
جامعة طرابلس
كلية تقنية المعلومات
البرمجة الشيئية
Object Oriented Programming
(with Java)
ITGS211
المحاضرة السابعة
الفصل الدراسي: ربيع 2022
1


خصائص البرمجة الشيئية
1 . تمثل نظام محكاة للعالم الحقيقي.
2 . نستطيع من خلالها إخفاء تفاصيل البرمجة والتي يطلق عليها
التغليف (Encapsulation)
3 . نستطيع من خلالها إعادة استخدام الكود مرة أخرى عن طريق
الوراثة (Inheritance)
4 . يمكن من خلالها استخدام اسم واحد يمثل عدة وظائف مختلفة
(Function Overloading , Polymorphism)
2

Classes

- الـ class: هو أساس البرمجة الشيئية، والذي يُبنى عليه البرنامج.
- عن طريق الـ class: يُمكن للمُبرمج استحداث عدد n من المتغيرات تُسمى objects يُعطى كل object بيانات مُحددة خاصة به.
- الـ class: بمثابة نموذج فيه:
  - وصف للبيانات وشكلها ومواصفاتها.
  - العمليات التي سيؤديها Object على البيانات خاصته.
- تعريف الـClass يكون على الشكل:

```
AccessModifier  class  ClassName
{
   [fields declaration]          \\ الخصائص
   [methods declaration] \\ العمليات
}
```

3

تصميم الـ Class

عند تصميم Class يجب تحديد ما يلي:

- البيانات (data) التي يجب أخذها بالاعتبار.
- الاجراءات أو الأفعال (actions) التي يجب انجازها.
- البيانات (data) التي يمكن تعديلها (modify).
- البيانات (data) التي يجب أن يُسمح بالوصول إليها.
- أي قواعد بشأن الكيفية التي ينبغي بها تعديل البيانات.

عادةً يتم تصميم الـ Class بالاستعانة بمُخطط لغة النمذجة الموحدة (UML).

4

الخصائص Attributes

- عناصر البيانات للـclass تُحدد الكائن ليكون instance من الـclass.

- الخصائص (attributes) يجب أن تكون ومُميزة للـclass ومُعرِفة له.
مثلا: يتم تعريف المستطيل عن طريق:
  - الطول length.
  - العرض width.

- يتم الوصول إلى الخصائص (attributes) عن طريق الدوال methods داخل الـclass.
  - كقراءة الطول والعرض ثم حساب محيط ومساحة المستطيل.

5

الدوال Methods

- الدوال تحقق العمليات التي يمكن أن تتم على الخصائص (البيانات) داخل الـ Class.
- الدوال التي يُمكن استخدامها من قِبل classes أخرى يتم جعلها عامة public.

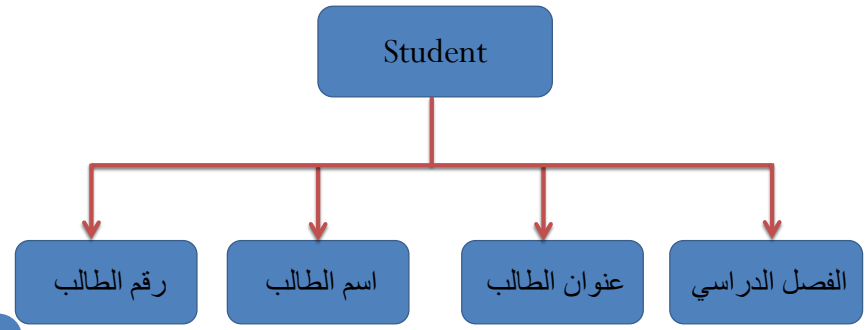
- شكل رأس الدالة (Method header)

```
AccessModifier ReturnType MethodName(Parameters)
{
  //Method body.
}
```

6

مثال Student Class

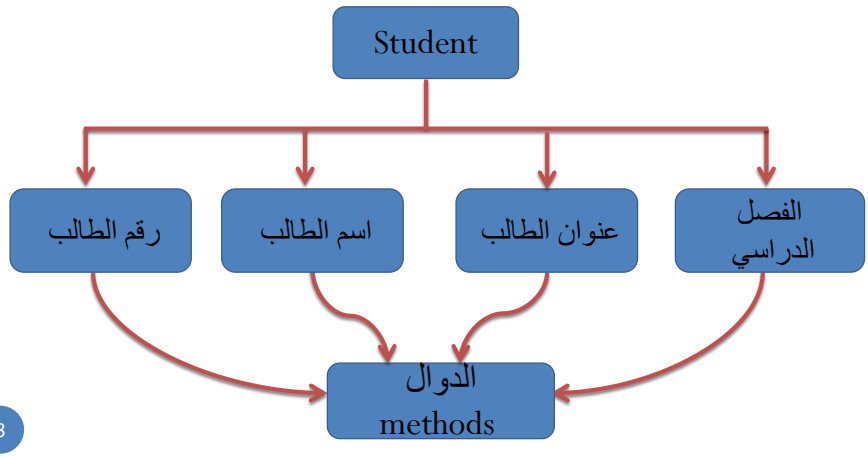
في الـStudent Class بياناته سوف تكون بيانات الطالب العامة مثل رقم الطالب، اسم الطالب، عنوان الطالب، الفصل الدراسي،... الخ



7

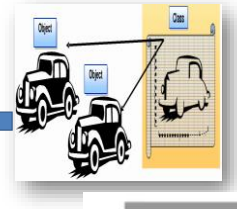
مثال Student Class

في الـStudent Class الدوال التي تحقق العمليات التي تتم على الطالب داخل مثل عملية إضافة أو حذف أو تعديل بيانات.



8

مثال Car class



Type="BMW"
Model=2015
Price=10000
MilesDrive=105
Owner="Hussein"

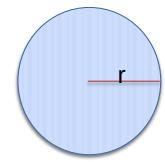
GetPrice()
GetOwner()

كيف نمثله برمجياً

```
public class Car {
      String Type;
      int Model;
      double Price;
      double MilesDrive;
      String Owners;
 double GetPrice(){
      double NewPrice=Price-(MilesDrive*100);
      return NewPrice;}
String GetOwner(){
      return owners;}
}
```

```
Car car2= new Car();
      car2.Type="BMW";
      car2.Model=2015;
      car2.Price=10000;
      car2.MilesDrive=105;
      car2.Owner="Hussein";
```

مثال Circle class




```
public class Circle {
 // بيانات الدائرة
     public double x, y; // مركز الدائرة
     public double r;    //نصف قطر الدائرة radius
// دوال (Methods) لارجاع محيط ومساحة الدائرة
     public double circumference() {
                return 2*3.14*r;
          }
          public double area() {
                return 3.14 * r * r;
          }
     }
```

fields declaration

Method Body

methods declaration

10

في هذا الـ class تم الإعلان عن متغيرات x,y,z وتم انشاء دالة بالاسم Printxyz وفيها تتم طباعة المتغيرات x,y,z.

```
public class class1 {
    int x, y,z;

    void Printxyz( )
    {
       System .out . println("x= " + x);
       System .out . println("y= " + y);
       System .out . println("z= " + z);
     }
}
```

fields declaration

methods declaration

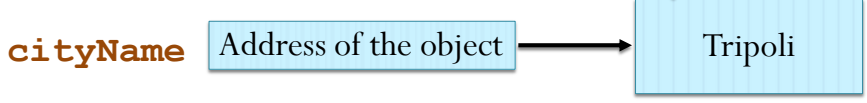
11

Classes and Instances

كما سبق وأن تعلمنا أن الـ reference variable يحتوي عنوان الـobject .

```
String cityName = "Tripoli"
```

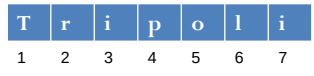
الـ object الذي يحتوي على السلسلة "Tripoli"



الدالة `length()` للـ class `String` تُرجع قيمة صحيحة تساوي طول السلسلة. فمثلاً:

```
int stringLength = cityName.length();
```

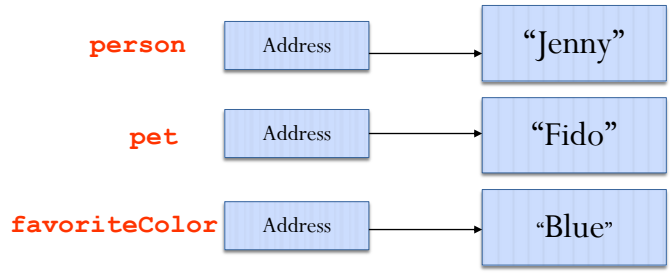
المتغير `stringLength` سيحمل القيمة 7، حيث أن السلسلة "Tripoli" تحوي 7 أحرف. كما يلي:



12

- إذاً عدة objects يمكن انشائها من class. و كل object مستقل عن غيره (independent). على سبيل المثال:

```
String person = "Jenny";
String pet = "Fido";
String favoriteColor = "Blue";
```



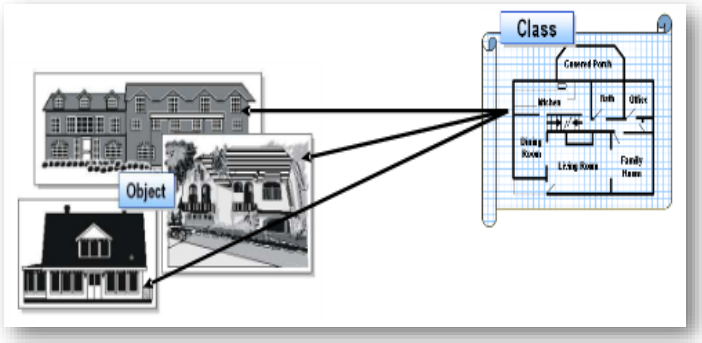
13

انشاء objects من الـ class

- إذًا لكي نتمكن من استخدام الكلاس Class لابد أن نُنشئ منه كائنات (Objects).
- الكائنات تُنشأ باستخدام الكلمة المحجوزة **new** التي تحجز مكان في الذاكرة لكائن من نوع الكلاس. كالتالي:
- بالعودة للمثال بالشريحة 9 بإمكاننا تكوين كائنات من الكلاس Circle مثلاً
- Circle aCircle = **new** Circle() ;
- Circle bCircle = **new** Circle() ;
- aCircle و bCircle يشير إلى كائنات Circle.

14

- كل instance من class يحتوي على بيانات مختلفة.
- الـ instances كلها مشتركة في نفس التصميم (design).
- ولكل instance كل الصفات (attributes) والدوال (methods) التي تم تعريفها في الـ class.
- يتم تعريف Classes لتمثيل مفهوم(concept) واحد.



15

كيفية استعمال الـــ class1

```
package oop1;
public class Oop1 {
    public static void main(String[] args) {
        class1 x =new class1();
        x.a=20;
        x.b=40;
        x.c=30;
        x.printabc();
    }
}
```

```
class class1{
    int a,b,c;
    void printabc() {
        System.out.println("a="+a);
        System.out.println("b="+b);
        System.out.println("c="+c);
    }
}
```

في هذا المثال:
-بعد إنشاء الكلاس class1 وتحديد متغيراته ودواله.
-تم استعمال الـclass1 داخل الدالة الرئيسية وتم الإعلان عن متغير باسم x من نوع class1 وفي هذه الحالة لا يسمى متغير بل يسمى كائن object مع استعمال كلمة new لإنشاء الكائنx.

a=20
b=40
c=30

16

مثال: اكتب برنامج العمليات الحسابية الأربع باستخدام الـ class

```
class mathoperation  {
      double  result;
 double sum( double v1,double v2)   {
            result =v1+v2;
             return result;
       }

   double subtract ( double v1,double v2)  {
             result =v1-v2;
             return result;
        }

 double multiple( double v1,double v2)  {
            result =v1*v2;
            return result;
       }
 double divisor ( double v1,double v2)  {
           result =v1/v2;
           return result ;
      }
}
```

1) إنشاء الـ class

17

2) البرنامج الرئيسي وتعريف الكائن mathobject من الـclass mathoperation

```
package oop2
import java.util.Scanner;
public class oop2 {
    public static void main(String[] args) {
        mathoperation mathobject =new  mathoperation ();
        Scanner input=new Scanner(System.in);
        double N1=input.nextDouble();
        double N2=input.nextDouble();
       System.out.println("\n Sum is :"+ mathobject.sum(N1,N2));
        System.out.println("\n subtract is :"+ mathobject.subtract(N1,N2));
        System.out.println("\n multiple is :"+ mathobject.multiple(N1,N2));
        System.out.println("\n divisor is :"+ mathobject.divisor(N1,N2));
    }
}
```

18

مثال: اكتب برنامج يعمل على طباعة مضروب عدد يتم ادخاله قِبل من المستخدم.

```
import java.util.Scanner;
public class oop3 {
    public static void main(String[] args) {
      Fact fact =new  Fact ();
       Scanner input=new Scanner(System.in);
       int N1=input.nextInt();
      System.out.println("Fact is :"+ fact.f2(N1));
    }
}
class Fact {
        static int f2(int x) {
            int f=1,i;
            for (i=1;i<=x;i++)
                f=f*i;
            return f ;
        }
    }
```

19

```
run:
5
Fact is :120
BUILD SUCCESSFUL
```

ملاحظة: سبق وأن اتفقنا أن كل **ملف** يمكن أن يحوي أكثر من Class.

ولكن Class واحد وواحد فقط يكون عام public. وهو الذي يُكون اسم الملف بامتداد (.java)

وإن حاولت أن تنشيء Public Class آخر لن يسمح بذلك وسيسألك هل ترغب في تغيير اسم الملف؟؟

```
5   package myfirstclass;
6
7   public class MyFirstclass {
8       public static void main(String[] args) {
9           class1 x =new class1();
10          x.a=20;
11          x.b=40;
12          x.c=30;
13          x.printabc();
14      }
15  }
16  public class class1{
17      Rename file to class1.java
18          void printabc() {
19              System.out.println("a="+a);
20              System.out.println("b="+b);
21              System.out.println("c="+c);
22      }
23  }
24
```

myfirstclass.class1

Notifications | Output - MyFirstclass (run)

دالة البناء Construction

وهي دالة في الـclass، تحمل نفس اسم الـclass ولا تُرجع قيمة، تنفذ تلقائياً عند استعمال الـclass، أي تُستدعى عند انشاء الـobjects وتجدها بعد كلمة new وظيفتها تسجيل أي قيم ابتدائية أو تعريف أي متغيرات أو أي شروط ابتدائية.

إذا لم يقم المبرمج بكتابتها ضمن الـclass فسيتم استدعاء دالة بناء Constructor افتراضية.

فمثلاً: عندما نقوم بإنشاء كائن Student من class1 بواسطة الكلمة new فإن دالة البناء الـconstructor يتم استدعائها مباشرة.

```
class1 student =new class1( );
```

دالة البناء

21

مثال: استعمل الـclass Student للإعلان عن كائنين (Ahmed and Khalid) لحساب متوسط الدرجات لكل كائن في مادتين ثم طباعة رقم الطالب والمعدل داخل البرنامج؟

```
class student {
        int id, oop, math;
        int  average() {
             int x=(oop+ math)/ 2;
              return x;
           }
}
```

1) إنشاء الـكلاس student ويتكون من ثلاث متغيرات ودالة .

```
package oop1;
public class Oop1 {
    public static void main(String[] args) {
        Student  ahmed =new Student();
        Student  khalid =new Student();
    }
}
```

class

2) إنشاء object من الـ class

دالة البناء

لاحظ عدم ذكر دالة البناء student() عند إنشاء الـ class المُسمى student، ومع ذلك تم استدعائها هنا عند انشاء الكائن ahmed وكذلك الكائن khalid.

22

3) استخدام الكائن داخل البرنامج

```
ahmed .id =0234567;
ahmed .oop =80;
ahmed .math =60;
System .out.println("ahmed id =" + ahmed.id +" average="+ ahmed.average());

khalid .id =4536871;
khalid .oop =70;
khalid .math =80;
System .out.println("khalid id =" + khalid.id +" average="+ khalid.average());
```

23

البرنامج كامل للمثال السابق

```
package oop_example;
public class OOP_Example {
    public static void main(String[] args) {
        Student ahmed =new Student();
        Student khalid =new Student();
        ahmed .id =0234567;
        ahmed .oop =80;
        ahmed .math =60;
        System.out.println("ahmed id =" + ahmed.id +" ,average="+ ahmed.average());
        khalid.id =4536871;
        khalid.oop =70;
        khalid.math =80;
        System.out.println("khalid id =" + khalid.id +" , average="+ khalid.average());
    }
}

class Student {
    int id,oop,math;
    int  average(){
            int x=(oop+ math)/ 2;
            return x;
        }
}
```

```
Output - OOP_Example (run)
run:
ahmed id =80247 ,average=70
khalid id =4536871 , average=75
BUILD SUCCESSFUL (total time: 0 seconds)
```

24

- **إذًا دالة البناء** هي من الخصائص التي تدعمها البرمجة الشيئية داخل الـ class وهي دالة كأي دالة **ولكن** تنفذ تلقائياً دون استدعاء أي بمجرد تعريف كائن object من الـ class.
- **ويجب** أن يكون لها نفس اسم الـ class الذي تقع ضمنه، وبما أنها دالة **فقد يكون لها مدخلات** parameters **أو لا يكون.**

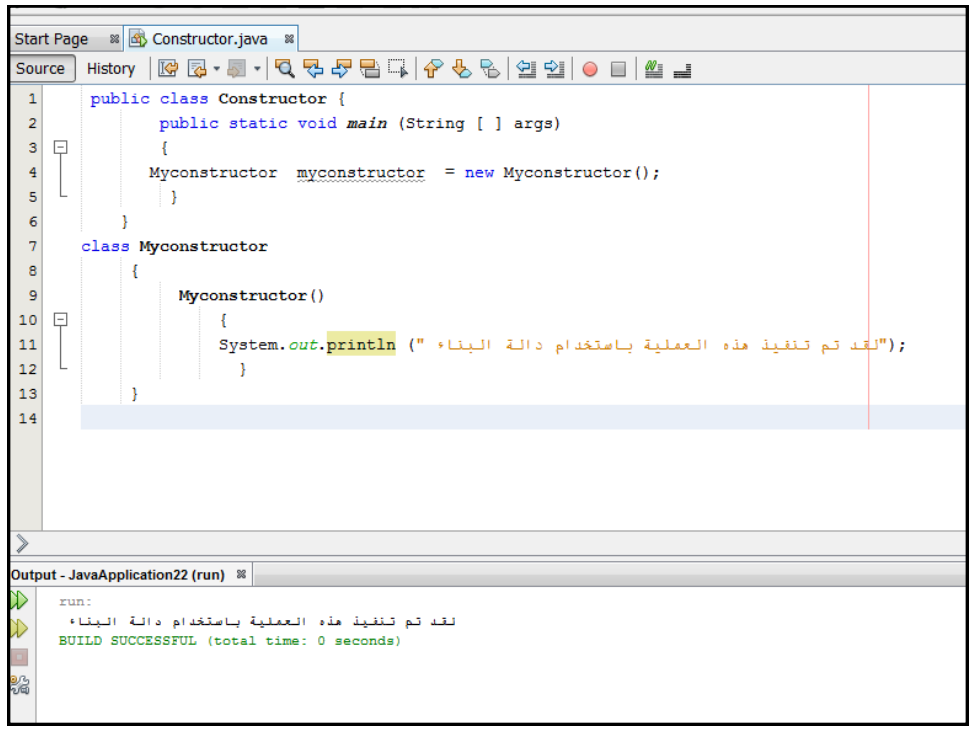
```
Class Myconstructor
  {
   Myconstructor(){
    System.out.println(" لقد تم تنفيذ هذه العملية باستخدام دالة البناء")
    }
}
```

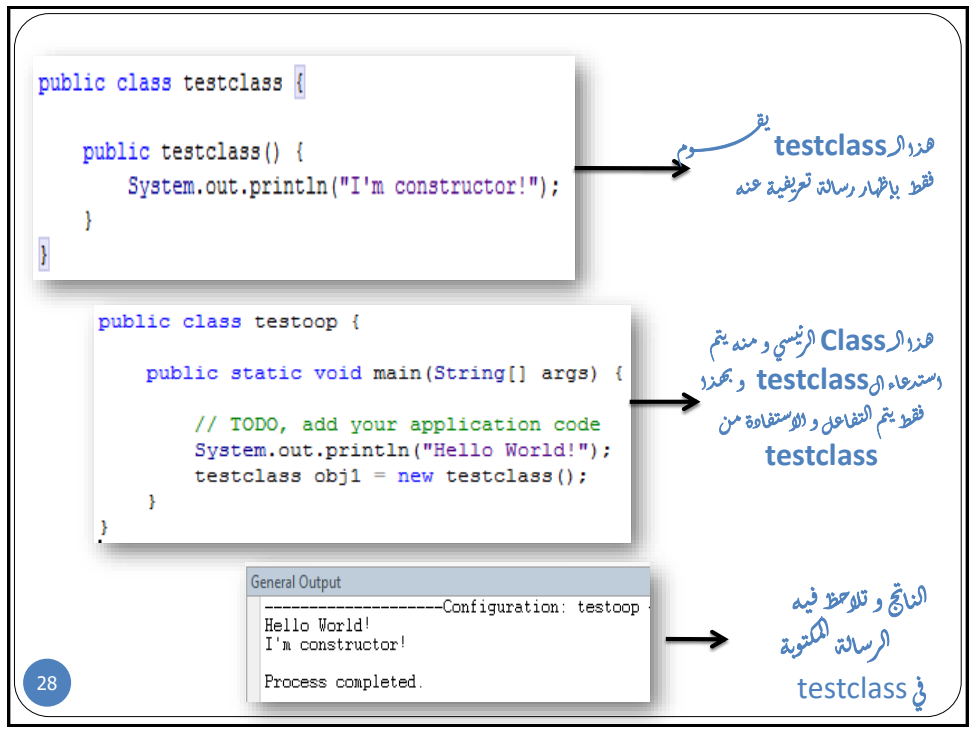
25

استخدام دالة البناء

```
public class Constructor {
    public static void main (String [ ] args)
     {
    Myconstructor  myconstructor  = new Myconstructor();
     }
  }
class Myconstructor
   {
      Myconstructor()
        {
        System.out.println (" لقد تم تنفيذ هذه العملية باستخدام دالة البناء");
         }
}
```

26


Start Page
Constructor.java
Source
History
public class Constructor {
public static void main (String [ ] args)
{
Myconstructor myconstructor = new Myconstructor();
}
}
class Myconstructor
{
Myconstructor()
{
System.out.println (" لقد تم تنفيذ هذه العملية باستخدام دالة البناء");
}
}
Output - JavaApplication22 (run)
run:
لقد تم تنفيذ هذه العملية باستخدام دالة البناء
BUILD SUCCESSFUL (total time: 0 seconds)


public class testclass {
public testclass() {
System.out.println("I'm constructor!");
}
}
هذا الـ testclass يقوم
فقط بإظهار رسالة تعريفية عنه
public class testoop {
public static void main(String[] args) {
// TODO, add your application code
System.out.println("Hello World!");
testclass obj1 = new testclass();
}
}
هذا الـ Class الرئيسي و منه يتم
استدعاء الـ testclass و بهذا
فقط يتم التفاعل و الاستفادة من
testclass
General Output
----------Configuration: testoop
Hello World!
I'm constructor!
Process completed.
الناتج و تلاحظ فيه
الرسالة المكتوبة
في testclass
28

استخدام دالة البناء بدون مُدخلات (بارامترات)

```
public class Constructor1 {
    public static void main (String [ ] args){
      Myrectangle  rect  = new Myrectangle();
       System.out.println ("width="+ rect.width+"\t height="+ rect.height);
        }
  }

class Myrectangle {
   double width,height;
     Myrectangle(){
       width=10;
       height=2 ;
     }
  }
```

إنشاء object من الـ class وفيه استدعاء دالة البناء

إنشاء دالة البناء داخل الـ class ولها نفس اسمه وليس لها مُدخلات.

29

استخدام دالة البناء ولها مدخلات

```
public class Constructor1 {
    public static void main (String [ ] args)
     {
     Myrectangle  rect  = new Myrectangle(20,4);
      System.out.println ("width="+ rect.width+"\t height="+ rect.height);
       }
  }
class Myrectangle
   { double width,height;
     Myrectangle(double w , double h )
      {
        width=w;
        height=h ;
      }
  }
```

إنشاء object من الـ class وفيه استدعاء دالة البناء ولها مدخلات

إنشاء دالة البناء داخل الـ class ولها نفس اسمه ولها مُدخلات.

Output:
width=20.0 height =4.0

30

تعريف كائن إضافي وهو rect2

```
public class Constructor1 {
    public static void main (String [ ] args)
     {
      Myrectangle  rect  = new  Myrectangle(20,4);
      Myrectangle  rect2  = new  Myrectangle(40,8);
      System.out.println ("width="+ rect.width+"\t height="+ rect.height);
      System.out.println ("width2="+ rect2.width+"\t height2="+ rect2.height); }
}
class Myrectangle
 { double width,height;
  Myrectangle(double w , double h )
  {
    width=w;
    height=h ;
  }
 }
```

Output:

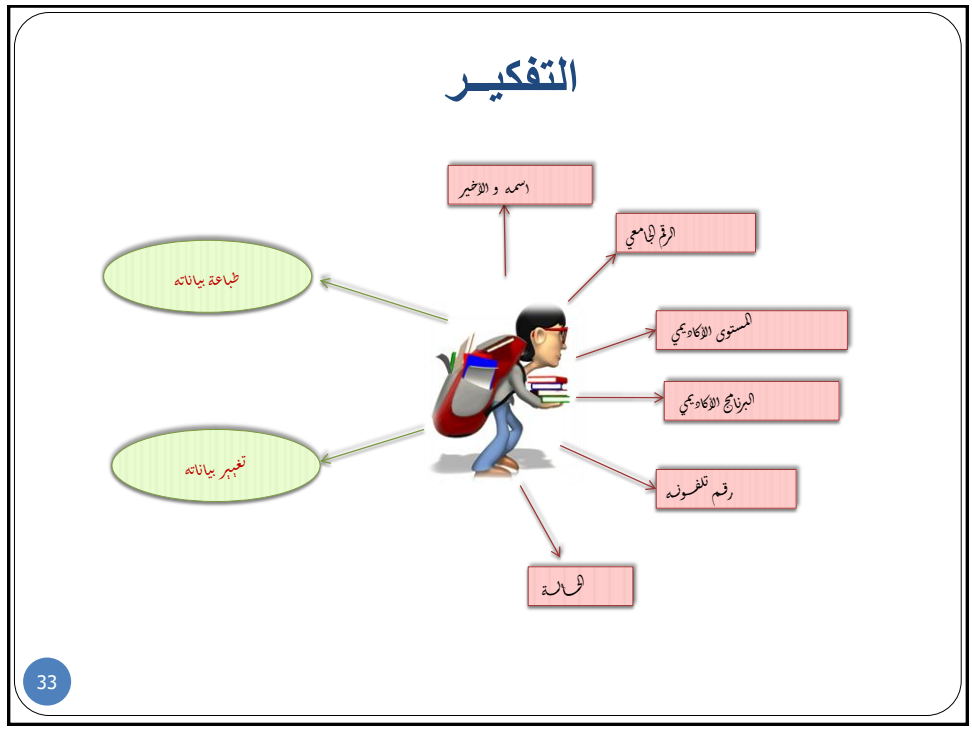
width=20.0 height =4.0
width2=40.0 height2 =8.0

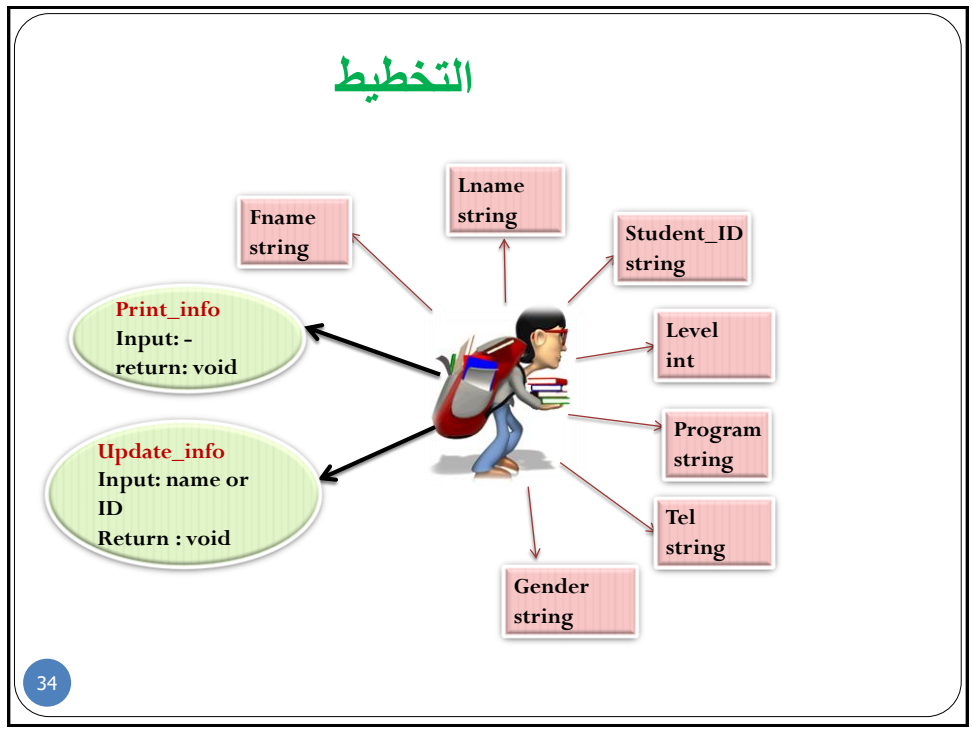
31

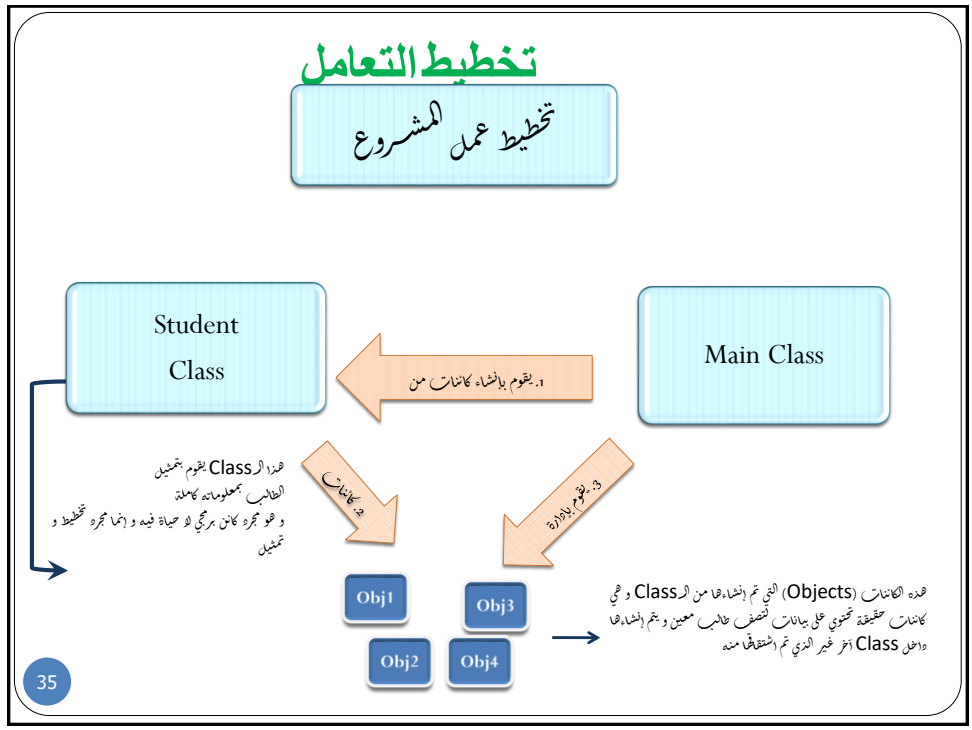
بناء و استخدام الـ Classes

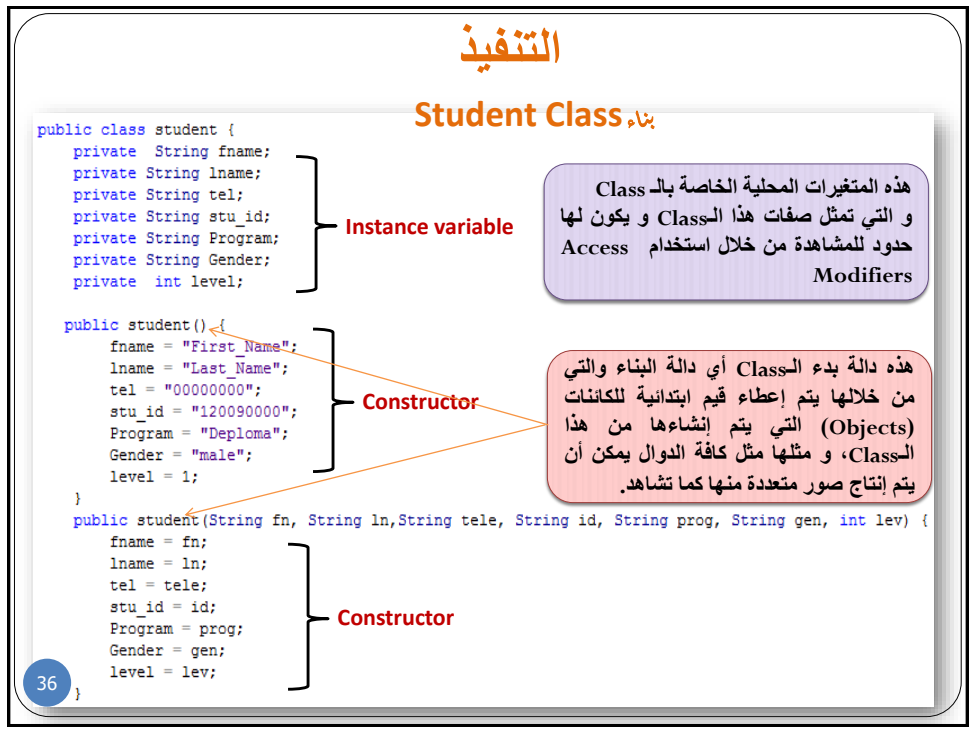
لنشيء نوع جديد باسم Student نتمكن من خلاله من تمثيل أي طالب بالكلية مع توفير الدوال التي تمكننا من التعامل مع بياناته.

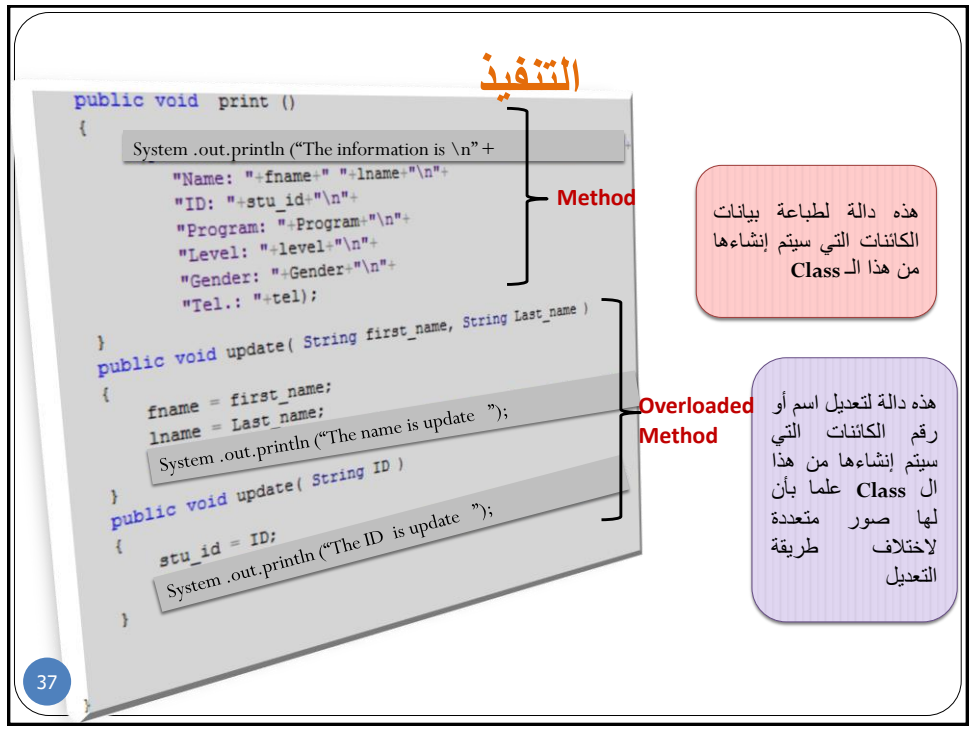
32

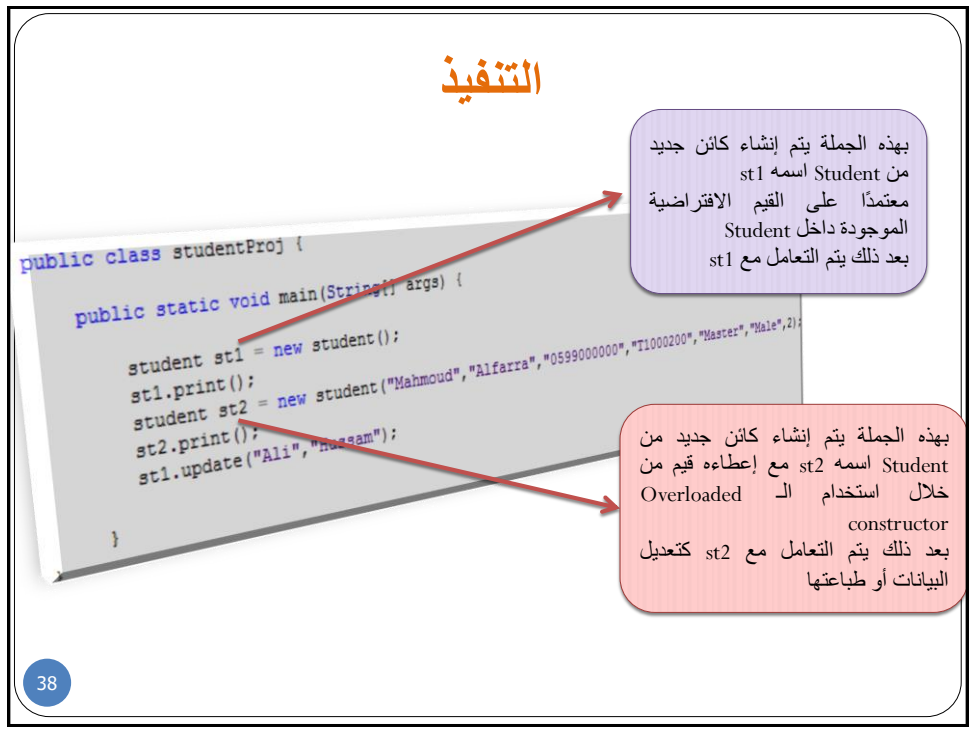

التفكيـر
اسمه و الاخير
الرقم الجامعي
المستوى الاكاديمي
البرنامج الاكاديمي
رقم تلفونه
الحالة
تغيير بياناته
طباعة بياناته
33


التخطيط
Lname
string
Fname
string
Student_ID
string
Print_info
Input: -
return: void
Level
int
Program
string
Update_info
Input: name or
ID
Return : void
Tel
string
Gender
string
34


تخطيط التعامل
تخطيط عمل المشروع
Student Class
Main Class
1. يقوم بانشاء كائنات من
هذا الـClass يقوم بتمثيل الطالب بمعلوماته كاملة و هو مجرد كائن برمجي لا حياة فيه و إنما مجرد تخطيط و تمثيل
2. كائنات
3. يقوم بإدارة
Obj1
Obj3
Obj2
Obj4
هذه الكائنات (Objects) التي تم إنشاءها من الـClass و هي كائنات حقيقية تحتوي على بيانات لشخص طالب معين و يتم إنشاءها داخل Class آخر غير الذي تم اشتقاقها منه
35

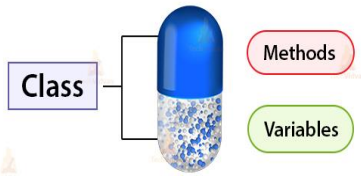

التنفيذ
بناء Student Class
public class student {
private String fname;
private String lname;
private String tel;
private String stu_id;
private String Program;
private String Gender;
private int level;
Instance variable
هذه المتغيرات المحلية الخاصة بالـ Class و التي تمثل صفات هذا الـClass و يكون لها حدود للمشاهدة من خلال استخدام Access Modifiers
public student() {
fname = "First_Name";
lname = "Last_Name";
tel = "00000000";
stu_id = "120090000";
Program = "Deploma";
Gender = "male";
level = 1;
}
Constructor
هذه دالة بدء الـClass أي دالة البناء والتي من خلالها يتم إعطاء قيم ابتدائية للكائنات (Objects) التي يتم إنشاءها من هذا الـClass، و مثلها مثل كافة الدوال يمكن أن يتم إنتاج صور متعددة منها كما تشاهد.
public student(String fn, String ln,String tele, String id, String prog, String gen, int lev) {
fname = fn;
lname = ln;
tel = tele;
stu_id = id;
Program = prog;
Gender = gen;
level = lev;
}
Constructor
36


التنفيذ
public void print ()
{
System .out.println ("The information is \n" +
"Name: "+fname+" "+lname+"\n"+
"ID: "+stu_id+"\n"+
"Program: "+Program+"\n"+
"Level: "+level+"\n"+
"Gender: "+Gender+"\n"+
"Tel.: "+tel);
}
public void update(String first_name, String Last_name)
{
fname = first_name;
lname = Last_name;
System .out.println ("The name is update ");
}
public void update(String ID)
{
stu_id = ID;
System .out.println ("The ID is update ");
}
Method
Overloaded Method
هذه دالة لطباعة بيانات الكائنات التي سيتم إنشاءها من هذا الـ Class
هذه دالة لتعديل اسم أو رقم الكائنات التي سيتم إنشاءها من هذا ال Class علما بأن لها صور متعددة لاختلاف طريقة التعديل
37


التنفيذ
public class studentProj {
public static void main(String[] args) {
student st1 = new student();
st1.print();
student st2 = new student("Mahmoud","Alfarra","0599000000","T1000200","Master","Male",2);
st2.print();
st1.update("Ali","Hassam");
}
بهذه الجملة يتم إنشاء كائن جديد من Student اسمه st1 معتمدًا على القيم الافتراضية الموجودة داخل Student بعد ذلك يتم التعامل مع st1
بهذه الجملة يتم إنشاء كائن جديد من Student اسمه st2 مع إعطاءه قيم من خلال استخدام الـ Overloaded constructor بعد ذلك يتم التعامل مع st2 كتعديل البيانات أو طباعتها
38

Encapsulation التغليف

- كل class يجب أن يحتوى على كل ما يحتاجه لعملِه.
- ضم الخصائص (attributes) الدوال (methods) المناسبة داخل class واحد، يُسمى التغليف (*encapsulation*).



- التغليف يضمن أن يكون الـclass مكتفي ذاتياً (*self-contained*).

39

Encapsulation التغليف

إخفاء البيانات (Data Hiding)

- هو جانب آخر من التغليف: فيه يجب أن لا يكون الـclass مكتفي ذاتياً (self-contained) **فقط ولكن** ينبغي أن يكون متمتع بتحكم ذاتي (*self-governing*) وبشكل جيد.
- الـClasses تستخدم ***private* access** على الحقول لتُخفيها عن باقي الـClasses.
- للسماح بالوصول لبيانات الـClass وتعديلها يحتاج إلى الدوال (methods).

40

Stale Data البيانات القديمة

- بعض البيانات تكون ناتجة من حسابات تجرى على عوامل مختلفة ، فمثلا مساحة (area) المستطيل ما هي إلا ناتج:

الطول(length)× العرض(width)

كلما تغيرت قيمة الطول أو العرض أو كليهما تغيرت قيمة المساحة ، ولأصبحت قيمة area قديمة ولا معنى لها لذلك أي خاصية قد تتغير بتغير عوامل أخرى وجب حسابها داخل دالة في الـclass لتجنب البيانات القديمة

```
public double getArea()
{
  return length * width;
}
```

41

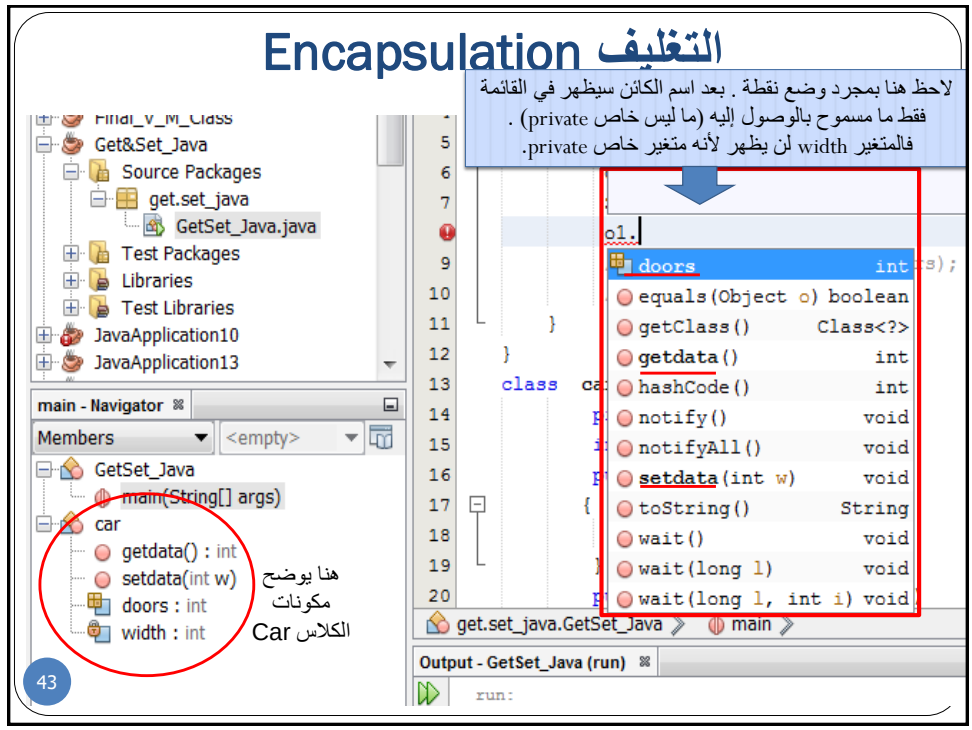
Encapsulation التغليف

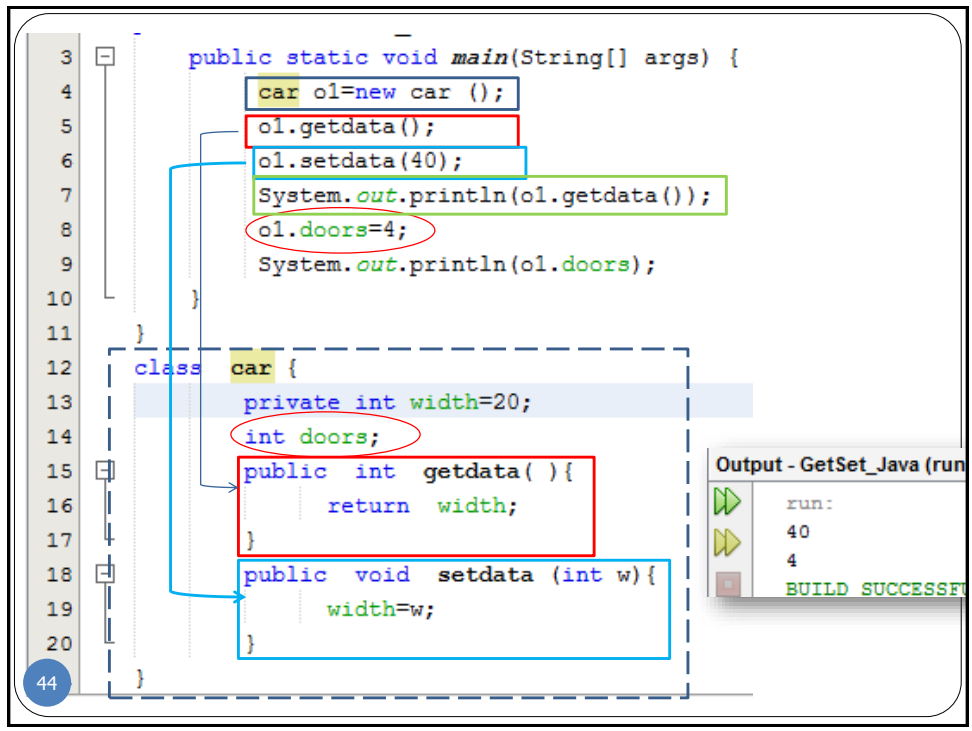
```
public class student{
    public static void main (String[ ] args){
        car o1=new car ();
        o1.getdata();
        o1.setdata(40);
        System.out.println(o1.getdata());
    }
}
class car {
    private int width=20;
    public int getdata( ) {
        return width;
    }
    public void setdata (int w) {
        width=w;
    }
}
```

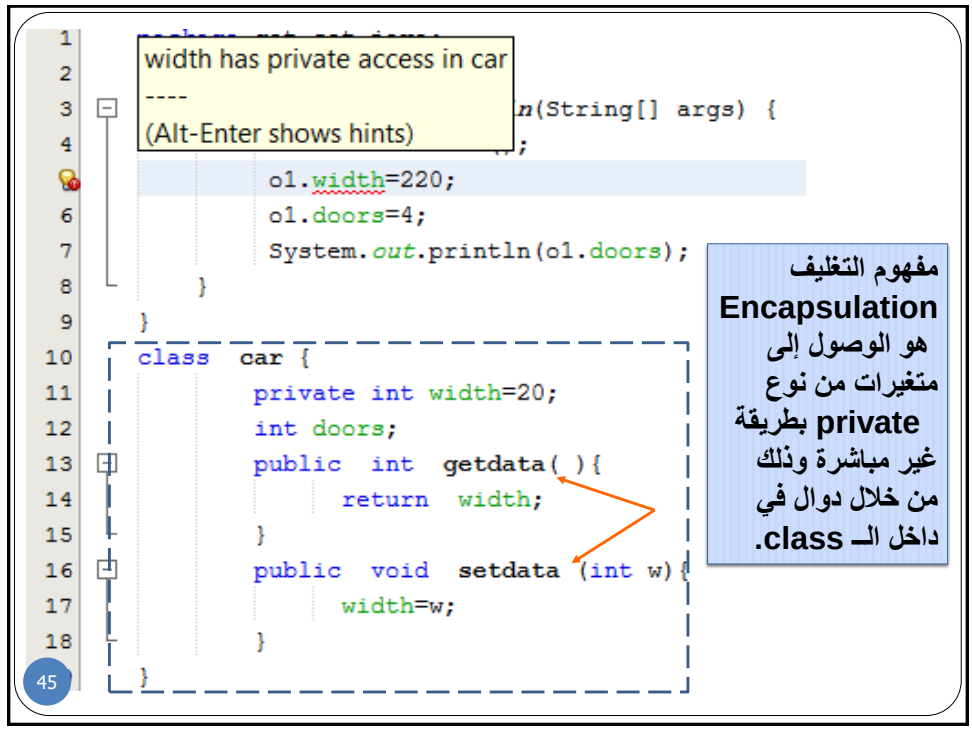
مفهوم Encapsulation
يمكننا من الوصول إلى متغيرات من نوع private بطريقة غير مباشرة وذلك من خلال دوال في داخل الـ class.

20
40

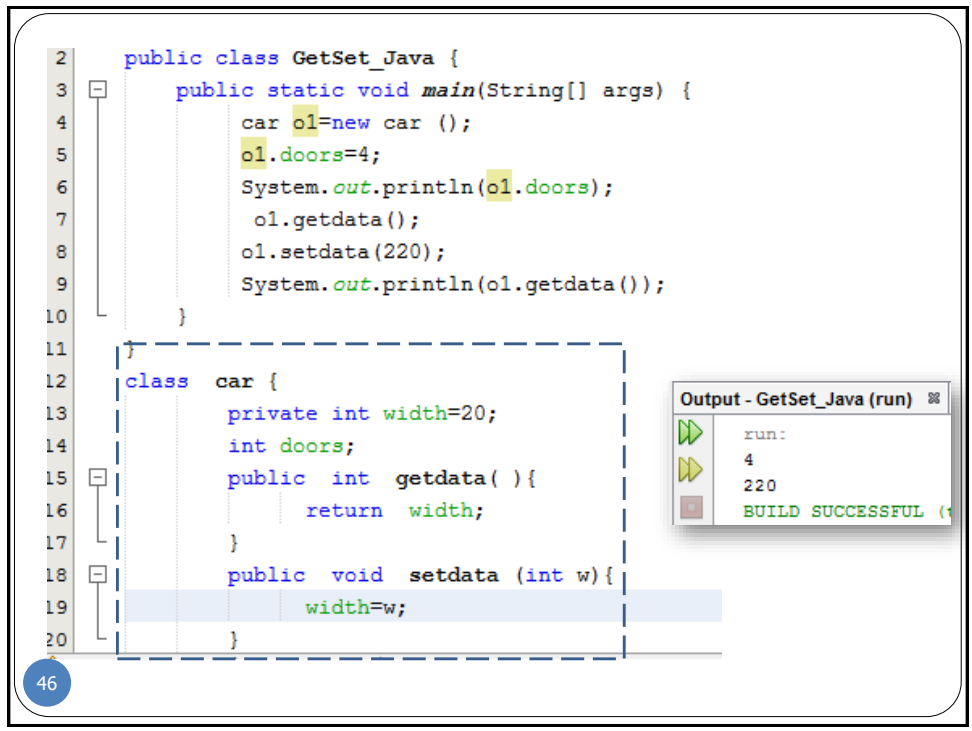
42


Encapsulation التغليف
لاحظ هنا بمجرد وضع نقطة . بعد اسم الكائن سيظهر في القائمة
فقط ما مسموح بالوصول إليه (ما ليس خاص private) .
فالمتغير width لن يظهر لأنه متغير خاص private.
Get&Set_Java
Source Packages
get.set_java
GetSet_Java.java
Test Packages
Libraries
Test Libraries
JavaApplication10
JavaApplication13
main - Navigator
Members
<empty>
GetSet_Java
main(String[] args)
car
getdata() : int
setdata(int w)
doors : int
width : int
هنا يوضح
مكونات
الكلاس Car
o1.
doors
equals(Object o) boolean
getClass() Class<?>
getdata() int
hashCode() int
notify() void
notifyAll() void
setdata(int w) void
toString() String
wait() void
wait(long l) void
wait(long l, int i) void
get.set_java.GetSet_Java
main
Output - GetSet_Java (run)
run:
43


public static void main(String[] args) {
car o1=new car ();
o1.getdata();
o1.setdata(40);
System.out.println(o1.getdata());
o1.doors=4;
System.out.println(o1.doors);
}
}
class car {
private int width=20;
int doors;
public int getdata( ){
return width;
}
public void setdata (int w){
width=w;
}
}
Output - GetSet_Java (run
run:
40
4
BUILD SUCCESSF
44


width has private access in car

(Alt-Enter shows hints)
n(String[] args) {
o1.width=220;
o1.doors=4;
System.out.println(o1.doors);
}
}
class car {
private int width=20;
int doors;
public int getdata( ){
return width;
}
public void setdata (int w){
width=w;
}
}
مفهوم التغليف
Encapsulation
هو الوصول إلى
متغيرات من نوع
private بطريقة
غير مباشرة وذلك
من خلال دوال في
داخل الـ class.
45


public class GetSet_Java {
public static void main(String[] args) {
car o1=new car ();
o1.doors=4;
System.out.println(o1.doors);
o1.getdata();
o1.setdata(220);
System.out.println(o1.getdata());
}
}
class car {
private int width=20;
int doors;
public int getdata( ){
return width;
}
public void setdata (int w){
width=w;
}
Output - GetSet_Java (run)
run:
4
220
BUILD SUCCESSFUL
46

الـClasses والمتغيرات (Variables) مجال الرؤية Scope

فيما يلي سرد لأنواع مجال الرؤية (Scope) للمتغير بناءًا على مكان الاعلان عنه (declaration):

- داخل الدالة (Inside a method):
  - مرئي فقط ضمن الدالة method.
  - يُسمى متغير محلي (local variable).
- في مُعامل الدالة (In a method parameter):
  - يُسمى parameter variable
  - مثل المتغير المحلي (local variable) ومرئي فقط ضمن الدالة (method)
- داخل الـclass ولكن ليس داخل الدالة (Inside the class but not in a method)
  - مرئي لكل عناصر الـClass.
  - يُسمى instance field.

47

Static Variable

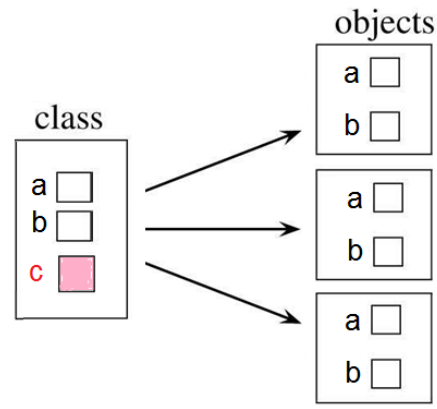
- ✓ عند تعريف متغير ما في (Class) بـ static فهذا يعني أن جميع الكائنات (objects) التي سيتم اشتقاقها من هذا الصنف سيكون لهم قيمة مشتركة لهذا المتغير.
- ✓ و هذا على عكس ما يحدث في الوضع الطبيعي حيث يحصل كل كائن على نسخة افتراضية دون أن تتأثر بالكائنات الآخرى.

```
private static int count = 0;
```

48

Static Variable

```
class myclass{
    int a;
    int b;
    static int c;
 void printabc( )
     {
          System.out.println("a="+a);
          System.out.println("b="+b);
          System.out.println("c="+c);
}
```



49

Static Variable

```
class class1{
    int a,b;
    static int c;
    void printabc()
    {
        System.out.println("a="+a);
        System.out.println("b="+b);
        System.out.println("c="+c);
```

عند تعريف متغير انه Static فهذا يعني انه مشترك بين كل الكائنات objects التي تنشأ من هذا الـClass، وأي تغير يطرأ عليه يظهر للجميع.

عند استخدامك هذا المتغير مع أي كائن object، سيظهر تلميح يوضح أنك تتعامل مع متغير Static

```
2   Accessing static field c
3   ----                          {
    (Alt-Enter shows hints)       c void main(String[] args) {
4                                 =new class1();
5           x.c=30;
6           x.a=20;
7           x.b=40;
```

لذلك من الأصح استخدام هذا المتغير مع اسم الـclass المعرف به بدلا من اسم الكائن object، للدلالة على انه مشترك.

```
public static void main(String[] args) {
    class1 x =new class1();
    class1.c=30;
    x.a=20;
```

50

Static Variable

```
class class1{
    int a,b;
    static int c;
    void printabc()
    {
        System.out.println("a="+a);
        System.out.println("b="+b);
        System.out.println("c="+c);
    }
```

عند تعريف متغير انه Static فهذا يعني انه مشترك بين كل الكائنات objects التي تنشأ من هذا الـClass، وأي تغير يطرأ عليه يظهر للجميع.

لاحظ النتائج تجد أن قيمة c في الحالتين ظهرت 500 رغم انها مع الكائن x كانت 30 ومع الكائن m كانت 500.

```
3   public static void main(String[] args) {
4       class1 x =new class1();
5       class1 M =new class1();
6       x.c=30;
7       x.a=20;
8       x.b=40;
9       M.a=200;
10      M.b=400;
11      M.c=500;
12      x.printabc();
13      M.printabc();
14  }
15  }
```

```
Output - OOp (run)
run:
a=20
b=40
c=500
a=200
b=400
c=500
BUILD SUCCESSFUL
```

51

Static Variable

```
public class StaticVariable {
  static int noOfInstances;
  StaticVariable() {
    noOfInstances++;
  }}
```

```
public static void main(String[] args){
   StaticVariable sv1 = new StaticVariable();
   System.out.println("No. of instances for sv1 : " + sv1.noOfInstances);      // 1
   StaticVariable sv2 = new StaticVariable();
   System.out.println("No. of instances for sv1 : "  + sv1.noOfInstances);     //2
   System.out.println("No. of instances for st2 : "  + sv2.noOfInstances);     //2
   StaticVariable sv3 = new StaticVariable();
   System.out.println("No. of instances for sv1 : "  + sv1.noOfInstances);     //3
   System.out.println("No. of instances for sv2 : "  + sv2.noOfInstances);     //3
   System.out.println("No. of instances for sv3 : "  + sv3.noOfInstances);     //3
      }
```

52

```
4       public static void main(String[] args) {
5       StaticVariable sv1 = new StaticVariable();
6       System.out.println("No. of instances for sv1 : "+ sv1.noOfInstances);
7       StaticVariable sv2 = new StaticVariable();
8       System.out.println("No. of instances for sv1 : "+ sv1.noOfInstances);
9       System.out.println("No. of instances for sv2 : "+ sv2.noOfInstances);
10      StaticVariable sv3 = new StaticVariable();
11      System.out.println("No. of instances for sv3 : "+ sv3.noOfInstances);
12      System.out.println("No. of instances for sv1 : "+ sv1.noOfInstances);
13      System.out.println("No. of instances for sv2 : "+ sv2.noOfInstances);
14      }
15  }
16  class StaticVariable{
17      int noOfInstances;
18      StaticVariable()
19      {
20          noOfInstances++;
21      }
22  }
```

```
Output - StaticV (run)
run:
No. of instances for sv1 : 1
No. of instances for sv1 : 1
No. of instances for sv2 : 1
No. of instances for sv3 : 1
No. of instances for sv1 : 1
```

53

```
5       public static void main(String[] args) {
6       StaticVariable sv1 = new StaticVariable();
        System.out.println("No. of instances for sv1 : "+ sv1.noOfInstances);
8       StaticVariable sv2 = new StaticVariable();
        System.out.println("No. of instances for sv1 : "+ sv1.noOfInstances);
        System.out.println("No. of instances for sv2 : "+ sv2.noOfInstances);
11      StaticVariable sv3 = new StaticVariable();
        System.out.println("No. of instances for sv3 : "+ sv3.noOfInstances);
        System.out.println("No. of instances for sv1 : "+ sv1.noOfInstances);
        System.out.println("No. of instances for sv2 : "+ sv2.noOfInstances);
15      }
16  }
17  class StaticVariable{
18      static int noOfInstances;
19      StaticVariable()
20      {
21          noOfInstances++;
22      }
23  }
```

نفس المثال مع اضافة static قبل المتغير

```
Output - StaticV (run)
run:
No. of instances for sv1 : 1
No. of instances for sv1 : 2
No. of instances for sv2 : 2
No. of instances for sv3 : 3
No. of instances for sv1 : 3
No. of instances for sv2 : 3
BUILD SUCCESSFUL (total time: 0 seconds)
```

54