نحتاج أن نكتب كود يتكرر عدة مرات، ومن غير المنطقي إذا أردنا أن نكرر تنفيذ الكود 10 مرات أن نكتبه 10 مرات، في هذه الحالة نلجأ إلى جمل التكرار.

وأول جملة من جمل التكرار هي الwhile وطريقة كتابتها قريبة جدا على الجملة الشرطية if:

```
while (condition) {
  // طالما أن الشرط صحيح نفذ هذا الكود الموجود بين القوسين
}
```

في الcondition نضع الشرط الذي نريده، ومن أبسط البرامج التي يمكن كتابتها من خلال جمل التكرار هي العداد:

```
int i= 0;
while (i < 10) {
  i = i+1;
  System.out.println(i);
}
```

- في البرنامج السابق قمنا بتعريف int قيمته المبدئية 0
- ثم قمنا بعمل جملة while تنفذ الكود الذي بداخلها طالما أن قيمة المتغير i أصغر من 10
- في الكود الذي بداخل الwhile نقوم أولا بزيادة قيمة i ثم طباعة القيمة الجديدة
- سيقوم البرنامج بطباعة الأرقام من 1 إلى 10

هناك طريقة مختصرة لكتابة نفس الكود من خلال استخدام أمر الزيادة (Increment):

- لدينا نوعين من العمليات الحسابية لم نتحدث عنها وهي الزيادة (Increment) والنقصان (Decrement)
- إشارة الزيادة هي (++) والنقصان (--)

- توضع هذه الإشارة قبل أو بعد المتغير كالتالي

```
int i =0;
// الطريقة التي السابقة لزيادة قيمة i
i = i+1;

// عن طريق استخدام أمر الزيادة يمكننا استخدام هذا الكود للقيام بنفس العملية
i++;
// أأو
++i;
```

- الفرق بين i++ و ++i أننا لو استخدمناها داخل أمر آخر ستقوم ال++i بتنفيذ الأمر ثم زيادة قيمة i أما i++ ستقوم بزيادة قيمة i ثم تنفيذ الأمر، مثال:

```
int i =0;
// i وهي 0 ثم إضافة 1 إلى i سيقوم هذا الكود بطباعة قيمة
System.out.println(i++);
// بعد تنفيذ الأمر السابق ستطبع القيمة الجديدة وهي 1 i لو طبعنا
System.out.println(i);

// 2 مثال
int x=0;
// ثم يقوم بطباعتها بقيمتها الجديدة مباشرة xسيقوم هذا الكود بإضافة 1 إلى ال
System.out.println(++i);
```

يمكننك تجريب نفس الأكواد السابقة ولكن مع --i و i--

لو أردنا كتابة البرنامج الأول ولكن بالعد العكسي وعملية النقصان بدلا من الزيادة سيكون البرنامج كالتالي:

```
int i= 10;
while (i > 0) {
  System.out.println(i--);
}
```

do while loop

في بعض الأحيان نحتاج إلى تنفيذ الكود مرة سواء كان موافق للشرط أو لا ثم إعادة تنفيذ الكود إذا تحقق الشرط وفي هذه الحالة لدينا خيارين:

1. كتابة الكود مرة خارج الwhile ومرة داخلها

```
int i= 10;
// سينفذ الكود هذا سواء تحقق الشرط أو لم يتحقق
System.out.println(i--);
while (i > 0) {
  System.out.println(i--);
}
```

2. استخدام do while

```
int i= 10;
do{
  System.out.println(i--);
}while (i > 0)
```

سيقوم الكود الثاني بنفس عمل الكود الأول لكنه أسهل وأقصر

for loop

هناك نوع آخر من جمل التكرار وهو الfor loops، طريقة عمله نفس طريقة عمل الwhile لكن طريقة الكتابة تختلف

وأول جملة من جمل التكرار هي الwhile وطريقة كتابتها قريبة جدا على الجملة الشرطية if:

```
for (statement1; condition; statement2) {
  // طالما أن الشرط صحيح نفذ هذا الكود الموجود بين القوسين
}
```

- statement1: تنفذ مرة واحة في بداية الloop
- condition: هنا نضع الشرط
- statement2: تنفذ كل مرة بعد نهاية الدورة

غالبا ما يتم تعريف المتغير الذي يستخدم في العد في الstatement1 وتنفذ جملة الزيادة في statement2، وغالبا ما تستخدم جملة for عندما نعرف عدد الدورات أما في الحالات الأخرى نستخدم while.

مثال:

```
for (int i= 0;i < 10;i++) {
  System.out.println(i);
}
```