جامعة طرابلس كلية تقنية المعلومات

البرمجة الشيئية

Object Oriented Programming (with Java)

رمز المقرر: ITGS211

محاضرة: التجريد Abstraction

الفصل الدراسي: ربيع 2022

التجريد Abstract

التجريد : هو عملية تقوم بتعريف السلوك والمهام التي يقوم بها الكائن (object) واهمال التفاصيل غير اللازمة.

مثال :

جهاز الراديو لديه هوائي وزر تحكم بالصوت وزر للفتح والإغلاق، ولا نحتاج لمعرفة كيف يلتقط جهاز الراديو الموجات من الإذاعة وكيف يقوم بتحويلها إلى إشارات رقمية ثم يخرجها في شكل صوت، فهذه التفاصيل مخفيه عنا ولا تحتاجها.

إخفاء التفاصيل الداخلية تعتبر ميزة فبهذه الطريقة يمكن لكل شخص أن يستخدم جهاز الراديو ولا يقتصر الأمر على الفنيين فقط.

2

Abstract Class

- في الـclasses السابقة كان بإمكاننا اشتقاق كائنات objects منها والتعامل معها، هذه الـclasses تسمى Concrete class.
- هناك نوع آخر من الـclasses لا يُسمح باشتقاق كائنات منه بل يتم فقط اتخاذه كأب (Superclass) وهذه تسمى الـclasses بالمجردة Abstract class.
- يتم تعريف الـclass من النوع المجرد بإضافة كلمة Abstract قبل الكلمة المحجوزة class .

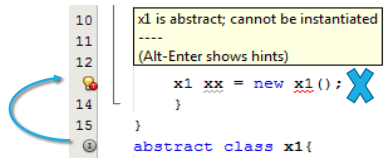
```
public abstract class Employee {
  //....
}
```

- يتم تعريف الـclass من النوع المجرد بإضافة كلمة Abstract قبل الكلمة الغرض منه أن يتم توفير class عام يمكن للأبناء subclasses أن ترثه ، ويشتركوا جميعاً في تصميم واحد، لكن لكل واحد منهم طريقة خاصة للتطبيق والعمل.

4

Abstract Class

- الـclass المجرد يحتوي على دالة مجردة أو أكثر. وهذه الدوال يجب أن يتم عمل override لها في الابناء Subclasses لكي تصبح الابناء concrete classes.
- المتغيرات و الدوال غير المجردة في الـClass المجرد تخضع لقواعد الوراثة العامة عند توريث هذا الصنف للأبناء.
- محاولة اشتقاق كائن من الـclass المجرد ينتج عنها Compilation error.



- مثال لاستخدام الصنف المجردAbstract Class:

يمكننا كتابة Abstract Class لتمثيل الأشكال الهندسية ثنائية الأبعاد، ثم نشتق منه concrete classes للمربع و الدائرة و المستطيل.

5

Abstract Class

من المفترض أن الصنف المجرد Abstract class يحوي على الأقل دالة من النوع المجرد Abstract method.

ملاحظة: إن لم يحوي دالة مجردة فلن يظهر خطأ ولكن ما الفائدة منه؟!

```
public class Abs_JavaApp {
    public static void main(String[] args) {
      B b=new B();
      b.printA();
    }

}
abstract class A{
    int M;
}
class B extends A{
    void printA(){
        System.out.println("B");
    }
}
```

```
run:
B
BUILD SUCCESSFUL (total time: 0 seconds)
```

6

الدالة المجردة Abstract method

من المفترض أن الصنف المجرد Abstract class يحوي على الأقل دالة من النوع المجرد Abstract method.

والدالة المجردة هي دالة يتم تعريفها فقط في الـclass دون عمل تطبيق لها، حيث يتم عمل التطبيق من خلال الابناء subclass مستخدمين مفهوم overridden.

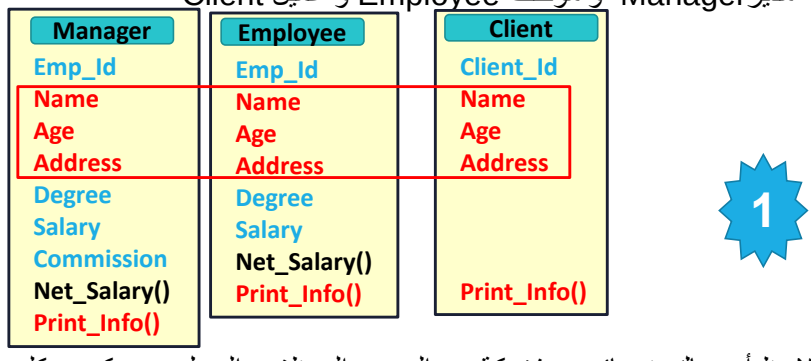
أي يوضع method signature فقط متبوعاً بـفاصلة منقوطة (;) أي بدون method body كالتي:

```
public abstract class Employee {
//---
Public abstract void cal();
}
```

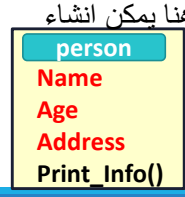
7

مثلاً: عند عمل برنامج لشركة ما، وكانت الكائنات التي نتعامل معها ثلاث:-

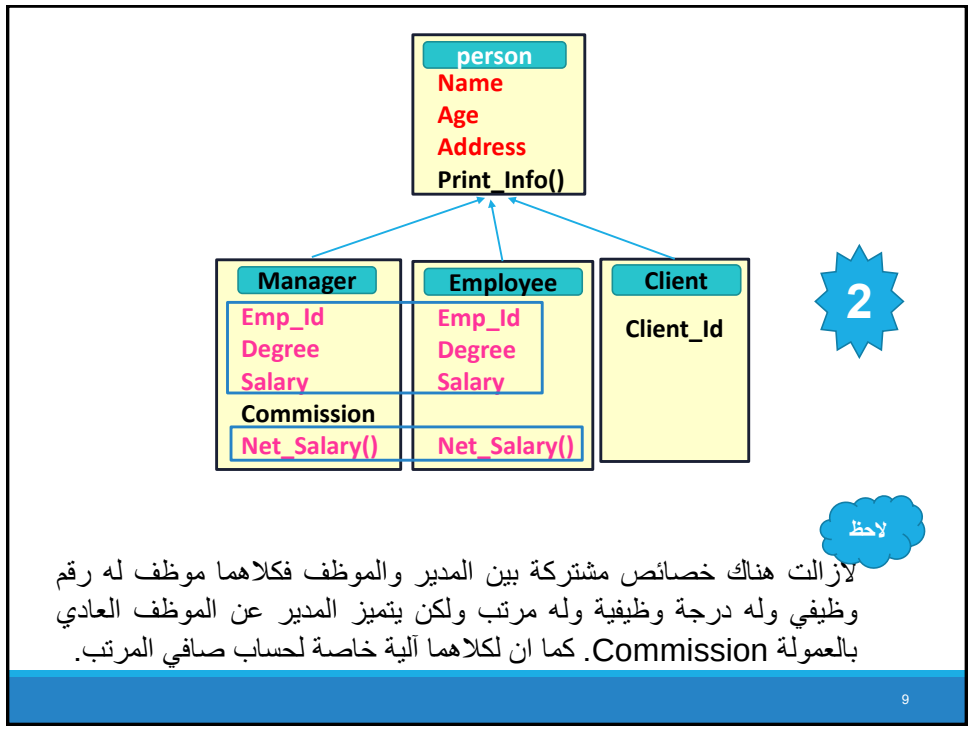
- مدير Manager و موظف Employee و عميل Client

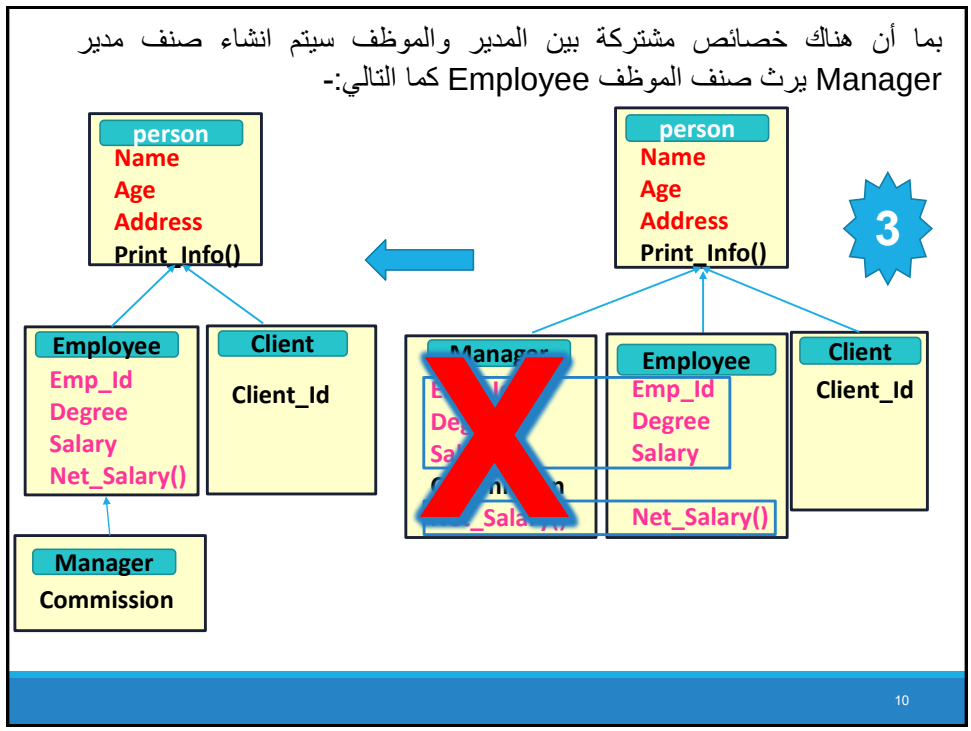


لاحظ أن هناك خصائص مشتركة بين المدير والموظف والعميل وهي كونهم كلهم شخص له اسم Name وعمر Age وعنوان Address من هنا يمكن انشاء صنف شخص person class يحوي الخصائص المشتركة وترثه الأصناف الثلاث كالتالي:



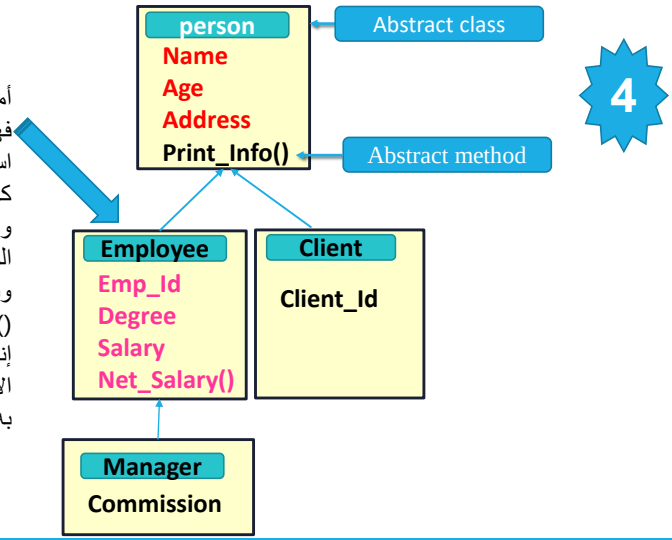
8


person
Name
Age
Address
Print_Info()
Manager
Emp_Id
Degree
Salary
Commission
Net_Salary()
Employee
Emp_Id
Degree
Salary
Net_Salary()
Client
Client_Id
2
لاحظ
لازالت هناك خصائص مشتركة بين المدير والموظف فكلاهما موظف له رقم وظيفي وله درجة وظيفية وله مرتب ولكن يتميز المدير عن الموظف العادي بالعمولة Commission. كما ان لكلاهما آلية خاصة لحساب صافي المرتب.
9


بما أن هناك خصائص مشتركة بين المدير والموظف سيتم انشاء صنف مدير Manager يرث صنف الموظف Employee كما التالي:-
person
Name
Age
Address
Print_Info()
Employee
Emp_Id
Degree
Salary
Net_Salary()
Client
Client_Id
Manager
Commission
person
Name
Age
Address
Print_Info()
3
Employee
Emp_Id
Degree
Salary
Net_Salary()
Client
Client_Id
10

الكلاس Person هو كلاس مجرد يُمنع استخدامه مباشرة أي يُمع انشاء كائنات objects منه، ولكن ترثه الكلاسات client و Employee والدالة Print_Info() هي دالة مجردة يجب إعادة إنشائها مرة أخرى بنفس الاسم في الـ class الأبناء client و Employee ولكن مع توضيح آلية العمل لكل منهما.

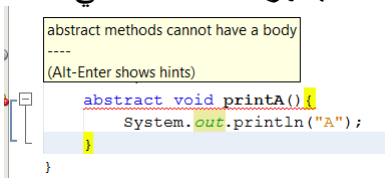
أما الكلاس Employee فهو كلاس غير مجرد ويُمكن استخدامه مباشرة بإنشاء كائنات objects منه.
والكلاس manager يرث الكلاس Employee وبالتالي يرث الدالة Net_Salary() مع إعادة إنشائها مرة أخرى بنفس الاسم ولكن بآلية عمل خاصة به.



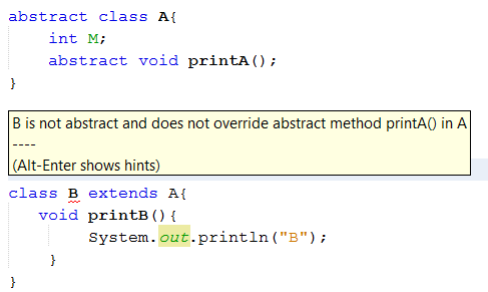
11

الدالة المجردة Abstract method

لو حاولت تكتب آلية عمل الدالة المجردة فسيظهر لك خطأ كالتالي:



لو كتبت الدالة المجردة بشكل صحيح ولكن لم نعمل لها overriding فسيظهر لك خطأ كالتالي:



12

الدالة المجردة Abstract method

- إذا كان الـclass المجرد abstract class فهذا يعني أنه لابد من وراثة هذا الـclass. والدالة المجردة Abstract method لابد من عمل دالة بنفس اسمها في الـSub class. أي overriding كما في المثال التالي:

```
abstract class x1{
  abstract  void show();
  }
final class x2 extends x1 {
    @Override
       void show(){
       System.out.print("Hi");
    }
```

إذا وضعت الأمر abstraction أمام دالة في الـclass لابد من وضع الأمر abstraction أمام الـclass الذي يحوي هذه الدالة وإلا سيظهر خطأ كالتالي :

```
x1 is not abstract and does not override abstract method show() in x1
----
(Alt-Enter shows hints)
class x1{
  abstract  void show();
```

13

```
public class Abstruct1 {
   public static void main(String[] args) {
      second s = new second();
      s.show();
      s.showtext();
         }
}
 abstract class first{
     abstract void show();
     void showtext()
     {
        System.out.println("first");
     }
 }
class second extends first{
   @Override
   void show(){
      System.out.println("second");
   }
}
```

second

first

14

ماهو الخطأ عند تعريف كائن من class مجرد (Abstruct class)؟؟

```
public class Abstruct1 {
   public static void main(String[] args) {
      second s = new second();
      first  f =new first();
      s.show();
      s.showtext();
         }
}
   abstract class first{
      abstract void show();
      void showtext()
      {
         System.out.println("first");
      }
   }
   class second extends first{
      @Override
      void show(){
         System.out.println("second");
      }
   }
```

```
4        second s = new second();
5        first f = new first();
6
7
8
```

first is abstract; cannot be instantiated

(Alt-Enter shows hints)

15

إذا كان الـsuperclass مجرد ويحوى دالة مجردة Abstruct ،، ولكن لم يتم إنشاء دالة بنفس الأسم في subclass فستظهر رسالة خطأ كالتالي:

```
3        public static void main(String[] args) {
4              second s = new second();
5            s.show();
6            s.showtext();
7                }
8    }
9        abstract class first{
10           abstract void show();
11           void showtext()
12           {
13
14
15
16       class second extends first{
17           void showAB(){
18              System.out.println("second");
19           }
20       }
```

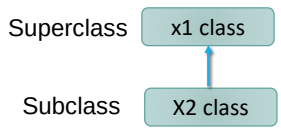
second is not abstract and does not override abstract method show() in first
---
(Alt-Enter shows hints)

16

Example :

قم بإنشاء class باسم x1 مع منع إستعماله مباشرةً، علماً بأنx1 يحتوي على دالة مجردة max تعمل على تحديد أكبر رقم من بين ثلاث أرقام. وx1 هو اب لـx2 الذي يحوي دالة numbers التي تعمل على استقبال ثلاث أرقام، ولاترجع أي قيمة. وعليك منع توريث x2 لأي classes أخرى.
في x2 قم بالتركيب على الدالة المجردة (max) بحيث تعمل على تحديد أكبر رقم من ثلاث أرقام .

أكتب الجمل البرمجية لإستعمال الـ class x2 من أجل طباعة قيمة العدد الأكبر وذلك من خلال استدعاء الدالة max , numbers.



17

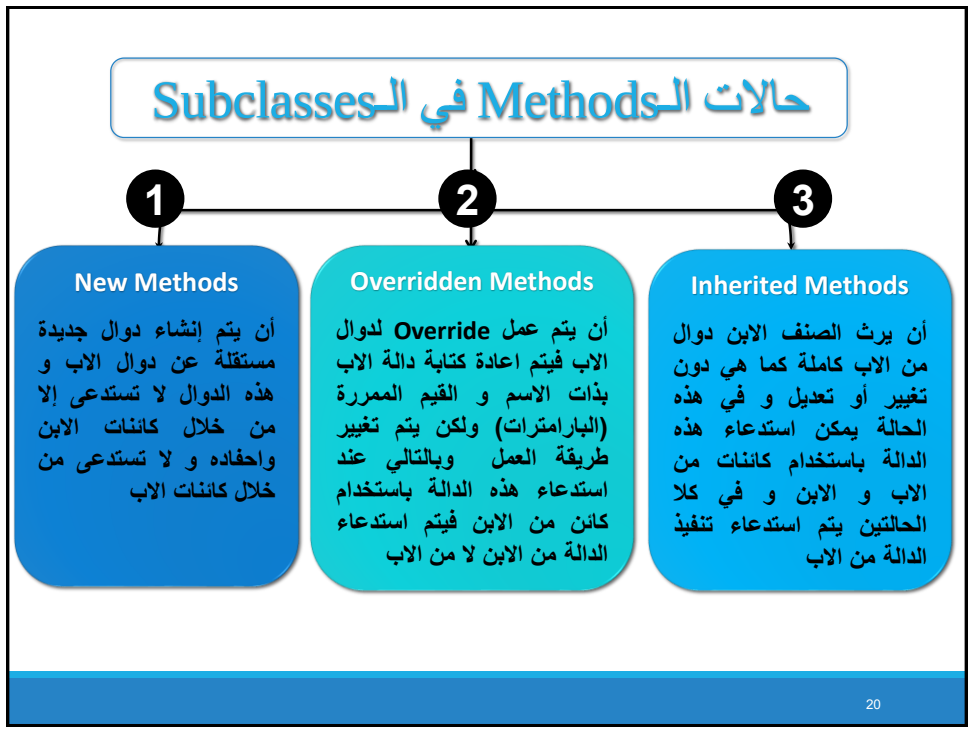
```
import java.util.Scanner;
public class Myclass {
    public static void main(String[] args) {
        x2 z= new x2();
        z.numbers();
       System.out.println("max is : " + z.max(z.n1,z.n2,z.n3));
    }
}
```

```
abstract class x1
{
  abstract  int max (int a, int b , int c);
 }
```

18

```
final class x2 extends x1 {
    int n1, n2, n3;
    void numbers(){
        Scanner in = new Scanner (System .in);
        n1= in.nextInt();
        n2= in.nextInt();
        n3= in.nextInt();
    }
    @Override
    int max (int a, int b, int c) {
        int result ;
        if (a>b && a>c)
            result  = a;
        if(b>a && b>c)
            result =b;
        else
            result =c;
        return result ;
     }
}
```

19



20

إنشاء Abstract Superclass Employee

```
public abstract class Employee{

  private String firstName;
  private String lastName;
  private String ID;

public Employee( String firstName, String lastName, String ID )
     {
        this.firstName = firstName;
       this.lastName = lastName;
        this.ID = ID;
    } // end three-argument Employee constructor

    // set firstName name
    public void setfirstName( String firstName )
    {
       this.firstName = firstName;
    } // end method setfirstName

    // return firstName name
    public String getfirstName()
    {
       return firstName;
    } // end method getfirstName

    // set last name
    public void setLastName( String lastName )
    {
       this.lastName = lastName;
    } // end method setLastName

    // return last name
    public String getLastName()
    {
       return lastName;
    } // end method getLastName
```

انتبه قمنا بتعريف هذا الصنف class بالكلمة abstract وبالتالي من المفترض أن يحتوي على دالة واحدة على الأقل من النوع abstract

21

إنشاء Abstract Superclass Employee

```
    // set last name
    public void setLastName( String lastName )
    {
       this.lastName = lastName;
    } // end method setLastName

    // return last name
    public String getLastName()
    {
       return lastName;
    } // end method getLastName

    // set ID
    public void setID( String ID )
    {
       this.ID = ID;
    } // end method setID

    // return ID
    public String getID()
    {
       return ID;
    } // end method getID

    // return information of Employee object
    public String info()
    {
       return "The information is: "+ getfirstName()+ getLastName()+ getID();
    } // end method info

    // abstract method overridden by subclasses
    public abstract double earnings(); // no implementation here
} // end abstract class Employee
```

انتبه: هذه الدالة المجردة (أي مجردة من التطبيق) ويتم تعريفها فقط من خلال method signature (نوعها، اسمها، قيمها).
وتذكر أن من دونها فإن هذا الصنف لا يعتبر مجرد

22

إنشاء Concrete Subclass **SalariedEmployee**

```
public class SalariedEmployee extends Employee
   {
      private double weeklySalary;

      // four-argument constructor
      public SalariedEmployee( String firstName, String lastName, String ID,
                               double weeklySalary )
      {
         super( firstName, lastName, ID ); // pass to Employee constructor
         this.weeklySalary = weeklySalary ;
      } // end four-argument SalariedEmployee constructor

      // set salary
      public void setWeeklySalary( double weeklySalary )
      {
        this.weeklySalary = weeklySalary < 0.0 ? 0.0 : weeklySalary;
      } // end method setWeeklySalary

      // return salary
      public double getWeeklySalary()
      {
         return weeklySalary;
      } // end method getWeeklySalary
```

استدعاء صريح للـ constructor الخاص بالأب يقيم ممررة عبر الكائن المشتق من الابن.

23

إنشاء Concrete Subclass **SalariedEmployee**

هنا قمنا بعمل override للدالة earnings من خلال:
تغيير طريقة حساب المرتب بينما احتفظنا بتوقيع الدالة كما هو)

```
   // calculate earnings; override abstract method earnings in Employee
   public double earnings()
   {
      return getWeeklySalary();
   } // end method earnings

   // return String representation of SalariedEmployee object
   public String info()
   {
      return  super.info()+ getfirstName()+ getLastName()+ getID() ;
   } // end method info


} // end class SalariedEmployee
```

هنا قمنا بعمل override للدالة info من خلال:
تغيير طريقة إرجاع البيانات واستدعاء دالة الاب أيضا من خلال super.info()

24