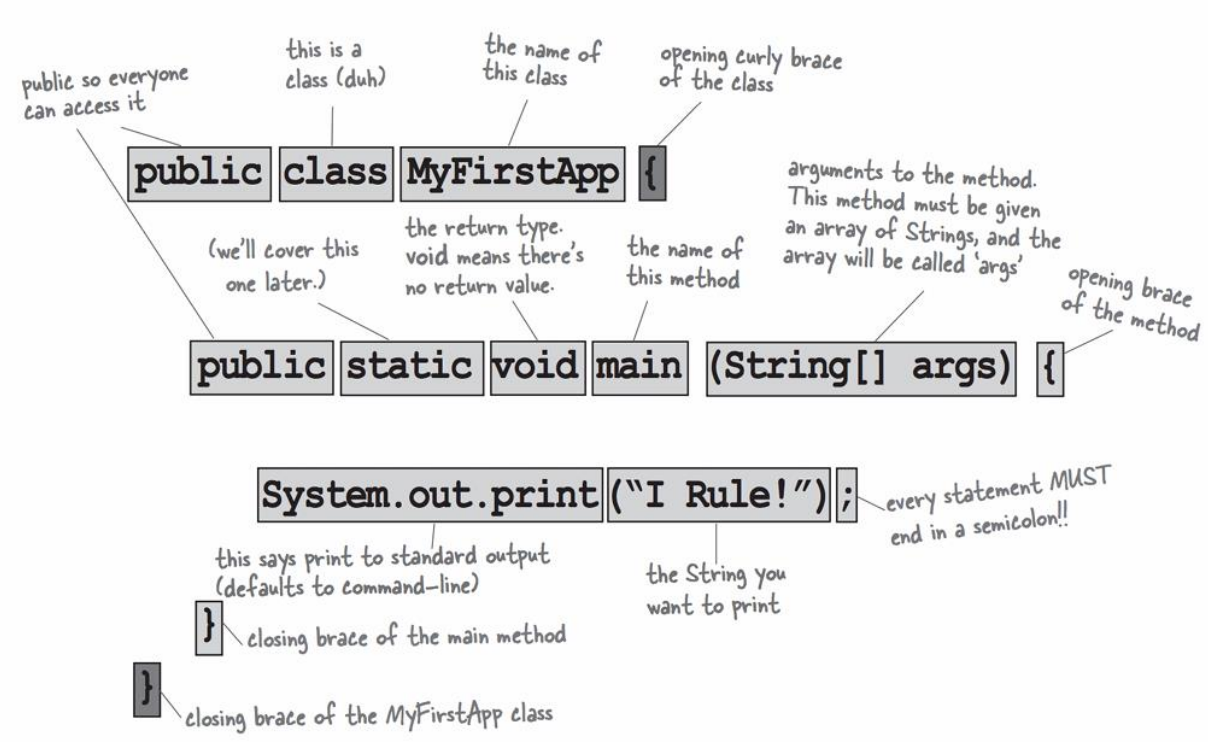
FREE

Topic 1 Basic Java and OOP

*الكلمات الي تحتها خط ركزوا عليها

❖ Basic Java

Java code structure (main method):



❖ Printing the output

```
public static void main(String[] args) {

    System.out.println("My First Project");
}
```

The Output : My First Project

❖ Java Variables

- **Programs work by manipulating data placed in memory.**
- **The data can be numbers, text, objects … etc.**
- **In Java, the main types of variables: int, double, char, boolean, String.**

- **البرامج تعمل عن طريق التلاعب بالبيانات الموضوعة في الذاكرة.**
- **يمكن أن تكون البيانات أرقامًا أو نصوصًا أو كائنات … إلخ.**
- **في جافا، الأنواع الرئيسية للمتغيرات int :، double، char، boolean ، String.**

- int *number1, age*;
- int *salary*= 8500;
- double *price* = 9.99;
- Float x = 8.6;
- char *letter* = ‘A’;
- boolean *flag* = true or False;
- String *FirstName* = “Salman”;

❖ Example on Variables

```
public static void main(String[] args) {

    int first_number, second_number, answer;

    first_number = 10;
    second_number = 20;
    answer = first_number + second_number;

    System.out.println("Addition Total = " + answer );
}
```

- **How many variables?**

  3 variables
- **What are the types of variables?**

  int
- **What are the names of the variables?**

  first_number , second_number , answer
- **What is the output?**

  Addition Total = 30

❖ Control Flow

- **IF statement**

```
public static void main(String[] args) {

    int user = 45;

    if (user <= 18) {
        System.out.println("User is 18 or younger");
    }
    else if (user > 18 && user < 40) {
        System.out.println("User is between 19 and 39");
    }
    else if (user == 45 || user == 50) {
        System.out.println("User is either 45 OR 50");
    }
    else {
        System.out.println("User is older than 40");
    }
}
```

The Output : User is either 45 OR 50

```
public static void main(String[] args) {

    int user = 18;

    switch ( user ) {
        case 18:
            System.out.println("You're 18");
            break;
      case 19:
            System.out.println("You're 19");
            break;
      case 20:
            System.out.println("You're 20");
            break;
      default:
          System.out.println("You're not 18, 19 or 20");
    }

}
```

The Output : You're 18

❖ Loop

- **For-loop statement**

```
public static void main(String[] args) {

    int loopVal;
    int end_value = 11;
    int addition = 0;

    for (loopVal = 1; loopVal < end_value; loopVal++) {

        addition = addition + loopVal;
    }

    System.out.println("Total = " + addition);
}
```

The Output : Total = 55

- **While statement**

```
int i = 0;
while (i < 5) {
    System.out.println(i);
    i++;
}
```

The Output :

0
1
2
3
4

❖ Arrays

- **An array is a way to have more than one value at a time.**
- **مصفوفة هي طريقة لتخزين أكثر من قيمة في نفس الوقت.**

```
public class ArraysTest {

    public static void main(String[] args) {

        int[] aryNums;

        aryNums = new int[6];

        aryNums[0] = 10;
        aryNums[1] = 14;
        aryNums[2] = 36;
        aryNums[3] = 27;
        aryNums[4] = 43;
        aryNums[5] = 18;

        System.out.println( aryNums[2] );
    }

}
```

The Output : 36

Object-oriented programming (OOP)

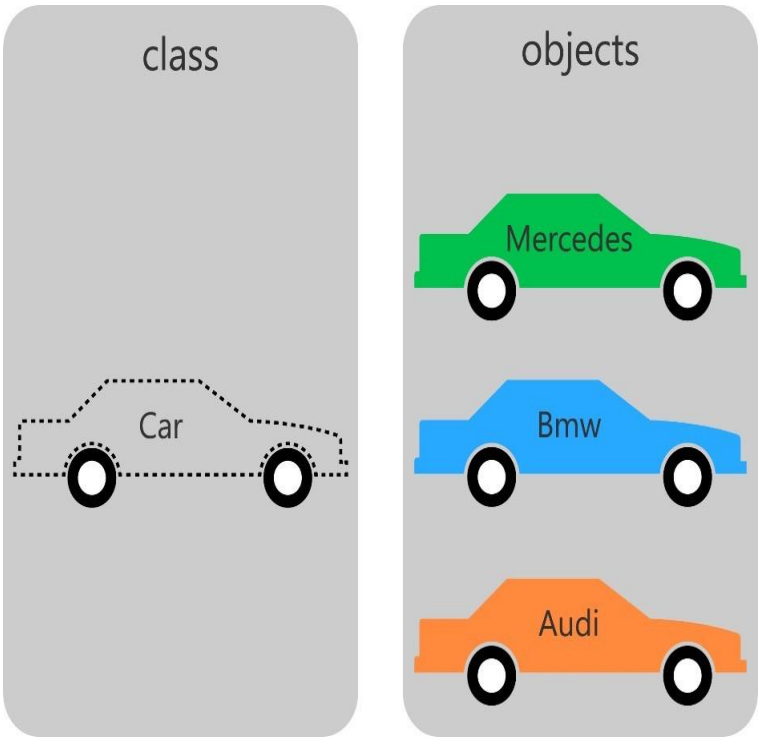
البرمجة الكائنية

- **OOP** is a programming paradigm based on objects, instead of just functions and procedures. The objects are usually organized or belong to specific classes.

- هي نمط برمجة يستند إلى الكائنات بدلاً من الوظائف والإجراءات فقط. يتم تنظيم الكائنات عادة أو تنتمي إلى classes محددة

توضيح لمصطلح programming paradigm يشير هذا المصطلح إلى مجموعة من المبادئ والممارسات التي تحدد كيفية بناء البرامج , في انواع كثيره من ال paradigm اولهم ال OOP وهي من خلالها تتكون البرامج من classes و عدة object الخ , وفي نوع مثلا procedure-style programming ومن خلاله تحتوي البرامج على methods واجراءات فقط بدون object

- An 'object' in an OOP language refers to a specific type, or 'instance' of a class.
- يشير 'الكائن' في لغة البرمجة الكائنية إلى نوع محدد أو 'مثيل' من الكلاس .



❖ Advantages of OOP المزايا

*يجب حفظ الأربع النقاط

1. Improved software development productivity

- Modularity: The software can be divided into several parts.
- Extensibility: Each part can be extended easily.
- Re-usability: Code can be re-used across applications.

1- زيادة إنتاجية تطوير البرمجيات

- التجزئة: يمكن تقسيم البرنامج إلى أجزاء عدة.
- القابلية للتوسع: يمكن توسيع كل جزء بسهولة.
- إعادة الاستخدام: يمكن إعادة استخدام الكود في تطبيقات أخرى.

2. Improved software maintainability

- Part of the system can be changed (in case of issues) without the need to change a large scale changes.

2- تحسين صيانة البرمجيات

- يمكن تغيير جزء من النظام (في حالة وجود مشاكل) دون الحاجة إلى إجراء تغييرات كبيرة.

3. Faster development

- OOP languages come with (built-in) rich libraries of objects.
- Code can be re-used in future projects.

3- تطوير أسرع

- لغات البرمجة الكائنية تأتي مع مكتبات غنية من الكائنات.
- يمكن إعادة استخدام الكود في مشاريع مستقبلية

4. Lower cost of development

More effort is put into the analysis and design which lowers the overall cost of development.

4- تكلفة تطوير أقل

- يتم وضع مزيد من الجهد في تحليل وتصميم البرمجة الكائنية مما يقلل من التكلفة الإجمالية للتطوير.

❖ Disadvantages of OOP العيوب

*يجب حفظ الأربع النقاط

1. Steep learning curve

- The process of learning OOP may not initially seem natural for some people. They need time to understand and apply the OOP concepts.

1- منحنى التعلم الشديد

- قد لا يبدو عملية تعلم البرمجة الكائنية طبيعيًا في البداية بالنسبة لبعض الأشخاص. يحتاجون إلى وقت لفهم وتطبيق مفاهيم البرمجة الكائنية.

2. Larger program size

- Object-oriented programs usually involve more lines of codes than procedural programs.

2- حجم البرنامج كبير

- البرامج الكائنية تتضمن عادة مزيدًا من سطور الشفرة من البرامج الإجرائية

3. Slower programs

- Object-oriented programs are typically slower than procedural programs because they require more instructions to be executed.

3- بطئ البرامج

- البرامج الكائنية عادة ما تكون أبطأ من البرامج الإجرائية لأنها تتطلب مزيدًا من التعليمات للتنفيذ.

4. Not suitable for all types of problems

Some problems can be better implemented in a procedure-style programming, functional-programming style. Applying OOP for those problems may not result in efficient and effective programs.

4- غير مناسبة لجميع أنواع المشاكل

- يمكن تنفيذ بعض المشاكل بشكل أفضل في برمجة بنمط الإجراءات أو البرمجة الوظيفية. تطبيق البرمجة الكائنية لهذه المشاكل قد لا يؤدي إلى برامج فعالة

❖ Objects & Classes

- Objects share two characteristics; (1) state and (2) behavior.
  - State is a well defined condition of an item. A state captures the relevant aspects of an object.
  - Behavioris the observable effects of an operation or event.

- الكائنات تشترك في خاصيتين؛ (1) الحالة و (2) السلوك.
  - الحالة: هي الوضع أو الشرط المعين لأي كائن في لحظة معينة. تحدد الحالة الخصائص والمميزات المهمة التي تصف الكائن.
  - الســـلوك: هو ما يميزه من حيث التأثيرات والنتائج الملحوظة التي تحدث عند تنفيذ عمليات أو إجراءات عليه.

- An object stores its state in variables and exposes its behavior through methods (also called functions).
- يخزن الكائن حالته في متغيرات ويكشف عن سلوكه من خلال الوظائف (تسمى أيضًا الدوال).

- Example 1:
  - Object: Student1
  - State (Variables): ID, name, GPA, birthDate
  - Behaviour (methods): setName, getName, calculate_age

- Example 2:
  - Object: calculatorX
  - State (Variables): number1, number1
  - Behaviour (methods): add, sub, multiply, divide

• **A class** is a **prototype** that defines the **variables and the methods** common to all objects of a specific type.

- Methods of a class operate upon the member variables of a class.
- An object is created when a class is instantiated.

- **الكلاس** هو نموذج يعرف المتغيرات والدوال المشتركة لجميع الكائنات من نوع محدد.
- دوال الكلاس تعمل على المتغيرات في الكلاس.
- يتم إنشـاء كائن عندما يتم تثبيت كلاس معين. بمعنى آخر، الكائن هو نسـخة محددة من الكلاس، وتعتبر العملية التي تنشئ الكائن من الكلاس بما يسمى "التثبيت" (Instantiation).

- Class deceleration in Java (Example):

```
class Book {
    // class body
}
```

Exercise:

- Define all possible variables and methods for the class Book.
- حدد جميع المتغيرات والدوال الممكنة لكلاس الكتاب.

- Variables: name, price , author
- methods: setName, getName, setPrice, getPrice , setAuthor, getAthor

- المتغيرات: الاسم, السعر , المؤلف
- الدوال : دوال ال Mutator وال Accessor او دوال ال setter وال getter لكل متغير

❖ In-Class Demonstration

- We want to create a very simple program using objects and classes in Java.
  - The program is for handling employees data in a company.
  - Employee data should have a name, age and salary.
  - We want to set the name, age and salary for each employee.
  - We also want to print all the employee information: (name, age and salary).

- نريد إنشاء برنامج بسيط جدًا باستخدام الكائنات والصفوف في جافا.
  - البرنامج للتعامل مع بيانات الموظفين في شركة.
  - يجب أن تحتوي بيانات الموظف على الاسم والعمر والراتب.
  - نريد تعيين الاسم والعمر والراتب لكل موظف.
  - نريد أيضًا طباعة جميع معلومات الموظف : الاسم والعمر والراتب

- **اولا** نقوم بإنشاء كلاس مثلا بإسم **Emp** ونحط فيه جميع المتغيرات والدوال

```
public class Emp {

    /*
    inside the class we define the
    variables & methods.
    */

} //end class
```

- **ثانيا** نقوم بتعريف جميع المتغيرات داخل كلاس **Emp**

```
String name;
int age;
double salary;
```

- **ثالثا** نقوم بإنشاء Constructor داخل كلاس Emp لتهيئة متغير ال name

*ملاحظة يمكنك تهيئة جميع المتغيرات ولكن من الأفضل ان تجعله للأسم فقط لان الأسم يظل ثابت ولكن العمر والراتب قد يتغير قيمته من فترة الى اخرى

```
public Emp(String n) {
      this.name = n;
}
```

Notes:

- When an object is created, Java calls the constructor first. Any code you have in your constructor will then be executed.
- You don't need to make any special calls to a constructor method- they happen automatically when you create a new object.
- Constructor methods take the same name as the class.
- A class can have more than one constructor.

ملاحظات :

- عند إنشـــاء كائن ، يقوم جافا بإســـتدعاء constructor أولاً. ســـيتم تنفيذ أي كود لديك في constructor.
- لا تحتاج إلى إجراء أي استدعاءات خاصة لطريقة constructor- تحدث تلقائيًا عند إنشاء كائن جديد.
- تأخذ طرق constructor نفس اسم الكلاس .
- يمكن أن يحتوي الكلاس على أكثر من constructor واحد.

- **رابعا** نقوم بإضافة جميع الدوال Methods داخل كلاس Emp

```
// Assign Emp age of to the variable age.
public void setAge(int empAge) {
   this.age = empAge;
 }
```

```
// Assign Emp salary to the variable salary.
public void setSalary(double empSalary) {
   this.salary = empSalary;
}
```

```
/* Print the Employee details */

public void printEmpData() {
   System.out.println("---------------");
   System.out.println("Name:"+ name );
   System.out.println("Age:" + age );
   System.out.println("Salary:" +   salary);
}
```

- **So far we have created/defined:**
  - **The class Emp**
  - **Its related variables (name, age, salary)**
  - **The constructor method of the class Employee**
  - **The method setAge() to set the age of the employee**
  - **The method setSalary()to set the salary of the employee**
  - **The method printEmpData()to print the employee information**

- **Now, we would test our class Employee. How?**
  - **الأن قد تم إنشـاء كلاس ال Employee مع المتغيرات والدوال , كيف الأن راح نختبر او نجرب هذا الكلاس , بحتاج لنا كلاس عشان نضع فيه دالة main انظروا الى الخطوة الخامسة**

❖ Program Testing

- **خامسا** نقوم بإنشاء كلاس EmployeeTest وننشئ داخله دالة main

```
public class EmployeeTest {

  public static void main(String[] args) {

       /*
          inside the main we create objects from
          the class Emp and we an call
          methods for each object.
       */

   } // end main

} // end class
```

❖ Creating Objects & Calling Methods

- **سادسا** نقوم داخل دالة main بإنشاء objects بعدد الموظفين كما نريد ونقوم بإستدعاء الدوال

```
/* Create two objects using constructor */
      Emp empOne = new Emp("Omar Khalid");
      Emp empTwo = new Emp("Osama Ali");

// Invoking methods for each object created
      empOne.setAge(26);
      empOne.setSalary(1000.40);
      empOne.printEmpData();

      empTwo.setAge(21);
      empTwo.setSalary(500);
      empTwo.printEmpData();
```

❖ Full program (كامل كود البرنامج)

```
class Emp{
      String name;
      int age;
      double salary;

      public Emp(String n) {
           this.name = n;
           }

      public void setAge(int empAge) {
           this.age = empAge;
           }

      public void setSalary(double empSalary) {
           this.salary = empSalary;
           }

      public void printEmpData() {
           System.out.println("---------------");
           System.out.println("Name:"+ name );
           System.out.println("Age:" + age );
           System.out.println("Salary:" + salary);
          }


}

public class EmployeeTest {

     public static void main(String[] args) {

           Emp empOne = new Emp("Omar Khalid");
           Emp empTwo = new Emp("Osama Ali");

           empOne.setAge(26);
           empOne.setSalary(1000.40);
           empOne.printEmpData();
           empTwo.setAge(21);
           empTwo.setSalary(500);
           empTwo.printEmpData();

     }}
```

❖ Program Output

```
------------------------
Name:Omar Khalid
Age:26
Salary:1000.4
------------------------
Name:Osama Ali
Age:21
Salary:500.0
```

❖ Exercise

❖ **Improve the previous program taking into account the following points:**

- **The job title of each employee can be added**
- **The program can print the employee information including the job title**
- **An object of the class Employee can also be created without initiating a name. Define a new method to add the name of the employee.**
- **Test your program, and check the result/output.**

- **قم بتحسين البرنامج السابق مع مراعاة النقاط التالية:**
  - **يمكن إضافة عنوان وظيفة لكل موظف**
  - **يمكن للبرنامج طباعة معلومات الموظف بما في ذلك عنوان الوظيفة**
  - **يمكن أيضًا إنشاء كائن من فئة الموظف دون تهيئة اسم. قم بتعريف طريقة جديدة لإضافة اسم الموظف**

ثم اختبر برنامجك وافحص النتيجة او المخرجات

❖ **الحل سنضيف متغير للوظيفة و عشان نطبعها مع معلومات الموظف سنضيفها لدالة الطباعة وعشان نسوي object بدون ما نهيئ قيمة الأسم راح ننشئ default constructor وعشان نسوي دالة جديدة لإضافة الأسم راح نسوي دالة setName**

❖ Full program (كاملا كود البرنامج)

```
class Emp{
      String name;
      int age;
      double salary;
      String jop;
      public Emp() {}
      public Emp(String n) {
           this.name = n;
           }

      public void setName(String n) {
           this.name = n;
           }

      public void setJop(String j) {
           this.jop = j;
           }

      public void setAge(int empAge) {
           this.age = empAge;
           }

      public void setSalary(double empSalary) {
           this.salary = empSalary;
           }

      public void printEmpData() {
           System.out.println("---------------");
           System.out.println("Name:"+ name );
           System.out.println("Age:" + age );
           System.out.println("Salary:" + salary);
           System.out.println("Jop:" + jop);
          }


}
```

```java
public class EmployeeTest {

	public static void main(String[] args) {
		Emp empOne = new Emp();
		Emp empTwo = new Emp();
		empOne.setName("Omar Khalid");
		empOne.setJop("Doctor");
		empOne.setAge(26);
		empOne.setSalary(1000.40);
		empOne.printEmpData();
		empTwo.setName("Osama Ali");
		empTwo.setJop("Manager");
		empTwo.setAge(21);
		empTwo.setSalary(500);
		empTwo.printEmpData();

	}

}
```

❖ Program Output

```
---------------
Name:Omar Khalid
Age:26
Salary:1000.4
Jop:Doctor
---------------
Name:Osama Ali
Age:21
Salary:500.0
Jop:Manager
```

❖ Type of Access Modifier

There are four access modifiers keywords in Java and they are:

يوجد اربع انواع لل modifiers مهمين جداااااااااا

Default	declarations are visible only within the package (package private)
private	declarations are visible within the class only
protected	declarations are visible within the package or all subclasses
public	declarations are visible everywhere

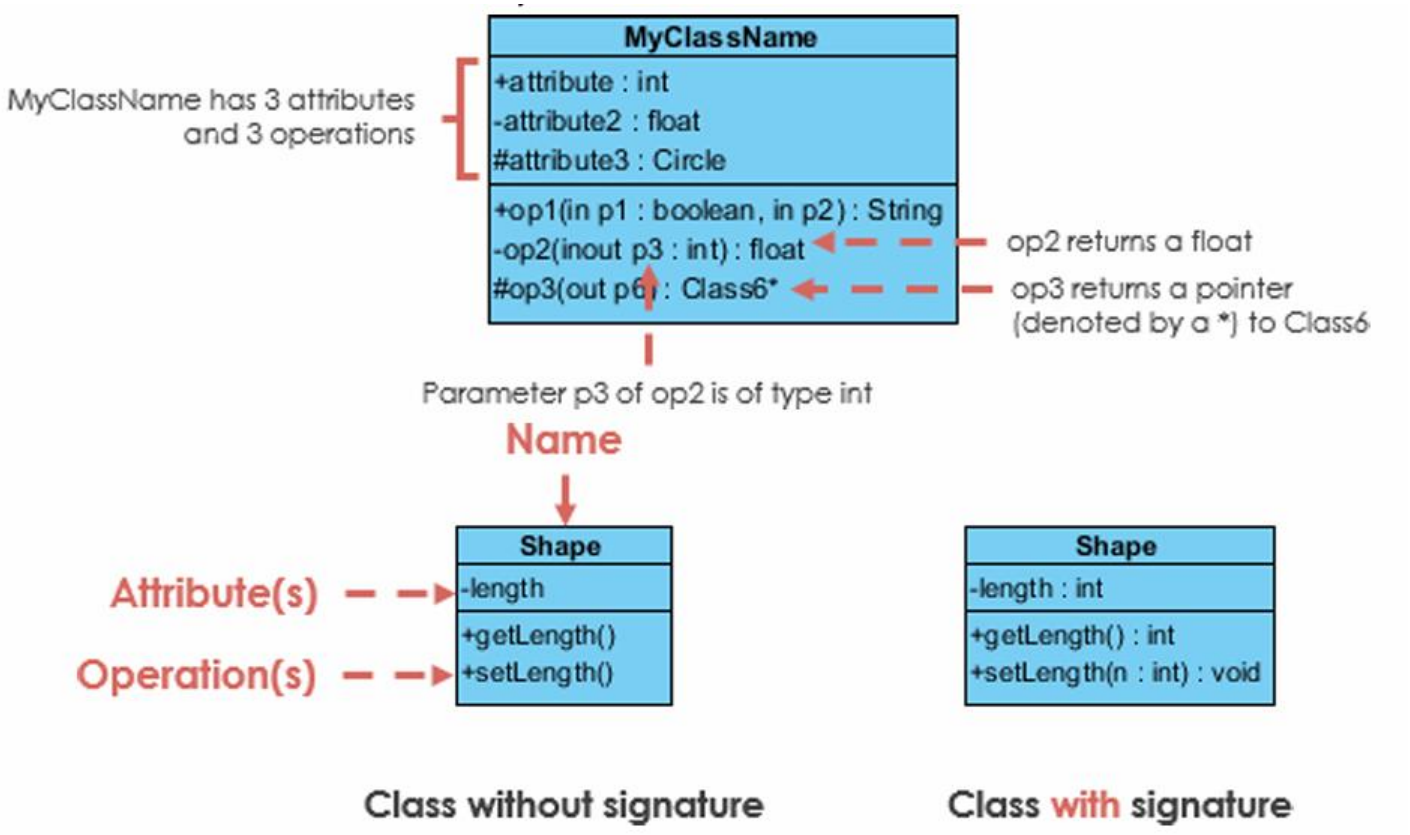
هناك أربعة مستويات للوصول في جافا:

- **الإفتراضـي (package private): يظهر العضـو داخل الكلاس التي أنشـئت فيها وداخل جميع الكلاسـات الأخرى الموجودة في نفس الحزمة. لا يمكن الوصـول إليه من خارج الحزمة.**
- **خاص (private): يقتصـر ظهور العضـو داخل الكلاس التي أنشـئت فيها فقط. لا يمكن الوصول إليه من خارج الكلاس ، حتى داخل نفس الحزمة.**
- **محمي (protected): يظهر العضـو داخل الكلاس التي أنشـئت فيها وفي جميع الفئات التي ترث منها (بما في ذلك فئات الكلاسات الكلاس الموروثة منها).**
- **عام (public): يظهر العضو في أي مكان داخل أو خارج الفئة والحزمة التي أنشئت فيها.**

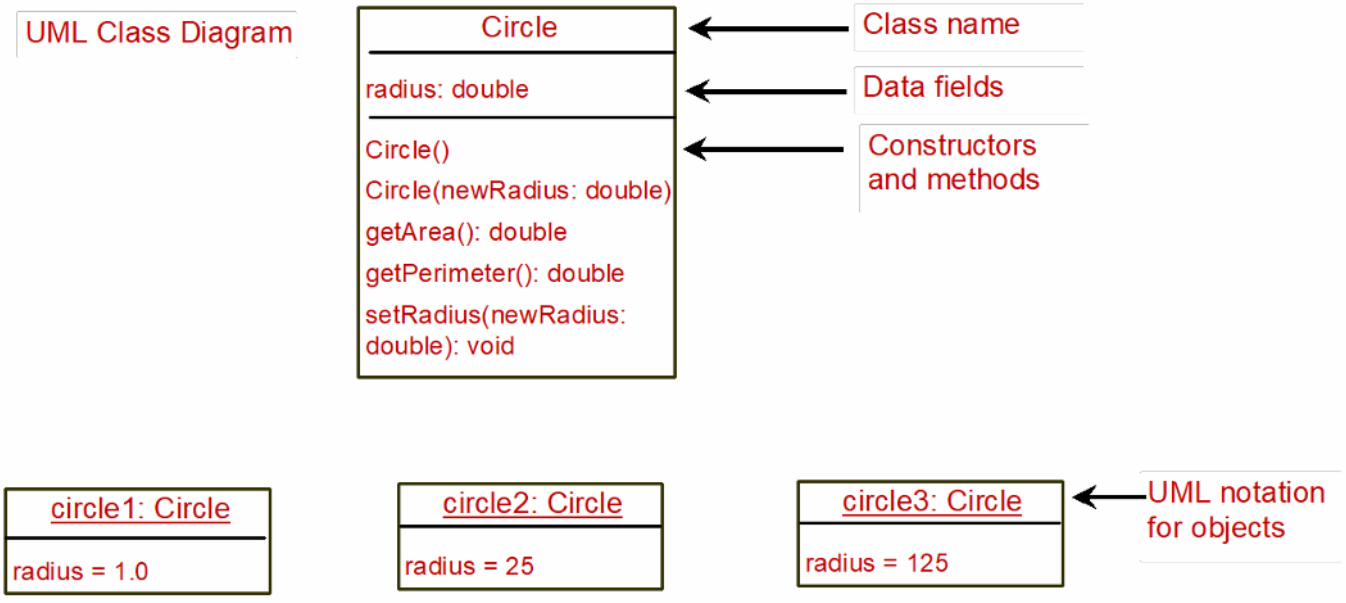
❖ Simple UML Class Diagram

Unified Modelling Language (UML) is a general-purpose visual modelling for designing systems. UML Class diagram is used to visualize classes and objects.

Unified Modelling Language (UML) هي نمذجة بصرية عامة الغرض لتصميم الأنظمة. يُستخدم رسم بياني لفئة UML لتصور الكلاسات والكائنات.



❖ An example of UML Class Diagram



Exercise: **Implement the above diagram in Java and test your program by creating objects. Search for the formula of circle area and implement it using getArea() method**

تمرين: **قم بتنفيذ الرسـم البياني أعلاه في جافا واختبر برنامجك عن طريق إنشـاء كائنات. ابحث عن صيغة مساحة الدائرة وقم بتنفيذها باستخدام طريقة getArea()**

```
4  class Circle{
5      double radius;
6      public Circle() {}
7      public Circle(double Newradius) {
8          this.radius= Newradius;
9
10     }
11
12     public double getArea() {
13         double area = 3.14 * radius*radius;
14         return area;
15     }
16
17     public double getperimeter() {
18         double perimeter =  2 *3.14 * radius;
19         return perimeter;
20     }
21
22     public void setRadius(double NewRadius) {
23         this.radius = NewRadius;
24     }
25
26
27 }
28
```

```java
public class CircleTest {

    public static void main(String[] args) {
        Circle circle1 = new Circle();
        Circle circle2 = new Circle();
        Circle circle3 = new Circle();
        circle1.setRadius(1.0);
        circle2.setRadius(25);
        circle3.setRadius(125);

        System.out.println("circle1 area : "  + circle1.getArea() + " , circle1 perimeter : "+ circle1.getperimeter() );
        System.out.println("circle2 area : "  + circle2.getArea() + " , circle2 perimeter : "+ circle2.getperimeter() );
        System.out.println("circle3 area : "  + circle3.getArea() + " , circle3 perimeter : "+ circle3.getperimeter() );


    }

}
```

output

```
circle1 area : 3.14 , circle1 perimeter : 6.28
circle2 area : 1962.5 , circle2 perimeter : 157.0
circle3 area : 49062.5 , circle3 perimeter : 785.0
```

* شوف الكود ذا اونلاين من خلال الضغط على اونلاين

اونلاين

ALZEEKA Tutorial

شروحات - مشاريع - خدمات - تصاميم

إنضم الآن

053 359 7191