## الجلسة الثانية - برمجة 3

**الغاية من الجلسة:** فهم الطريقة ()toString بالإضافة إلى توضيح فكرة الـ static وتطبيقها من خلال مثال.

ليكن لدينا:

1. الصف Test يحوي على الطريقة ()main.
2. الصف Student.

أولاً: قم بإنشاء الصفين:

إنشاء الصف Test .

```
1   package lec2;
2   import java.util.Scanner;
3
4
5   public class Test {
6       public static void main(String[] args) {}
7   }
8
9
```

ثم إنشاء الصف Student في ملف آخر.

```
1   package lec2;
2
3   public class Student {}
4
5
```

كما نعلم أن لكل صف حقول معطيات تعرف هذا الصف ,

في مثالنا الصف Student لديه الحقول التالية :

اسم الشخص (name)

عمر الشخص (age)

```
1   package lec2;
2
3   public class Student {
4
5       private String name;
6       private int age;
7
8   }
```

كما نعلم أن الواصفات يجب أن تكون محمية وبالتالي نمط الوصول private . بالتالي نحتاج لتوابع set & get للتعامل معها.

```
6
7   public void setName(String n)
8   {
9     this.name=n;
10  }
11  public String getName()
12  {
13   return this.name;
14  }
15   public void setAge(int a)
16  {
17       this.age=a;
18  }
19  public int getAge()
20  {
21   return this.age;
22  }
23  }
24
```

قم بكتابة بان افتراضي يهيئ الاسم بـ Ahmad و العمر ب 20.

```
public Student()
{
    this.name="Ahmad";
    this.age=20;

}
```

قم بكتابة بان بوسطاء:

```
}
public Student(String name,int age)
  {
    this.name=name;
    this.age=age;

  }
```

this مؤشر يشير للغرض الحالي.

قم ببناء غرض من الطالب في صف test

```java
package lec2;


public class Test {
    public static void main(String[] args) {
            Student s1 = new Student();
            System.out.println(s1);

    }
}
```

سيظهر الخرج كمايلي :

```
run:
lec2.Student@15db9742
BUILD SUCCESSFUL (total time: 3 seconds)
```

وهو عبارة عن اسم الصف وبعده إشارة @ ثم الـ hashCode (وهو رقم صحيح مرتبط بالغرض ويميزه عن غيره) حيث تم (ضمنياً) استدعاء الدالة الجاهزة ()toString المعرفة في صف Object والتي تعيد موقع الغرض، إذا اردنا ان نعرض بيانات الغرض عند طباعته فيجب ان نحمل بشكل زائد الدالة ()toString:

```java
    public String toString()
    {
      return " name: "+this.name +" age: "+this.age ;
    }
    }
```

نعيد التنفيذ، ليصبح كما يلي:

```
run:
 name: Ahmad age: 20
BUILD SUCCESSFUL (total time: 2 seconds)
```

نريد أن يكون لكل طالب رقم جامعي تسلسلي، أي يزداد بشكل تلقائي، وبالتالي قم بإضافة حقل الرقم الجامعي لصف الطالب.

```java
package lec2;

public class Student {
    private String name;
    private int age;
    private int id ;
```

سنقوم بإضافة تابع getId، لا نريد تابع تهيئة setId كي لا يتم التلاعب بقيمة هذا الحقل.

```
public String getName()
{
 return this.name;
}
 public void setAge(int a)
{
     this.age=a;
}
public int getAge()
{
 return this.age;
}
public int getId()
{
 return this.id;
```

الرقم الجامعي خاص بكل طالب وقيمته ستزداد تلقائيا عند انشاء طالب جديد، وبالتالي نحتاج إلى عداد ستاتيكي count:

```
3   public class Student {
4       private String name;
5       private int age;
6       private int id ;
7       private static int count=0;
8
9
10
```

```
31  public Student()
32  {
33      this.name="Ahmad";
34      this.age=20;
35      count++;
36      id=count;
37
38    }
39  public Student(String name,int age)
40    {
41      this.name=name;
42      this.age=age;
43      count++;
44      id=count;
45
46    }
```

قم بإنشاء طالبين مع عرض كافة المعلومات الخاصة بهم:

سنعدل دالة toString() لتطبع الرقم الجامعي مع البيانات الخاصة بالطالب:

```
    }
  public Student(String name,int age)
    {
      this.name=name;
      this.age=age;
      count++;
      id=count;

    }
     public String toString()
    {
      return " name: "+this.name +" age: "+this.age+" id: "+this.id  ;
    }
    }
```

دالة main:

```
    public class Test {
        public static void main(String[] args) {
                Student s1 = new Student();
                Student s2 = new Student("karam",33);

                System.out.println(s1);
                System.out.println(s2);


        }
    }
```

يكون الخرج كما يلي :

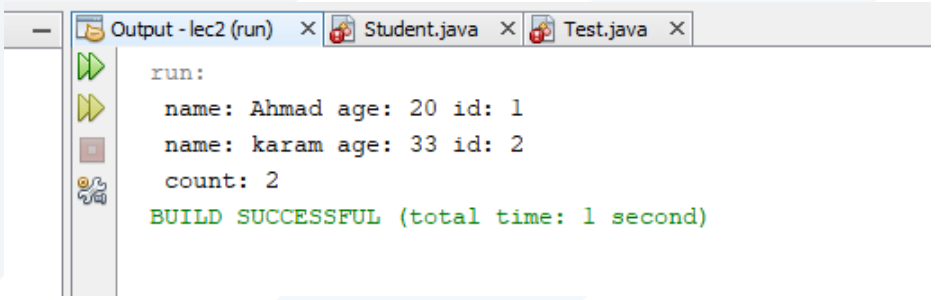
```
run:
 name: Ahmad age: 20 id: 1
 name: karam age: 33 id: 2
BUILD SUCCESSFUL (total time: 59 seconds)
```

اذا اردنا ان نتعامل مع count سنقوم بانشاء دالة ستاتيكية لطباعته.

```
    }
    public String toString()
    {
      return " name: "+this.name +" age: "+this.age+" id: "+this.id  ;
    }
    public static void print()
    {
      System.out.println( " count: "+ count) ;
    }
    }
```

```
public class Test {
    public static void main(String[] args) {
            Student s1 = new Student();
            Student s2 = new Student("karam",33);

            System.out.println(s1);
            System.out.println(s2);
            Student.print();
```




ملاحظة :

لايمكن للدوال الستاتيكية أن تصل للمتحولات غير الستاتيكية، بينما يمكن للدوال العادية ان تصل للمتحولات الستاتيكية وغير الستاتيكية .

شرح الملاحظة:

ليكن لدي الصف التالي:

```
6  package lec2;
7
8
9  public class Number {
10
11     private int x;
12     private static int y;
13
14
15     public static void print()
16     {
17       System.out.println( " x: "+ x +"y:"+ y) ;
18     }
19       public  void print2()
20     {
21       System.out.println( " x: "+ x +"y:"+ y) ;
22     }
23     }
24
```

كما نعلم ان المتحول y ستاتيكي وبالتالي هو على مستوى الصف، بينما المتحول x سيتم انشائه بانشاء الغرض.

كذلك الدالة print() هي دالة ستاتيكية وبالتالي لاستدعائها لا أحتاج الى غرض وبالتالي x غير منشأة، فهذه الدالة خاطئة.

بينما الدالة print2() لاستدعائها احتاج الى غرض وبالتالي بانشاء الغرض سيتم انشاء الحقل x و الحقل y موجود بوجود الصف وبالتالي الدالة الثانية صحيحة.

سؤال:

هل يمكن استخدام this مع المتحول الستاتيكي ؟؟؟

سؤال آخر: فكر في مثال آخر عن المتحول الستاتيكي غير فكرة العداد count الذي استخدمناه سابقاً.