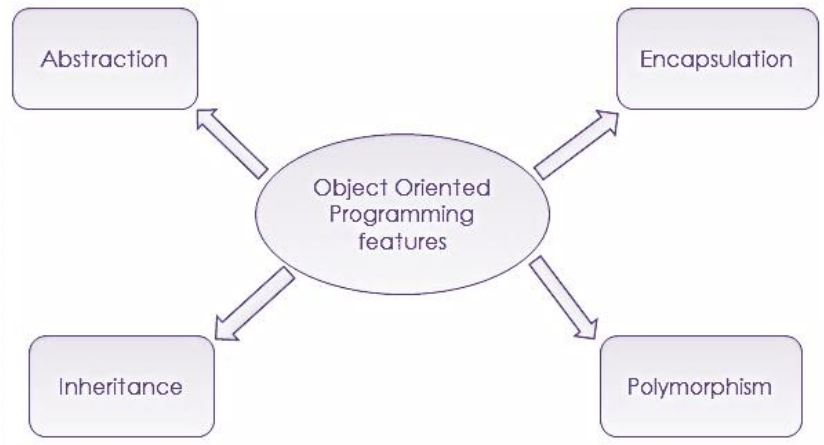
ملخص lecture 2 (Inheritance)

*الكلمات الي تحتها خط ركزون عليها

❖ Overview

- **Object-originated Programming (OOP) concepts:**
  1. **Inheritance**
  2. **Polymorphism**
  3. **Abstraction**
  4. **Encapsulation**

- **مفاهيم البرمجة المبنية على الكائنات:(OOP)**
  1. **الوراثة**
  2. **التعددية**
  3. **التجريد**
  4. **التغليف**



❖ Inheritance

- **Inheritance is The process by which one class acquires the properties (variables) and functionalities (methods) of another class.**
- **الوراثة هي العملية التي يكتسب فيها الكلاس الواحد خصائص (متغيرات) ووظائف (دوال) من كلاس آخر.**

- **The existing (or original) class is called the base-class or super-class or parent class.**

- **The newclass which inherits from the base class is called the derived class or sub-class or child class.**

- **الكلاس الأول (أو الأصلي) يسمى base-class او super-class او parent class**
- **الكلاس الجديد الذي يرث من الكلاس الأساسي يسمى class derived او sub-class او child class**

- **A derived class automatically has all the instance variables and all the methods that the base class has.**
- **The derived class can have additional methods and/or additional instance variables.**

- **يحتوي الكلاس المشتق تلقائيًا على جميع المتغيرات والدوال التي يحتوي عليها الكلاس الأساسي.**
- **يمكن للكلاس المشتق أن يحتوي على دوال إضافية و/أو متغيرات إضافية.**

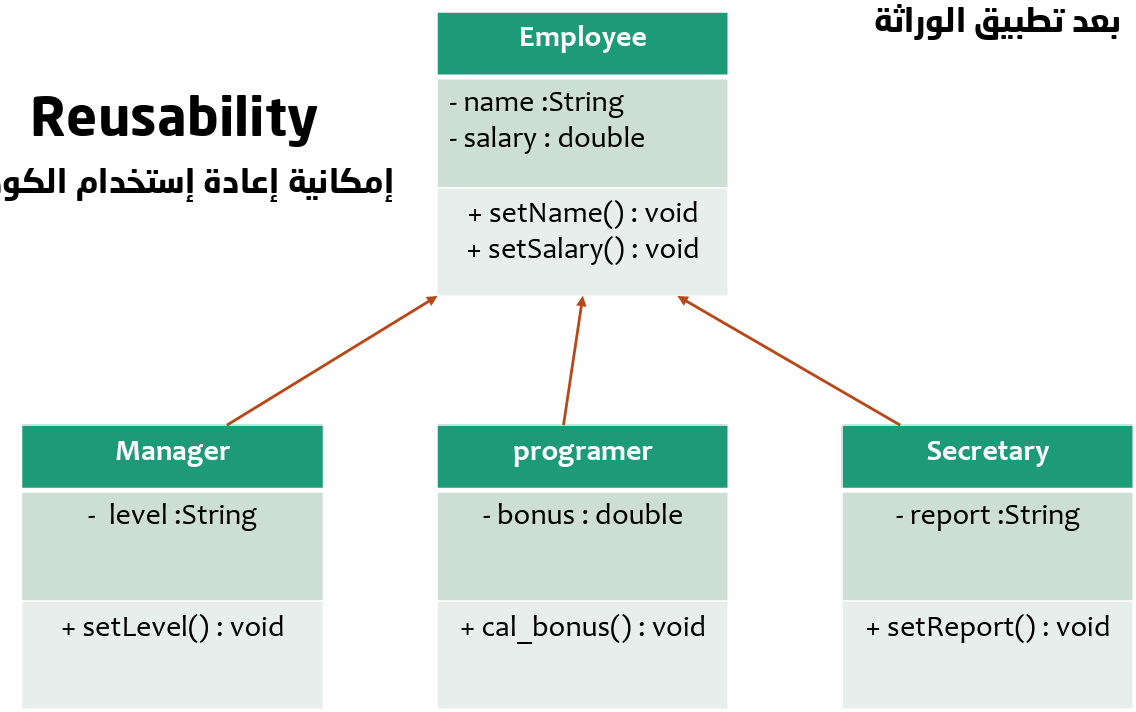
مثال قبل تطبيق الوراثة : مشروع عن شركة تحتوي على 3 classes المدير والمبرمج والسكرتير وكل كلاس له المتغيرات والدوال الخاصة فيه ، بس لاحظ هنا راح يكون في تكرار للأكواد مثلا المتغيرات name , salary متكررة في كل كلاس وبرضو الدوال setName() , setsalary() وهذا عيب بحق البرمجة ، عشان نحل المشكلة راح نستخدم الوراثة بحيث نجعل كل الصفات المشتركة بكلاس يسمى الأب او base class ونجعل بقية الكلاسات (sub classes) ترث من هذا الكلاس.

Manager	programer	Secretary
- name :String - salary : double - level :String	- name :String - salary : double - bonus : double	- name :String - salary : double - report :String
+ setName() : void + setSalary() : void + setLevel() : void	+ setName() : void + setSalary() : void + cal_bonus() : void	+ setName() : void + setSalary() : void + setReport() : void

بعد تطبيق الوراثة

Reusability

إمكانية إعادة إستخدام الكود



❖ Why we need Inheritance? لماذا نحتاج للوراثة

- **Code Reusability:** The code written in the Super-class is common to all sub-classes. Sub-classes can directly use the super-class class code.
- **إعادة استخدام الكود:** الكود المكتوب في الكلاس الأساسي مشتركة بين جميع الكلاسات الفرعية. يمكن للكلاسات الفرعية استخدام الكود الخاص بالكلاس الأساسي مباشرة.

- **Code Organization:** Inheritance helps in organizing and structuring code. This makes the code more manageable, modular, and easier to understand.
- **تنظيم الكود:** الوراثة تساعد في تنظيم وهيكلة الكود. هذا يجعل الكود أكثر إدارة وتجزئة وأسهل فهمًا.

- **Method Overriding:** Inheritance allows subclasses to override methods inherited from the superclass. This means that you can provide specialized implementations of methods in the subclasses to modify or extend the behavior defined in the superclass.
- **استبدال الدوال:** الوراثة تسمح للكلاسات الفرعية بتجاوز الدوال الموروثة من الكلاس الأساسي. هذا يعني أنه يمكنك توفير تنفيذات متخصصة للدوال في الكلاسات الفرعية لتعديل أو توسيع السلوك المحدد في الكلاس الأساسي.

- **Polymorphism:** Inheritance is closely related to polymorphism, which is another important OOP concept. It allows objects of different classes to be treated as objects of the same superclass. This means that you can write code that can work with objects of multiple subclasses without knowing their specific types. It enables you to write more flexible and extensible code.
- **التعددية:** الوراثة مرتبطة ارتباطًا وثيقًا بالتعددية، وهي مفهوم آخر مهم في البرمجة المبنية على الكائنات. تسمح بمعاملة كائنات كلاسات مختلفة ككائنات من نفس الكلاس الأساسي. هذا يعني أنه يمكنك كتابة كود يمكن أن يعمل مع كائنات من كلاســات فرعية متعددة دون معرفة أنواعها المحددة. يتيح لك ذلك كتابة كود أكثر مرونة وقابلية للتوسع.

❖ How to use Inheritance? كيف استخدم الوراثة

- The extends keyword is used for inheritance in Java. Using the extends keyword indicates you are derived from an existing class.

- بإســتخدام الكلمة extends يمكن عمل وراثة في الجافا , حيث نكتبها هذه الكلمة في الكلاس المشتق او الموروث من الكلاس الأول أو الأب.

• Syntax:

```
class derived-class extends base-class
{
   // fields and methods
}
```

❖ Java Code Example of Inheritance

```
// Base or Super-Class
class Employee {
    int salary = 60000;
}
```

1

```
// derived or Sub-Class
class Teacher extends Employee
{
    int benefits = 10000;
}
```

2

```
// Driver Class
class inheritanceTest {
   public static void main(String args[])
   {
       Teacher T1 = new Teacher();
       System.out.println("Salary : " +
       T1.salary + "\n Benefits : " +
       T1.benefits);
   }
}
```

3

Output:

Salary : 60000

Benefits : 10000

❖ Types of Inheritance انواع الوراثة

1. Single Inheritance
2. Multilevel Inheritance
3. Hierarchical Inheritance
4. Multiple Inheritance (not supported by java)

- جميع الأنواع تدعمها الجافا ما عدا النوع الأخير

1. Single Inheritance

- In single inheritance, subclasses inherit the features of one superclass.
- In the image below, class A is the base-class for the sub-class B.

- في الوراثة الفردية، ترث الكلاسات الفرعية ميزات كلاس واحد أساسي.
- في الصورة أدناه، الكلاس A هو الكلاس الأساسي للكلاس الفرعي B.

Single Inheritance

❖ **Example of Single Inheritance**

```
class A {
   public void print_Hello() {
      System.out.println("Hello");
   }
}


// sub-class B
class B extends A {
   public void print_Hail() {
      System.out.println("Hail");
   }
}


// driver class
public class Main {
   public static void main(String[] args) {
      B object = new B();
      object.print_Hello();
      object.print_Hail();
      object.print_Hello();
   }
}
```

Output:
Hello
Hail
Hello

2. Multilevel Inheritance

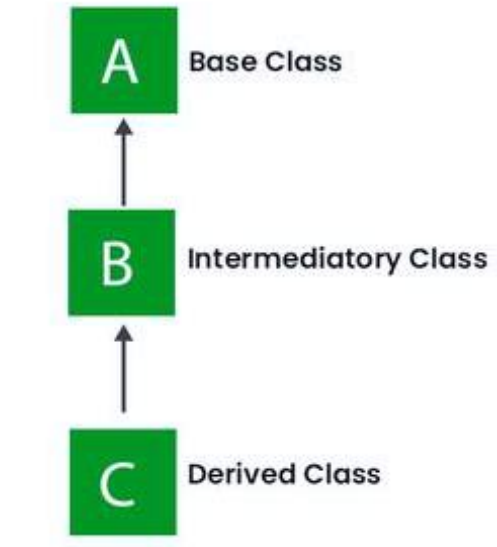
- In Multilevel Inheritance, a derived class will be inheriting a base class, and also acts as the base class for other classes.

- في الوراثة المتعددة المستويات، سيكون الكلاس المشتق يرث الكلاس الأساسي، ويعمل أيضًا ككلاس أساسي للكلاسات الأخرى.

- In the image, class A serves as a base class for the derived class B, which in turn serves as a base class for the derived class C.

- In Java, a class cannot directly access the grandparent's members.

- في الصـورة، يعمل الكلاس A ككلاس أسـاسـي للكلاس المشـتق B ، الذي بدوره يعمل ككلاس أساسي للكلاس المشتق C
- في جافا ، لا يمكن للكلاس الوصـول مباشـرة إلى أعضـاء الجد الأكبر اذا كان من نوع private.



Multilevel Inheritance

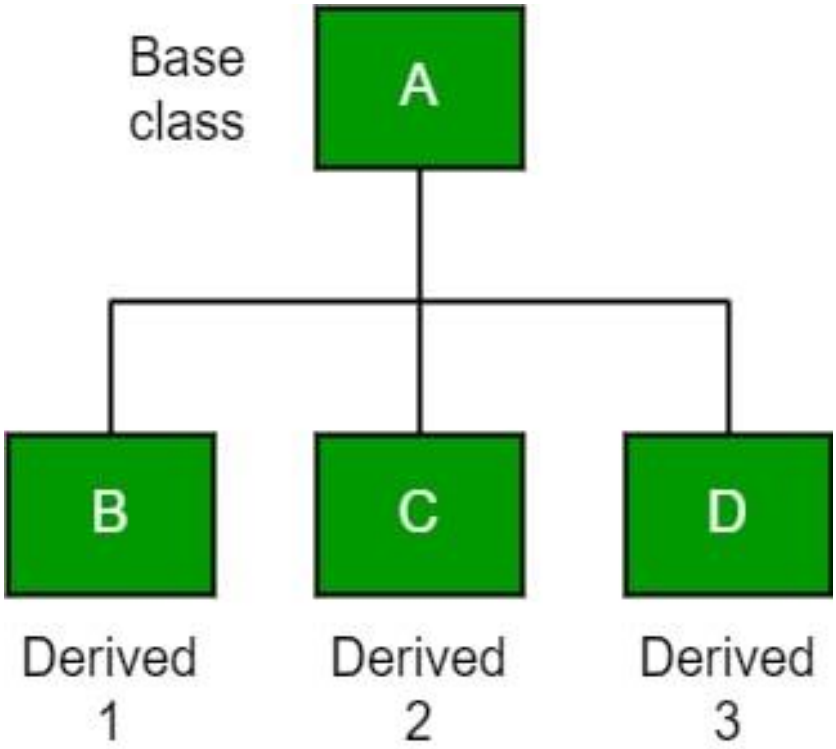
```
// base-class A
class A {
 public void print_Hello() {
  System.out.println("Hello");
 }
}


// sub-class B
class B extends A {
 public void print_Hail() {
  System.out.println("Hail");
 }
}


// sub-class C
class C extends B {
 public void print_X() {
  System.out.println("X");
 }
}



// driver class
public class Main
{
        public static void main(String[] args) {
          C object = new C();
        object.print_Hello();
        object.print_Hail();
        object.print_X();

        }
}
```

Output:
Hello
Hail
X

3. Hierarchical Inheritance

- In Hierarchical Inheritance, one class serves as a superclass (base class) for more than one subclass.
- في الوراثة الهرمية، يعمل كلاس واحد ككلاس أساسي (كلاس أب) لأكثر من كلاس فرعي.

- In the image, class A serves as a base class for the derived classes B, C, and D.
- في الصورة، يعمل الكلاس A ككلاس أساسي للكلاسات المشتقة B و C و D.



```java
//**********************************************************
// base-class A
class A {
 public void print_A() {
  System.out.println("Class A");}
}

// sub-class B
class B extends A {
 public void print_B (){
  System.out.println("Class B");}
}

// sub-class C
class C extends A {
 public void print_C () {
  System.out.println("Class C");}
}

// sub-class D
class D extends A {
 public void print_D() {
  System.out.println("Class D");}
}

// driver class
public class Main
{
    public static void main(String[] args) {
        A obj_a = new A();
        obj_a.print_A();

        B obj_b = new B();
        obj_b.print_A();
        obj_b.print_B();

        C obj_c = new C();
        obj_c.print_A();
        obj_c.print_C();

        D obj_d = new D();
        obj_d.print_A();
        obj_d.print_D();

    }
}
```

```
Class A
Class A
Class B
Class A
Class C
Class A
Class D
```

- لو ودك تجرب الكود اونلاين اضغط هنا **اونلاين**

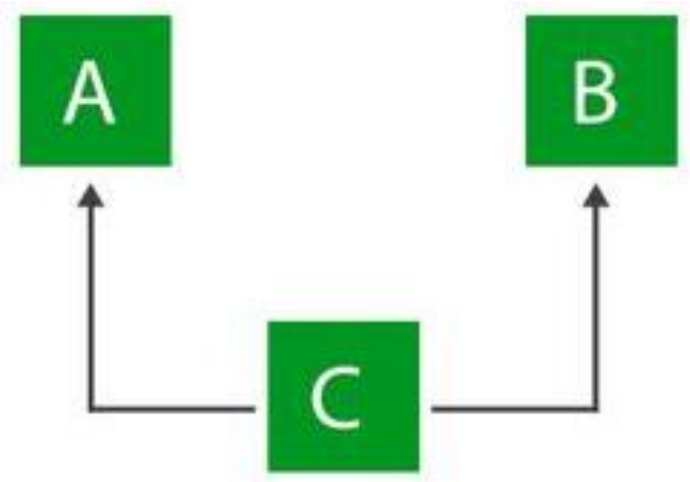
053 359 7191

4. Multiple Inheritance

- In Multiple inheritances, one class can have more than one superclass and inherit features from all parent classes.
- في الوراثة المتعددة، يمكن لفئة واحدة أن تحتوي على أكثر من فئة أساسية وترث الميزات من جميع الفئات الأم.

- Note that Java does not support multiple inheritances with classes. In Java, we can have multiple inheritances only using Interfaces.
- في يرجى ملاحظة أن جافا لا تدعم الوراثة المتعددة مع الفئات. في جافا، يمكننا أن نمتلك الوراثة المتعددة فقط باستخدام interfaces.

- In the image below, Class C is derived from interfaces A and B.
- في الصورة، يرث الكلاس c من الكلاسين A , B



Multiple Inheritance

❖ Example of Multiple Inheritance

```java
interface One { public void print_Hello();}
interface Two { public void print_Test(); }

interface Three extends One, Two { public void print_Yesss();}

class Child implements Three {
    public void print_Hello() {System.out.println("Hello");}
    public void print_Test()  {System.out.println("Test"); }
    public void print_Yesss() {System.out.println("Yesss");}
}



public class Main
{
    public static void main(String[] args) {
        Child c = new Child();
        c.print_Hello();
        c.print_Test();
        c.print_Yesss();
    }
}
```

```
Hello
Test
Yesss
```

- لو ودك تجرب الكود اونلاين اضغط هنا **اونلاين**

❖ Overriding

- Although a derived class inherits methods from the base class, it can change or override an inherited method if necessary
    - In order to override a method definition, a new definition of the method is simply placed in the class definition, just like any other method that is added to the derived class
- على الرغم من أن الفئة المشتقة ترث دوال من الفئة الأساسية، إلا أنها تستطيع ذلك تغيير أو استبدال الدالة الموروثة إذا لزم الأمر
- لكي تستبدل تعريف الدالة، يتم وضع تعريف جديد للدالة ببساطة في تعريف الصف، تمامًا مثل أي دالة أخرى يتم إضافتها إلى الصف المشتق.

- Also, given the following method header in a base case:

  private void doSomething()
- The following method header is valid in a derived class:

  public void doSomething()

- However, the opposite is not valid. That is using a public modifier for the base class and using a private modifier in the derived class.

- ومع ذلك ، فإن العكس غير صحيح. وهذا يعني استخدام public modifier للفئة الأساسية واستخدام private modifier في الفئة المشتقة.

```
8  // base-class A
9  class A {
10  private void print() {
11    System.out.println("Class A");}
12 }
13
14 // sub-class B
15 class B extends A {
16  public void print (){
17    System.out.println("Class B");}
18 }
19
```

❖ Notes on Inheritance (super)

- *Are Superclass's Constructor Inherited? No. They are not. They are invoked explicitly (using the super keyword) or implicitly.
  - هل يتم توريث ال Constructor للفئة الأساسية؟ لا، لا يتم ذلك. يتم استدعاؤها بشكل صريح (باستخدام كلمة super) .

- A call to the base class constructor can never use the name of the base class, but uses the keyword super instead. A call to super must always be the first action taken in a constructor definition.
  - لا يمكن ابدا أن يستخدم استدعاء constructor الفئة الأساسية بإسم الفئة الأساسية ، ولكنه يستخدم بدلاً من ذلك الكلمة الأساسية super يجب أن يكون استدعاء super دائمًا أول إجراء يتم اتخاذه في تعريف constructor.

- If a derived class constructor does not include an invocation of super, then the no-argument constructor of the base class will automatically be invoked
  - إذا لم يتضمن constructor الكلاس المشتق استدعاءً لـــ super ، فسيتم تلقائيًا استدعاء constructor الكلاس الأســاســي الذي لا يحتوي على وسـيط او بما يسـمى default constructor

- This can result in an error if the base class has not defined a no-argument constructor
  - يمكن أن يؤدي ذلك إلى حدوث خطأ إذا لم يتم تعريف كلاس الأســاس ب no-argument constructor , يعني اجباري انه الكلاس الأب يكون فيه default constructor

❖ Invoking a Base class Constructor

- A derived class uses a constructor from the base class to initialize all the data inherited from the base class.
- In order to invoke a constructor from the base class, it uses a special syntax:

- الكلاس المشتق او الإبن يستخدم constructor الكلاس الاساسي لكي يقوم بتهيئة جميع البيانات الموروثة من الكلاس الأساسي.
- ولكي تستدعي constructor من الفئة الأساسية، استخدم super(,); شوف المثال:

```
class baseClass {
 int p1,p2;

    public baseClass(int p1 , int p2){
        this.p1=p1;
        this.p2=p2;
    }

}


class derivedClass extends baseClass {
     double p3;
     public derivedClass(int p1 , int p2 , double p3){
        super(p1,p2);
        this.p3=p3;
    }
}

// driver class
public class Main
{
    public static void main(String[] args) {
    derivedClass o = new derivedClass(4,5,3.5);


    }
}
```

- A In the above example, super(p1, p2); is a call to the base class constructor
- في المثال super(p1, p2); قامت بإستدعاء base class constructor

❖ Where to use super keyword? متى نستخدم كلمة السوبر

- We can summarize the cases where to use the super keyword as follows:
  1. To call methods of the superclass that is overridden in the subclass.
  2. To access attributes (variables) of the superclass if both superclass and subclass have attributes with the same name.
  3. To explicitly call superclass no-arg (default) or parameterized constructor from the subclass constructor.

1. لاستدعاء دوال الفئة الأساسية التي تم تجاوزها في الفئة فرعية
2. للوصول إلى السمات (المتغيرات) للفئة الأساسية إذا كانت كل من الفئة الأساسية والفئة الفرعية لديهما سمات بنفس الاسم.
3. للإستدعاء الصريح ل constructor الكلاس الأساسي سواء كان default او لديه parameter

❖ The this Constructor

- Within the definition of a constructor for a class, this can be used as a name for invoking another constructor in the same class
- The same restrictions on how to use a call to super apply to the this constructor

- في تعريف مُنشئ لفئة ما، يمكن استخدام كلمة "this" لاستدعاء مُنشئ آخر ضمن نفس الفئة.
- تسري نفس القيود المفروضة على استخدام استدعاء "super" على استدعاء "this" للمُنشئ.

```
9  class MyClass {
10     private int value;
11
12     public MyClass() {
13         this(42);
14     }
15
16     public MyClass(int value) {
17         this.value = value;
18     }
19 }
```

- If it is necessary to include a call to both super and this, the call using this must be made first, and then the constructor that is called must call super as its first action.
- عندما يكون لديك حاجة لاستدعاء كل من super و this في البناء(constructor) ، يجب أن يتم استدعاء this أولاً، وبعد ذلك يجب أن يقوم constructor الذي يتم استدعاؤه بدوره بدعوة superكأول إجراء له.

```
14
15  class MyClass extends My {
16    private int value;
17
18    public MyClass(int value) {
19        this();
20     this.value = value;
21    }
22
23    public MyClass() {
24      super();
25     System.out.print("java");
26    }
27  }
28
```

- *هاااااام ، إذا حاولت وضع استدعاء لكل من super() و this في نفس (constructor)، فسيحدث خطأ (error) يجب أن يتم استدعاء super() أو this() في constructor مختلفة.

An example of using this

```
Class Student {

 int ID; String name, address; float fee;

 Student (int ID, String name, String address){
        this.ID = ID;
        this.name = name;
        this.address = address;
 }

  Student (int ID, String name, String address, float fee){
        this(ID, name, address); // using the first constructor
        this.fee = fee;
  }

 void display(){
        System.out.println(ID+" "+name+" "+address+" "+fee);
 }

} // end class
```

```
Class Test {
  public static void main(String args[]){
        Student s1 = new Student(20222020, “Ali”, “Hail”);
        Student s2 = new Student(20219012, “Omar”, “Jeddah”, 12000f);
        s1.display();
        s2.display();
   }
}
```

Output:

```
20222020 Ali Hail 0.0
20219012 Omar Jeddah 12000.0
```

❖ Notes on Inheritance (final)

- It the modifier final is placed before the definition of a class, then that class may not be used as a base class to derive other classes
- If the modifier final is placed before the definition of a method, then that method may not be redefined in a derived class
- إذا تم وضـع final قبل تعريف الكلاس، فلا يمكن اسـتخدام ذلك الكلاس ككلاس أسـاسـي لاشتقاق كلاسات أخرى. *لا يمكن الوراثة منه

- إذا تم وضع final قبل تعريف الدالة ، فلا يمكن إعادة تعريف تلك الدالة في الكلاس المشتق. *لا يمكن عمل override لتلك الدالة منه.

- In summary:
- The final keyword is a non-access modifier used for classes, attributes and methods, which makes them non-changeable (impossible to inherit or override)
- (final) هو يعتبر non-access modifier، ويُستخدم مع الفئات والسمات والدوال لجعلها غير قابلة للتغيير (لا يمكن توريثها أو إعادة تعريفها).

An example of using **final**

```
final int AGE = 32; // create a final variable
AGE = 45; // you cannot change the value of AGE!
```

```
class A {
 public final void display() {
        System.out.println("This is a final method.");}
}

class Test extends A {
 // you cannot override a final method
 public final void display() {
        System.out.println("This is another final method.");}
}
```

```
final class Person { int ID; String name; }
// you cannot extend a final class
Class Student extends Person { …. }
```

❖ The Object class

- In Java, the Object class is the root of the class hierarchy. It is the superclass of all other classes in Java, and every class implicitly extends Object if it doesn't explicitly extend another class.

- في لغة جافا، تُعتبر فئة Object بمثابة جذر التسلسل الهرمي للفئات. وهي تمثل الفئة العليا لجميع الفئات الأخرى في جافا، وكل فئة ترث من Object class في حال لم ترث صـــراحةً من فئة أخرى.

- The Object class provides several methods that are available to all Java objects. Here are some of the commonly used methods of the Object class:

- يوفّر Object class بالعديد من الدوال المتاحة لجميع كائنات جافا. فيما يلي بعض الدوال المُستخدمة بشكلٍ شائع في Object class:

  - toString()
  - getClass()
  - equals(Object ob)

❖ toString()

- **toString() method returns a string representation of the object.**
- **By default, the toString() method returns a string that includes the class name, an "@" symbol, and the object's hash code; something like this className@hashCode. It is often overridden in subclasses to provide a more meaningful string representation. See the example below.**

- **تُرجع طريقة toString() تمثيلاً نصياً للكائن.**
- **بشكلٍ افتراضي، تُرجع طريقة toString() نصاً يتضمن اسم الصنف ، ورمز "@"، ورمز تجزئة الكائن؛ شيء مثل className@hashCode. غالباً ما يتم تجاوز هذه الدالة في الكلاسات الفرعية لتوفير تمثيل نصي أكثر وضوحاً. راجع المثال أدناه.**

```
public class Person {
    private String name;
    private int age;

    // ...constructor and other code...

    public String toString() {
        return "Person [name=" + name + ", age=" + age + "]";
    }
}

// Usage:
Person person = new Person("John", 25);
System.out.println(person.toString()); // Output: Person [name=John, age=25]
```

*ملاحظات

- * اذا كانت دالة toString() ليست مكتوبة سيكون ال output بهذا الشكل packageName.Person@1234abcd

- * وايضا يمكن استدعاء دالة toString() فقط بإسم ال object مثل كذا

```
Person person = new Person("John", 25);
System.out.println(person);
```

❖ getClass()

- This method returns the Class object that represents the runtime class of the object. The class provides various methods to obtain information about the class, such as its name, superclass, interfaces implemented, etc.
- تُرجع هذه الدالة كائن Class object الذي يمثل كلاس وقت التشغيل للكائن. يوفر Class دوال متنوعة للحصول على معلومات حول الكلاس، مثل اسمه، كلاسه الأب ، و interfaces التي يُنفّذها، وغيرها.

```
public class Person {
    // ...other code...

    public void printClassName() {
        System.out.println("Class name: " + getClass().getName());
    }
}

// Usage:
Person person = new Person();
System.out.println(person.getClass()); // Output: class Person
System.out.println(person.getClass().getName()); // Output: Person
person.printClassName(); // Output: Class name: Person
```

- Here are some common methods available in the Class class in Java:

1.getName(): Returns the fully qualified name of the class.

تُرجع هذه الطريقة الاسم الكامل للكلاس، والذي يتضمن اسم الحزمة الذي ينتمي إليه الصنف.

مثال: إذا كان لدينا صنف اسمه "Student" موجود في الحزمة"com.example" ، فإن استدعاء getName() على كائن من هذا الصنف سيُرجع السلسلة."com.example.Student"

2.getSimpleName(): Returns the simple name of the class.

تُرجع هذه الطريقة الاسم البسيط للصنف، بدون اسم الحزمة.

مثال: إذا كان لدينا صنف اسمه "Student" موجود في الحزمة"com.example" ، فإن استدعاء getSimpleName() على كائن من هذا الصنف سيُرجع السلسلة."Student"

3.getDeclaredMethods(): Returns an array of Method objects representing all the declared methods in the class.

تُرجع هذه الطريقة مصفوفة من كائنات Method التي تمثّل جميع الطرق المُعلنة في الصنف، بما في ذلك الطرق الخاصة والمحميّة.

يمكن استخدام هذه الطريقة للحصول على معلومات حول الطرق الموجودة في الصنف، مثل اسم الطريقة، ونوع الإرجاع، وأنواع المتغيرات.

4.getMethods(): Returns an array of Method objects representing all the public methods, including inherited methods.

تُرجع هذه الطريقة مصفوفة من كائنات Method التي تمثّل جميع الطرق العامة في الصنف، بما في ذلك الطرق الموروثة.

تختلف هذه الطريقة عن getDeclaredMethods() في أنها لا تُرجع الطرق الخاصة أو المحميّة، ولكنها تُرجع الطرق الموروثة من الأصناف العليا.

5.getDeclaredFields(): Returns an array of Field objects representing all the declared fields in the class.

تُرجع هذه الطريقة مصفوفة من كائنات Field التي تمثّل جميع الحقول المُعلنة في الصنف، بما في ذلك الحقول الخاصة والمحميّة.

يمكن استخدام هذه الطريقة للحصول على معلومات حول الحقول الموجودة في الصنف، مثل اسم الحقل، ونوعه، وقيمة الحقل الثابت.

6.getFields(): Returns an array of Field objects representing all the public fields, including inherited fields.

تُرجع هذه الطريقة مصفوفة من كائنات Field التي تمثّل جميع الحقول العامة في الصنف، بما في ذلك الحقول الموروثة.

تختلف هذه الطريقة عن getDeclaredFields() في أنها لا تُرجع الحقول الخاصة أو المحميّة، ولكنها تُرجع الحقول الموروثة من الأصناف العليا.

7.getDeclaredConstructors(): Returns an array of Constructor objects representing all the declared constructors in the class.

تُرجع هذه الطريقة مصفوفة من كائنات Constructor التي تمثّل جميع المُنشئات المُعلنة في الصنف.

يمكن استخدام هذه الطريقة للحصول على معلومات حول المُنشئات الموجودة في الصنف، مثل عدد المُعاملات وأنواعها.

8.getConstructors(): Returns an array of Constructor objects representing all the public constructors in the class.

تُرجع هذه الطريقة مصفوفة من كائنات Constructor التي تمثّل جميع المُنشئات العامة في الصنف.

تختلف هذه الطريقة عن getDeclaredConstructors() في أنها لا تُرجع المُنشئات الخاصة أو المحميّة.

9.isInstance(Object obj): Determines if the specified object is an instance of the class.

تحدّد هذه الطريقة ما إذا كان الكائن المُعطى هو مثيل للصنف المُمثل بواسطة كائن.Class

تُستخدم هذه الطريقة عادةً للتحقق من نوع كائن في وقت التشغيل.

❖ equals(Object obj)

- **This method is used to compare two objects for equality. By default, the equals() method compares object references for equality (they refer to the same memory location or not). Subclasses often override this method to define their own notion of equality based on the object's state.**

- **تُستخدم هذه الدالة لمقارنة كائنين من حيث التساوي. بشكلٍ افتراضي، تقارن طريقة equals() مراجع الكائنات من حيث التساوي (أي ما إذا كانت تشير إلى نفس موقع الذاكرة أم لا). غالباً الكلاسات الفرعية تعمل override لهذه الدالة لتحديد مفهومها الخاص للتساوي بناءً على حالة الكائن.**

```
public class Person {
    private String name;
    private int age;

    public boolean equals(Object obj) {
        if (this == obj) return true;

        if (obj == null || getClass() != obj.getClass()) return false;

        Person otherPerson = (Person) obj; // casting obj to be from a person class
        // below we check the name and age of obj with that we are comparing with
        return name.equals(otherPerson.name) && age == otherPerson.age;
    }
}

// Usage:
Person person1 = new Person("John", 25);
Person person2 = new Person("John", 25);
System.out.println(person1.equals(person2)); // Output: true
```

ALZEEKA Tutorial

شروحات - مشاريع - خدمات - تصاميم

إنضم الآن

053 359 7191