جَامعة المَنارة
MANARA UNIVERSITY

كلية الهندسة المعلوماتية

برمجة 3
Java Programming

ا. د. علي عمران سليمان

محاضرات الأسبوع الأول

الفصل الصيفي 2023-2024

Contents 1

1-Java Primer.

1.1 Comments, Base Types

1.2 Classes and Objects (Defining, Access Control Modifiers, static, abstract, final Modifier), Declaring , (Instance Variables, Methods, Parameters, Constructors, main Method)

1.3 Strings, Wrappers, Arrays, and Enum Types (Concatenation, StringBuilder Class, Wrapper Types, Automatic Boxing Unboxing).

1.4 Arrays, Enum.

1.5 Expressions.

1.6 Simple Input and Output (Simple Output Methods, Scanner Class).

1.7 Packages and Imports.

1.8 Software Development.

References

- Deitel & Deitel, Java How to Program, Pearson; 10th Ed(2015)
- د.علي سليمان، بنى معطيات بلغة JAVA، جامعة تشرين 2013-2014

Contents 2

2- Object-Oriented Design

2.1 Goals, Principles, and Patterns
2.2 Inheritance
2.3 Interfaces and Abstract Classes
2.4 Exceptions
2.5 Casting and Generics
2.6 Nested Classes
2.7 Exercises

3- GUI

3.1 Introduction
3.2 Overview of Swing Components
3.3 JLabel
3.4 Event Handling
3.5 TextFields
3.6 How Event Handling Works

References

- Deitel & Deitel, Java How to Program, Pearson; 10th Ed(2015)
- د.علي سليمان، بنى معطيات بلغة JAVA، جامعة تشرين 2013-2014

Contents 3

References

- Deitel & Deitel, Java How to Program, Pearson; 10th Ed(2015)
- د.علي سليمان، بنى معطيات بلغة JAVA، جامعة تشرين 2013-2014

What IS Object Oriented Programming ?

- البرمجة غرضية التوجه (Object Oriented Programming OOP) هي نموذج برمجة programming paradigm يعتمد على مفهوم "الكائنات".
- نموذج البرمجة: هو نمط style من البرمجة او طريقة way للتفكير في بناء البرمجيات.
- لا يرتبط نموذج البرمجة بلغة معينة بل هو طريقة لبناء برنامج أو منهجية methodology للتطبيق.
- تسهل بعض اللغات الكتابة في بعض النماذج دون غيرها.
- تسمح بعض لغات البرمجة للمبرمج بتطبيق أكثر من نموذج واحد C++.

Programming Paradigms

- The programming paradigm refers to a way of conceptualizing and structuring the tasks a computer performs.

Paradigm	Languages	Description
Procedural	BASIC, Pascal, COBOL, FORTRAN, Ada	Emphasizes linear steps that provide the computer with instructions on how to solve a problem or carry out a task
Object-oriented	Smalltalk, C++, Java	Formulates programs as a series of objects and methods that interact to perform a specific task
Declarative	Prolog	Focuses on the use of facts and rules to describe a problem
Functional	LISP, Scheme, Haskell	Emphasizes the evaluation of expressions, called functions
Event-driven	Visual Basic, C#	Focuses on selecting user interface elements and defining event-handling routines that are triggered by various mouse or keyboard activities

Some notes about Java 1

- يتضمن أي ملف مكتوب بلغة Java صنفاً class عاماً فقط(أو واجهه)، وقد يتضمن البرنامج عددًا كبيرًا من الأصناف، والملف الواحد يملك صنف عام يخزن الملف باسمه على القرص.
- قد يحتوي أي برنامج على عدد كبير من المناهج methods، واحدة منها هي المنهج الرئيس main method، وإذا اشتملت على منهج واحدة فقط، فسيكون هو المنهج الرئيس ، والتي يبدأ منه التنفيذ ويعرف بنقطة الدخول إلى البرنامج.
- يتضمن الصنف (والذي يشبه المخطط blueprint) وصفًا للبيانات الاعضاء data member، حقول البيانات, data fields،خصائص البيانات properties of data والمعروف باسم السمات attribute،والمناهج الاعضاء member method، أو العمليات operations أو الأحداث Actions، والمعروفة بالسلوكيات behaviors (وتؤمن التواصل مع بقية الأصناف).
- يتم اشتقاق الكائنات Objects من الاصناف وتُعرف بالأمثال instances.
- يفضل تضمين التعليقات comments التي تسهل تذكرومعرفة عمل البرنامج.

- عند التصريح عن المناهج والبيانات ، تبدأ عادتاً بمحددات الوصول access specifiers.
- يكتب كود البرنامج داخل جسم الصنف.
- الأصناف هي مصدر الكائنات.
- تتم كتابة برامج Java بلغة عالية المستوى بامتداد java. ويتم مطابقتها لتحويلها إلى لغة وسيطة تُعرف باسم bytecodes وتخزينها بامتداد class.ومن ثم تشغيلها للحصول على النتائج.

- في معظم لغات البرمجة، قد تظهر الأخطاء أثناء:
  - مطابقة البرامج وتخطي قواعد اللغة، والمعروف باسم syntax error.
  - عند تشغيل البرنامج، وإدخال قيم خطأ مثلاً، والمعروف باسم runtime error.
  - إعطاء نتائج غير متوقعة، خطأ معالجة يُعرف بالخطأ المنطقي logical error.

An overview of the Java programming language

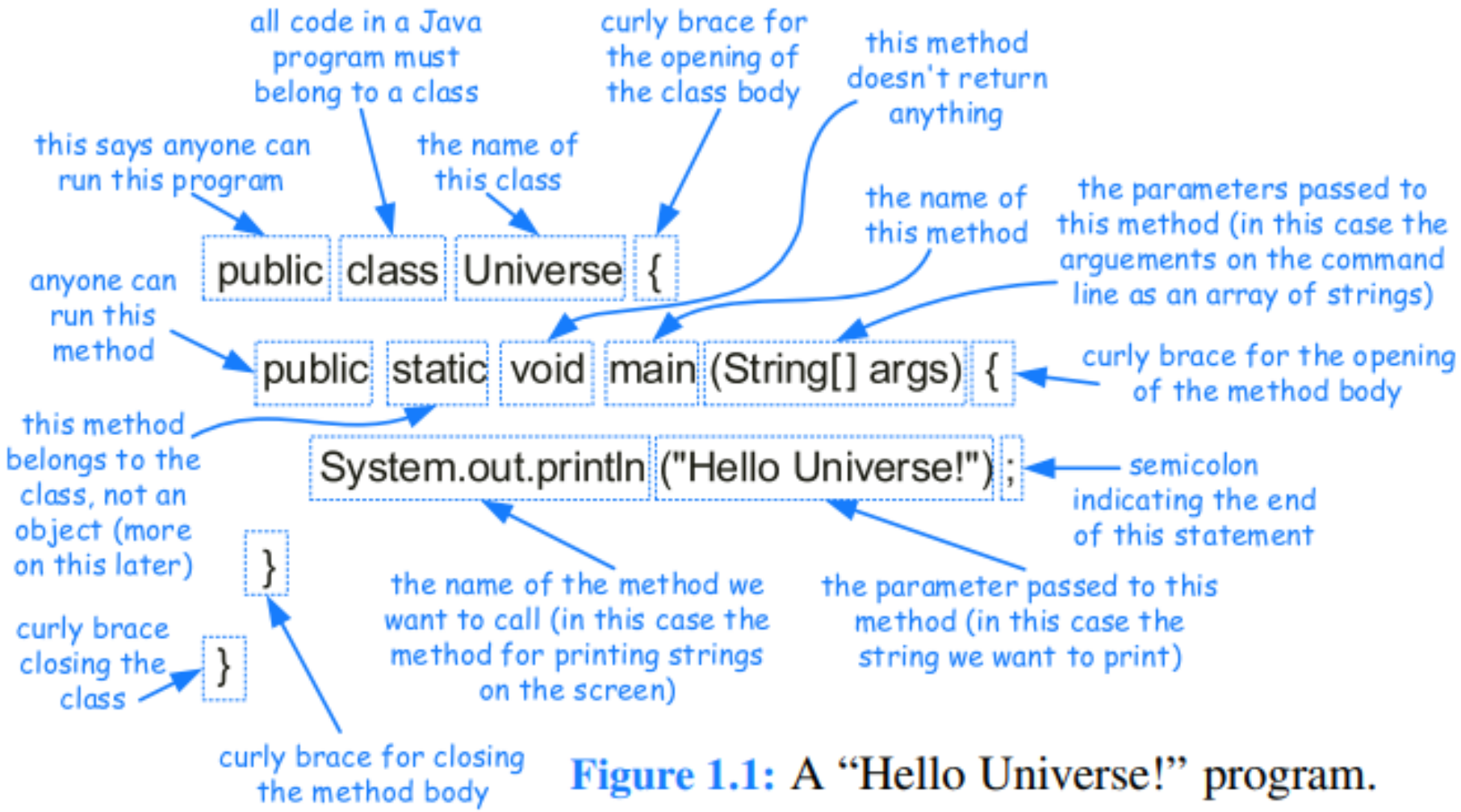


Figure 1.1: A "Hello Universe!" program.

- لإنشاء كائن من صنف يتم استدعاء المنهج الباني من خلال المعامل new في الجافا يكون الحجز ديناميكياً فقط:

```
Scanner input = new Scanner (System.in);
```

- نجد ثلاثة أحداث:

1. سيتم البحث ضمن الذاكرة عن أماكن متماسة كافية للغرض وحجزها (يتم تخصيص مثيل جديد ديناميكيًا في الذاكرة).
2. يتم تهيئة جميع متغيرات الحالة بالقيم الافتراضية القياسية. null للمتغيرات المرجعية reference و0 لكل الانواع الأساسية، باستثناء المتغيرات المنطقية Boolean (التي تكون افتراضيًا false). وقد يقوم الباني بتعيين قيم أكثر أهمية (غير الافتراضية) لأي من متغيرات الحالة، وإجراء أي عمليات حسابية إضافية يجب إجراؤها عند إنشاء هذا الكائن.
3. بعد تنفيذ الباني يُرجع **returns** المُعامل new مرجعًا reference (أي عنوان بداية الحجز للكائن في الذاكرة) إلى الكائن الذي تم إنشاؤه حديثًا في مثالنا input.

- وإذا لم يجد حجماً كافياً فسيتم إعادة **null**تخزين هذا العنوان في متغير الكائن.

access control modifiers

- التحكم بمُعدِّلات الوصول المختلفة (من يستطيع الوصول إليها):
- public **class** محدد وصول عام للصنف: أي أن جميع الأصناف يمكنها الوصول إلى الصنف العام، يجب تحديد كل صنف عامة في ملف منفصل باسمه، مثلاً **classname.java**.
- protected **class** محدد وصول محمي للصنف: أي أن الوصول إليه يُمنح فقط للمجموعات التالية من الأصناف الأخرى:
  - الأصناف االوارثة لهذا الصنف المحمي.
  - الأصناف التي تنتمي إلى نفس الحزمة التي ينتمي لها الصنف المحمي.
- private **class** الصنف الخاص: أن الوصول إلى أعضائه يتم فقط للتعليمات البرمجية الموجودة داخل تلك الصنف. لا يمكن للأصناف الفرعية (الموارثة) ولا أي أصناف أخرى الوصول إلى أعضاء هذا الصنف.
- في حالة عدم وجود أي معدِّل صريح للتحكم بالوصول، فستكون الحالة الافتراضية default (friendly)له وتُعرف بمستوى الوصول الخاص بالحزمة package-private. يسمح للأصناف في نفس الحزمة بالوصول إلى أعضائه.

The static Modifier

- static modifier في Java يمكن أن يحدد لأي كائن Objectأو منهج Method ضمن صنف .
- عندما يتم الإعلان عن كائن أومتغير من صنف على أنه static، فإن قيمته ترتبط بالصنف ككل، بدلاً من ارتباطها مع كل مثيل بمفرده مشتق من الصنف، تُستخدم المتغيرات staticلتخزين معلومات global عن الصنف.
- عندما يتم الإعلان عن منهج من صنف ما على أنه static، فإنه يرتبط أيضًا بالصنف نفسه، وليس بكائن معين من الصنف. عادةً ما يتم استدعاؤه باستخدام اسم الصنف نقطه اسم المنهج.
- abstract يمكن الإعلان عن منهج من صنف ما على أنه مجرد، وفي هذه الحالة يتم توفير توقيعها أو بصمتها signature ولكن بدون تنفيذ implementation جسم المنهج.
- final يمكن تهيئة المتغير الذي تم التصريح عنه بالمعدِّل final كجزء من الإعلان عنه، ولكن لا يمكن أبدًا تعيين قيمة جديدة له تعتبر قيمتة ثابته وهو ضمنا static، وإذا كان متغير مرجعي reference، فسيشير دائمًا إلى نفس مكان الكائن ولكن يمكن تغيير محتوى الكائن.
- Final للمناهج لا يمكن أن يعمل لها overridden سيتم معالجتها في الوراثة .

Data Hiding

- يخفي الكائن حقوله الداخلية الخاصة عن التعليمات البرمجية الموجودة خارج الصنف وبالتالي لايمكن استخدامها.

- فقط مناهج الصنف يمكنها الوصول مباشرة إلى البيانات الداخلية للكائن وتغييرها، ولكي نصل إليها من خارج الصنف يجب أن تكون عامة، وبالتالي يمكن وضع الاحتياطات المناسبة عند برمجتها لمنع المستثمر من عمل غير منطقي أي حماية المعطيات من الاستثمار الخاطئ.

- Data hiding إخفاء البيانات: يعد أمرًا مهمًا لأن الأصناف تُستخدم عادةً كمكونات في أنظمة البرامج الكبيرة، والتي تضم فريقًا من المبرمجين ويستخدمها العديد من المستثمرين.

- يساعد إخفاء البيانات في تعزيز integrity صحة وتكامل البيانات الداخلية للكائن.

declaring instance variables

- **general syntax الشكل العام للتصريح عن متغير مثل أو أكثر من صنف يكون بالشكل التالي:**

[*modifiers*] *type identifier1*[=*initialValue1*], *identifier2*[=*initialValue2*];

مثال:

private int count;

- **حيث** private **هي نمط الوصول** modifier.
- int **هي نوع المتعير.**
- count **اسم المتعير.**

- **نظراً لعدم التعريف والتجهيز بقيمة، بشكل تلقائي سيتم اسناد القيم الافتراضية صفر لإنه عدد كما مر سابقاً.**

declaring method

- general syntax الشكل العام للتصريح عن منهج من صنف يكون بالشكل التالي:

[*modifiers*] *returnType methodName(type1 param1 ,..., typen paramn*)
{ // method body . . . }

مثال:

public void increment(int delta) { count += delta; }

- modifiers **such as** public, private, **and** protected.
- returnType **defines the type of** value returned **by the method**.
- methodName **can be any valid Java identifier**.
- The list of parameters (or **Parameter variable declaration**) **and their** types declares **the local variables that are to be passed as arguments to this method.**

declaring constructor

- constructor الباني هو نوع خاص special type من مناهج الصنف وموجود افتراضيًا , تقوم java ومعظم اللغات ببنائة.
- يتم استدعاء الباني المكتوب تلقائيًا called automatically عند إنشاء كائن وإذا لم يكتب ينادى الافتراضي.
- يتم استخدامه لتهيئة حقول الكائن الذي يتم إنشاؤه من الصنف بحيث يكون في حالة أولية منسجمة ومستقرة بالقيم الإقتراضية الأولية. أو بأية قيم مرغوبه تمرر له.
- يمكنه إجراء عمليات حسابية أكثر تعقيدًا من إسناد القيم في وقت إنشاء الكائن من خلال تضمينها في الباني.
- الصيغة العامة للتصريح عن مُنشئ في Javaهي كما يلي:

modifiers name(*type0 parameter0* , ..., *typen*−1 *parametern*−1)

{ // constructor body . . . }

- modifiers **typically public And can be** protected, private**, or the default** package-level **visibility** But **cannot be** static, abstract, **or** final.

- **We don't specify a** return type **for a constructor** (**not even void**).

-**The name must be the same name as the class**.

Counter d = new Counter(5);

The Keyword this

- هناك ثلاثة أسباب شائعة لضرورة استخدام this (المرجع) من داخل جسم الطريقة:

1. لتخزين المرجع في متغير، أو إرساله كمعامل إلى منهج أخرى يتوقع مثيلاً (مرجعاً) من هذا النوع كوسيطة. (نقول أن المتغير المحلي يخفي متغير الحالة).

2. للتمييز بين متغير حالة ومتغير محلي بنفس الاسم.

3. للسماح لجسم باني واحد باستدعاء جسم باني آخر.

Objects

- **Object-oriented programming** (**OOP**) is a programming paradigm based on the concept of "objects"

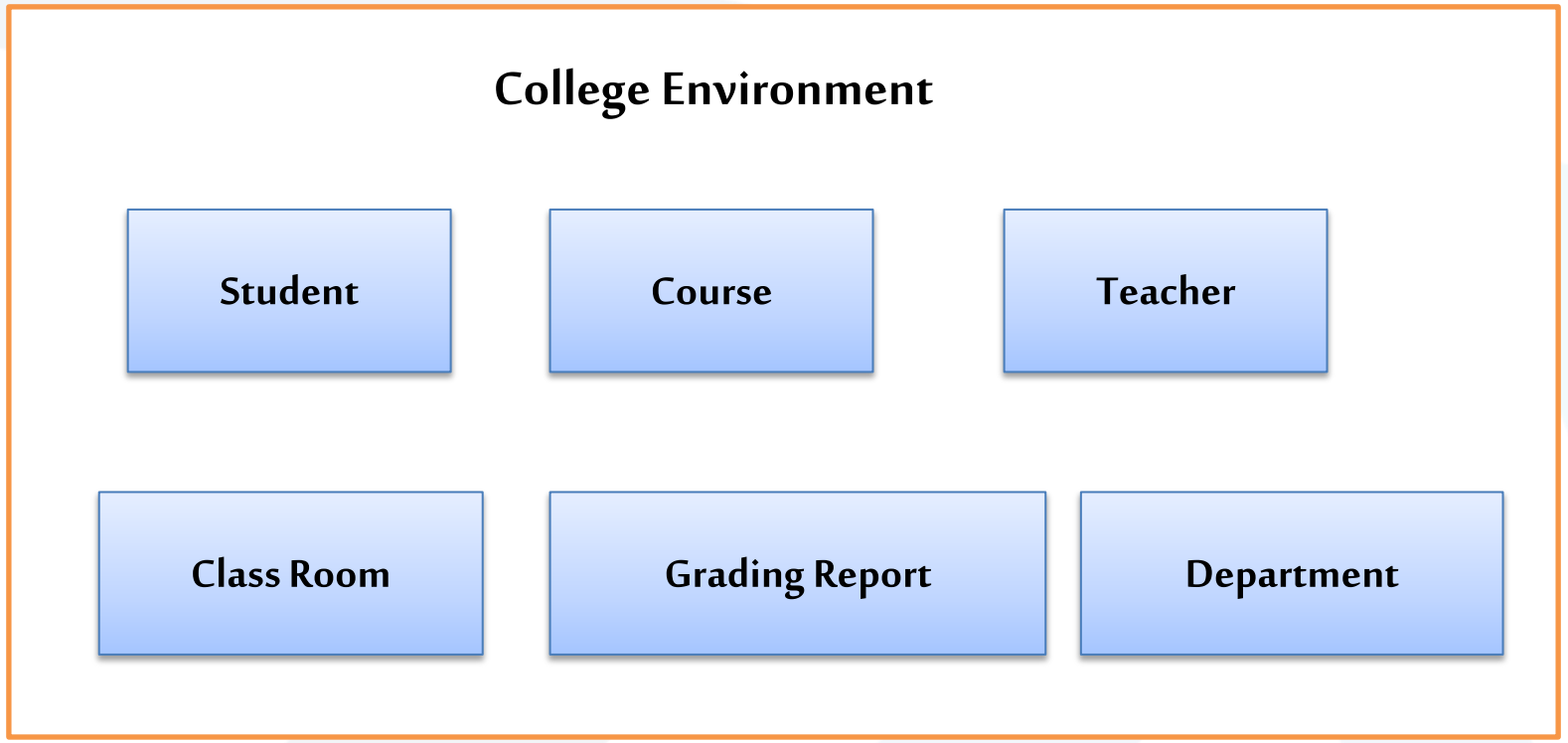
Object : is a thing (Tangible – Intangible)



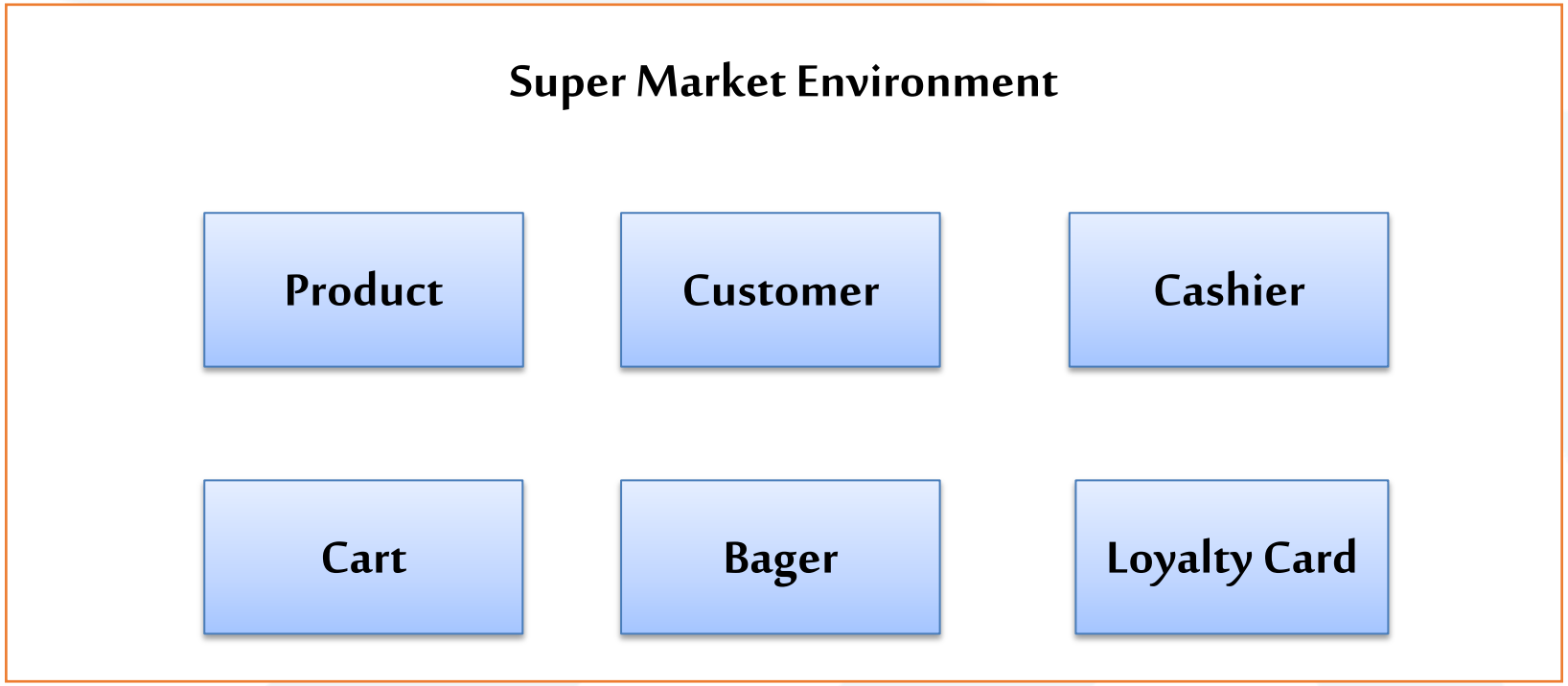




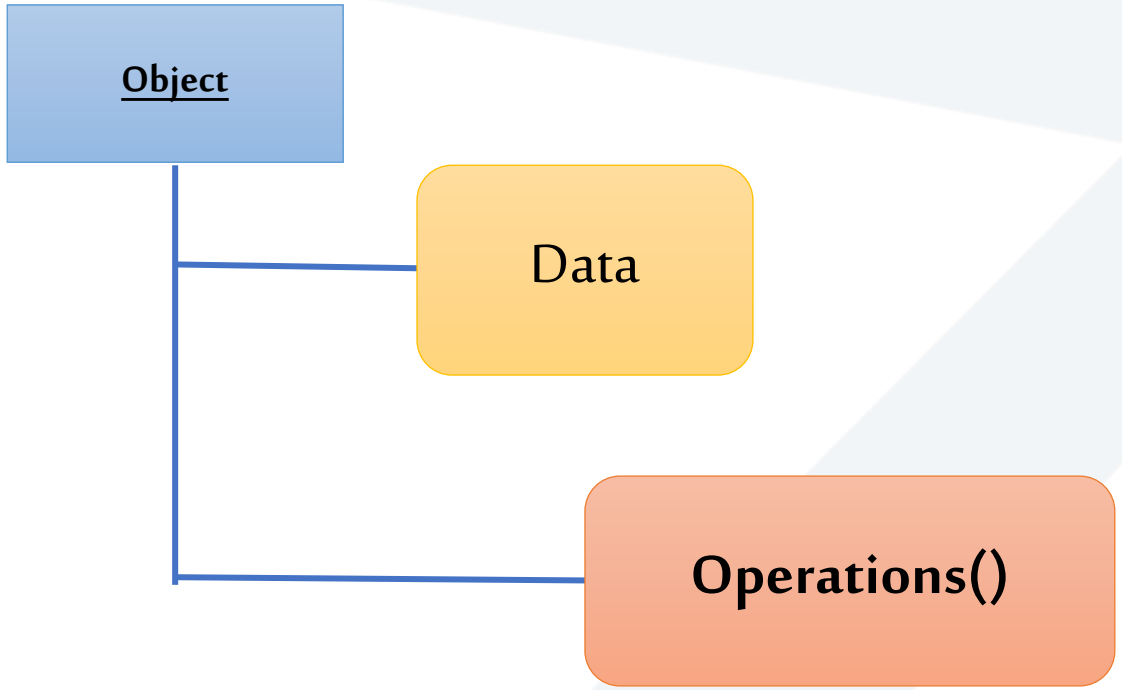
Objects in College Management Program



Objects in Super market Program



Object Is comprised Of ?



Product Object

Data

1- Product _name
2- Product _code
3- Price
4- Producer
5-Discount

Operations ( )

1- ModifyPrice ()
2- SetDiscount ()
3- GetProductName ()
4- GetProductPrice ()

Object Is comprised Of ?

StudentOb

ject

Data

1- Student_name

2- University_Id

3- Birth_Date

4- Address5-GPA

6- Study_Level

Operations ( )

1- Modify GPA()

2- Change Study level ()

3- Get Student Name ()

4- Get Student Address ()

Car Object

Data

1- Factory

2- Model

3- Fuel_Capacity

4- No_of_doors

5-Color

6- Shape

Operations ( )

1- Set Factory Name()

2- Change Color ()

3- Get Car Info ()

4- .....

What is Class ? Why we need It ?

Student 1	Student 2	Student 3
Data:	**Data:**	**Data:**
1- Student_name	1- Student_name	1- Student_name
2- University_Id	2- University_Id	2- University_Id
3- Birth_Date	3- Birth_Date	5-GPA
4- Address	4- Address	6- Study_Level
5-GPA	6- Study_Level	
6- Study_Level		
Operations ( )	**Operations ( )**	**Operations ( )**
1- Get Student Name ()	1- Get Student Name ()	1- Get Student Name ()
2- Change Study level ()	2- Change Study level ()	2- Change Study level ()
3-Modify GPA()	4- Get Student Address ()	3- Modify GPA()
4- Get Student Address ()		5- Get University_Id ()

What is Class ? Why we need It ?

Class Student	Student 1	Student 2	Student 3
Data: 1- Student_name 2- University_Id 3- Birth_Date 4- Address 5-GPA 6- Study_Level	**Data:** 1- Student_name 2- University_Id 3- Birth_Date 4- Address 5-GPA 6- Study_Level	**Data:** 1- Student_name 2- University_Id 3- Birth_Date 4- Address 6- Study_Level	**Data:** 1- Student_name 2- University_Id 5-GPA 6- Study_Level
Operations ( ) 1- Get Student Name () 2- Change Study level () 3-Modify GPA() 4- Get Student Address ()	**Operations ( )** 1- Get Student Name () 2- Change Study level () 3-Modify GPA() 4- Get Student Address ()	**Operations ( )** 1- Get Student Name () 2- Change Study level () 4- Get Student Address ()	**Operations ( )** 1- Get Student Name () 2- Change Study level () 3- Modify GPA() 5- Get University_Id ()

What is Class ? Why we need It ?

Class Student	Student 1	Student 2	Student 3
Data:	**Data:**	**Data:**	**Data:**
1- Student_name	1- Student_name	1- Student_name	1- Student_name
2- University_Id	2- University_Id	2- University_Id	2- University_Id
3- Birth_Date	3- Birth_Date	3- Birth_Date	5-GPA
4- Address	4- Address	4- Address	6- Study_Level
5-GPA	5-GPA	6- Study_Level	
6- Study_Level	6- Study_Level		
7- Email	7- Email	7- Email	7- Email
Operations ( )	**Operations ( )**	**Operations ( )**	**Operations ( )**
1- Get Student Name ()	1- Get Student Name ()	1- Get Student Name ()	1- Get Student Name ()
2- Change Study level ()	2- Change Study level ()	2- Change Study level ()	2- Change Study level ()
3-Modify GPA()	3-Modify GPA()	4- Get Student Address ()	3- Modify GPA()
4- Get Student Address ()	4- Get Student Address ()		5- Get University_Id ()

What is Class ? Why we need It ?

Class Student	Student 1	Student 2	Student 3
Data:	**Data:**	**Data:**	**Data:**
1- Student_name	1- Student_name	1- Student_name	1- Student_name
2- University_Id	2- University_Id	2- University_Id	2- University_Id
3- Birth_Date	3- Birth_Date	3- Birth_Date	5-GPA
4- Address	4- Address	4- Address	6- Study_Level
5-GPA	5-GPA	6- Study_Level	
6- Study_Level	6- Study_Level		
7- Email	7- Email	7- Email	7- Email
Operations ( )	**Operations ( )**	**Operations ( )**	**Operations ( )**
1- Get Student Name ()	1- Get Student Name ()	1- Get Student Name ()	1- Get Student Name ()
2- Change Study level ()	2- Change Study level ()	2- Change Study level ()	2- Change Study level ()
3-Modify GPA()	3-Modify GPA()	4- Get Student Address ()	3- Modify GPA()
4- Get Student Address ()	4- Get Student Address ()		5- Get University_Id ()
5- Print Student Info ()	5- Print Student Info ()	5- Print Student Info ()	5- Print Student Info ()

What is Class ? Why we need It ?

Class Student

Data:

1- Student_name
2- University_Id
3- Birth_Date
4- Address
5-GPA
6- Study_Level
7- Email

Operations ( )

1- Get Student Name ()
2- Change Study level ()
3-Modify GPA()
4- Get Student Address ()
5- Print Student Info ()

What is Class ? Why we need It ?

Class Student

Data:

1- Student_name
2- University_Id
3- Birth_Date
4- Address
5-GPA
6- Study_Level
7- Email

Operations ( )

1- Get Student Name ()
2- Change Study level ()
3-Modify GPA()
4- Get Student Address ()
5- Print Student Info ()

Student 1

Data:

1- Student_name
2- University_Id
3- Birth_Date
4- Address
5-GPA
6- Study_Level
7- Email

Operations ( )

1- Get Student Name ()
2- Change Study level ()
3-Modify GPA()
4- Get Student Address ()
5- Print Student Info ()

What is Class ? Why we need It ?

Class Student

Data:

1- Student_name
2- University_Id
3- Birth_Date
4- Address
5-GPA
6- Study_Level
7- Email

Operations ( )

1- Get Student Name ()
2- Change Study level ()
3-Modify GPA()
4- Get Student Address ()
5- Print Student Info ()

Student 1

Data:

1- Student_name =Adam
2- University_Id =1232024
3- Birth_Date =20/10/2003
4- Address =Hama TTT
5-GPA = 3.5
6- Study_Level = 6
7- Email = Adam@YY.com

Operations ( )

1- Get Student Name ()
2- Change Study level ()
3-Modify GPA()
4- Get Student Address ()
5- Print Student Info ()

What is Class ? Why we need It ?

Class Student

Data:

1- Student_name
2- University_Id
3- Birth_Date
4- Address
5-GPA
6- Study_Level
7- Email

Operations ( )

1- Get Student Name ()
2- Change Study level ()
3-Modify GPA()
4- Get Student Address ()
5- Print Student Info ()

Student 1

Data:

1- Student_name =Adam
2- University_Id =1232024
3- Birth_Date =20/10/2003
4- Address =Hama TTT
5-GPA = 3.5
6- Study_Level = 6
7- Email = Adam@YY.com

Operations ( )

1- Get Student Name ()
2- Change Study level ()
3-Modify GPA()
4- Get Student Address ()
5- Print Student Info ()

Student 1

Data:

1- Student_name =Sam
2- University_Id =1242025
3- Birth_Date =10/5/2003
4- Address =homs III
5-GPA = 3.6
6- Study_Level = 6
7- Email = Sam@YY.com

Operations ( )

1- Get Student Name ()
2- Change Study level ()
3-Modify GPA()
4- Get Student Address ()
5- Print Student Info ()

جَامعة المَنارة
MANARA UNIVERSITY

انتهت المحاضرة الأولى من الأسبوع 1

جَامعة المَنارة
MANARA UNIVERSITY

قسم الهندسة المعلوماتية

برمجة 3

Java Programming

ا. د. علي عمران سليمان

المحاضره الثانية من الأسبوع الأول

الفصل الصيفي2023-2024

Contents 1

1. Objects and Classes .	11. Inheritance and Constructors.
2. UML class diagrams .	12. Overriding Superclass Methods.
3. Performing output Displaying with print, println, printf.	3.6 Class JOptionPane Using Dialog Boxes showMessageDialog(), showInputDialog()
4. Performing Input Scanner and some of its methods.	4.15 GUI &Graphics,
5. default constructor.	4.15 Creating Simple Drawings—Displaying and drawing lines on the screen
6. Overloaded Constructors, and methods.	5.11 Drawing Rectangles and Ovals—Using shapes to represent data.
7. Static Method , and Data fields.	
8. Call by value and references.	
9. copy constructor.	
10. inherited class.	

References

- Deitel & Deitel, Java How to Program, Pearson; 10th Ed(2015)
- د.علي سليمان، بنى معطيات بلغة JAVA، جامعة تشرين 2013-2014

Objects and Classes 1

- Class: الصنف كود يصف أنماط الكائنات التي يمكن أن تشتق منه أي مفهوم مجرد لايمثل في ذاكرت الحاسب، يصف المعطيات التي يمكن للكائن أن تملكها (حقول المعطيات، أو المعطيات الأعضاء، الخصائص Properties جميعها تعبر عن الصفات Attribute)، والأحداث التي يمكن للكائن أن يتضمنها(المناهج methods، الأداء Actions ، العمليات operations جميعها تعبرعن السوكيات behaviors).
  - يمكن التفكير بالصنف كمخطط blueprint لبناء الكائنات الفعلية (نمط).
  - عند جريان البرنامج، يمكن أن تستخدم الاصناف من أجل بناء العديد من الكائنات في الذاكرة وبالأنواع المطلوبة.
  - الصنف هو مصدر الكائنات وتعرف بأمثال الصنف.

Objects and Classes 2

Example:

This expression creates a Scanner object in memory.

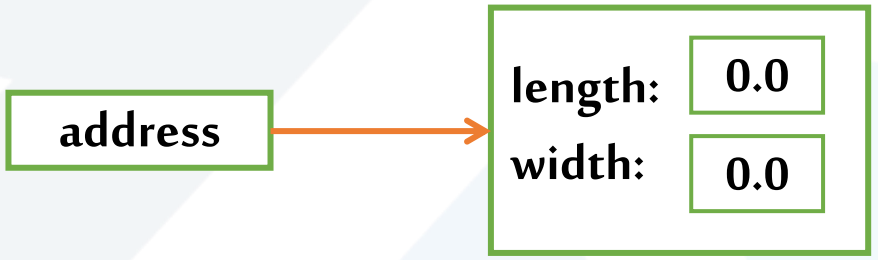
```
Scanner input = new Scanner(System.in);
```

The object's memory address is assigned to the input variable.

Creating a Rectangle object

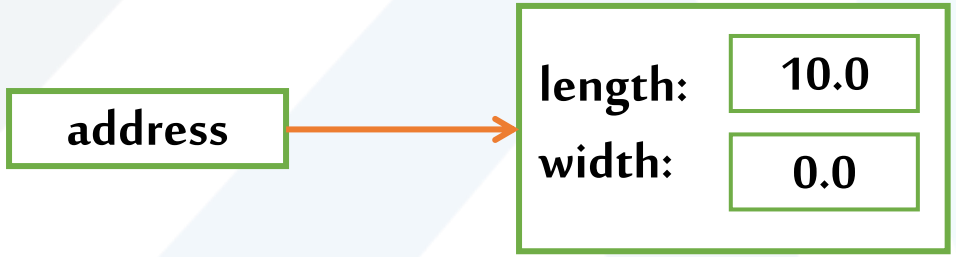
The box variable holds the address of the Rectangle object.

```
Rectangle box = new Rectangle ();
```



Calling the setLength Method

```
box.setLength(10.0);
```



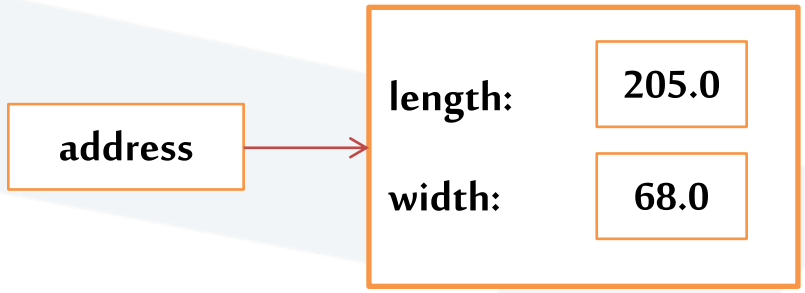
This is the state of the box object after the setLength method executes.

Instance Fields and Methods

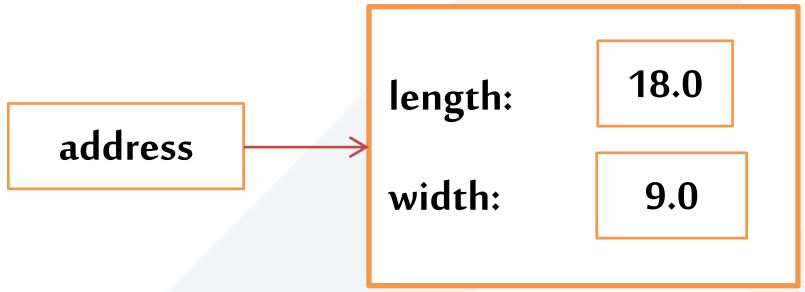
- الحقول والمناهج المصرح عنها سابقاً تدعى أمثال حقول وأمثال مناهج.
- كل كائن من صنف يملك نسخة خاصة به عن أمثال الحقول وأمثال مناهج.
- والمنهج المثل هو المنهج الذي لم يعرف كا static.
- الأمثال للحقول والمناهج تتطلب إنشاء كائن لاستخداماتها اللاحقة والمناهج static ترتبط بالصنف ولاتحتاج لكائن كي تستثمر بل يكتب اسم الصنف واسم المنهج وبينهما نقطة.
- كل مستطيل (كائن) يمكن أن يملك ابعاده الخاصة به.

States of Three Different Rectangle Objects

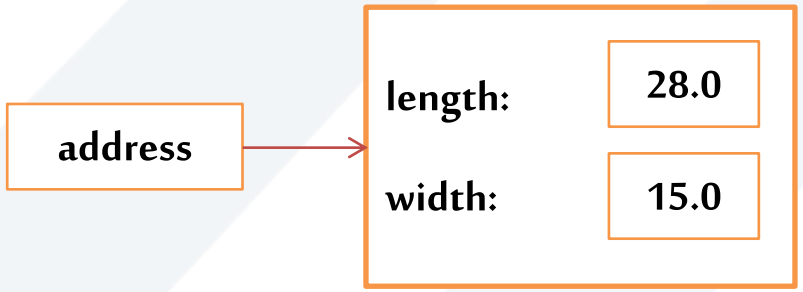
The football **variable holds the address of a Rectangle Object.**



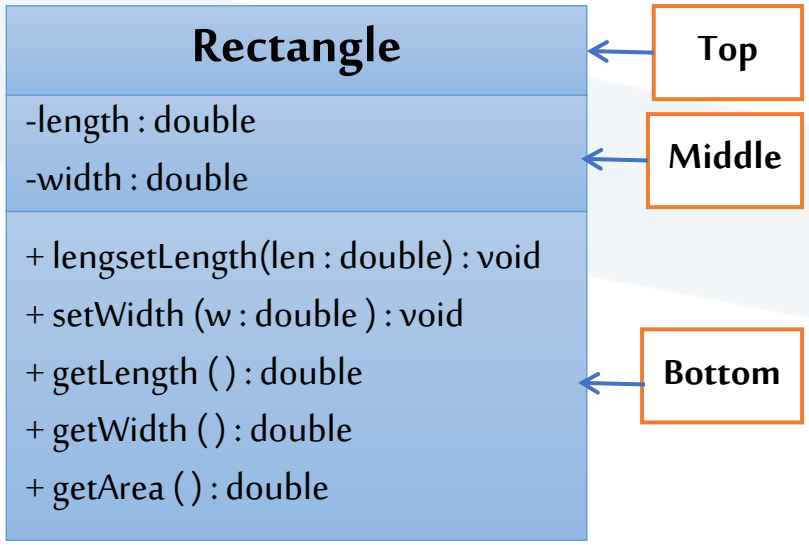
The Volleyball **variable holds the address of a Rectangle Object.**



The basketball **variable holds the address of a Rectangle Object.**



UML class diagrams 1



- يستخدم مخطط UML (Unified Modeling Language) للصنف لتلخيص صفات ومناهج الصنف.
- تساعد مخططات UML مصممي الأنظمة على تحديد نظام بطريقة رسومية ومستقلة عن لغة البرمجة ، قبل أن يتم تنفيذ النظام بلغة برمجة ما.
- في تخطيط UML يمثل الصنف بمستطيل يحوي ثلاث مكونات compartments.
- الجزء العلوي : يتضمن أسم الصنف بخط عريض وبهامش توسط .
- الجزء الوسطي : يتضمن أسماء الحقول وتسبق باشارة – للنوع الخاص، + للعام، و# للنوع المحمي ويليه : وبعدها نوع معطيات الحقل.
- الجزء السفلي : يتضمن أسماءالمناهج حيث يذكر اسم المنهج ومسبوقاً بإشارة نوع الوصول (+ , # , -) عام, محمي, خاص على الترتيب كما هو في حالة أسماء الحقول.
  - قوسان دائريان فارغان إذا لم يمرر متغيرات للمنهج، وفي حال تمريرها يذكر الأسم للمتغير يلي كل منها : ثم نمطه وتفصل المتغيرات (الاسماء) عن بعضها بعضاً بفواصل عادية.
  - يوضع مابعد القوس الدائري المغلق : ونوع القيمة المعادة وإذا كان لايعيد شيئ يوضع void.
  - الباني يوضع في حال كتابته وفق الأتي: يوضع في بداية المناهج بذكر مابين «*constructor*» ثم اسم صنفة وهو اسمه بطبيعة الحال وبعدها ينطبق عليه ماينطبق على المناهج الأخرى، إلا أنه لن يعيد شيئ ولا حتى void.

Writing a Class

- **A Rectangle object will have the following fields and methods:**

```
public class Rectangle
{       private double length;
        private double width;
         public Rectangle () { System.out.println("default constructer");}
        public Rectangle (double l, double w) {length=l;width = w;
                        System.out.println("constructer with argument");}
        public void setLength(double le) { length=le;}
        public void setWidth(double wi) {width = wi;}
        public double getLength()  {  return length; }
        public double getWidth()  {   return width;}
        public double perRectangle() {return (length + width)*2;          }
        public double areaRectangle()  { return length * width;           }
}
```

Setter , Mutator

Getter, Accessor

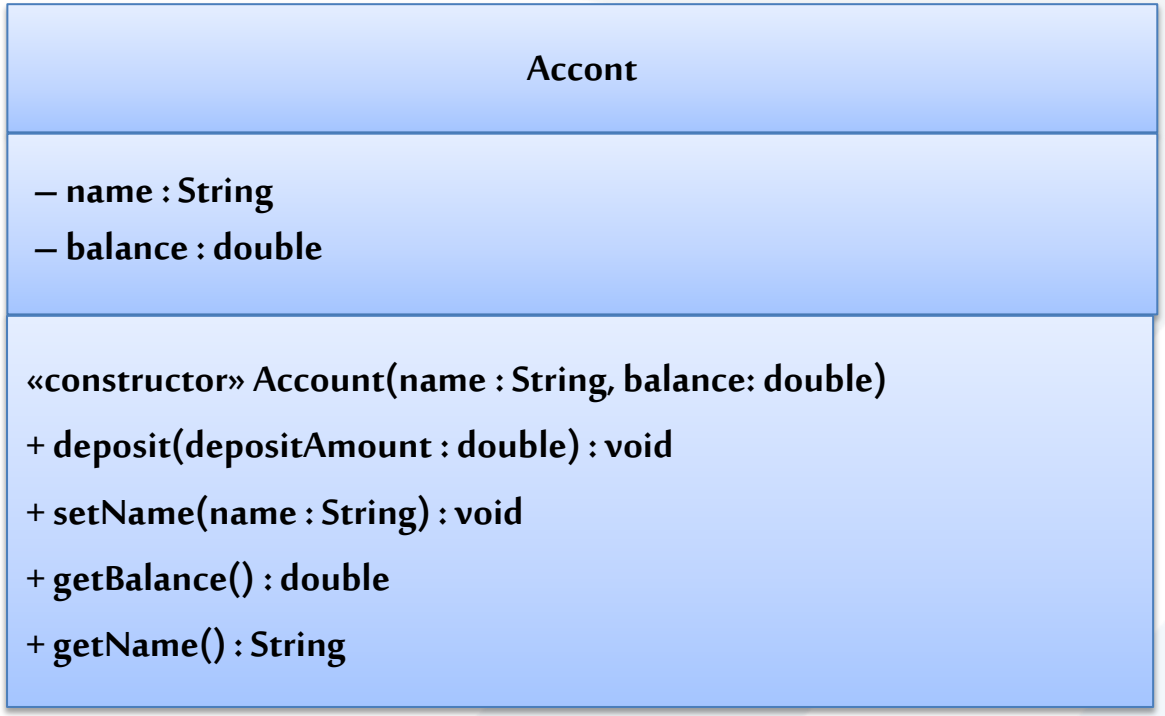
Rectangle
-length : double -width : double
+«constructor» Rectangle () +«constructor» Rectangle(length double , width: double) + setLength(length : double) : void + setWidth (width : double) : void + getLength ( ) : double + getWidth ( ) : double + perRectangle( ) : double + areaRectangle ( ) : double

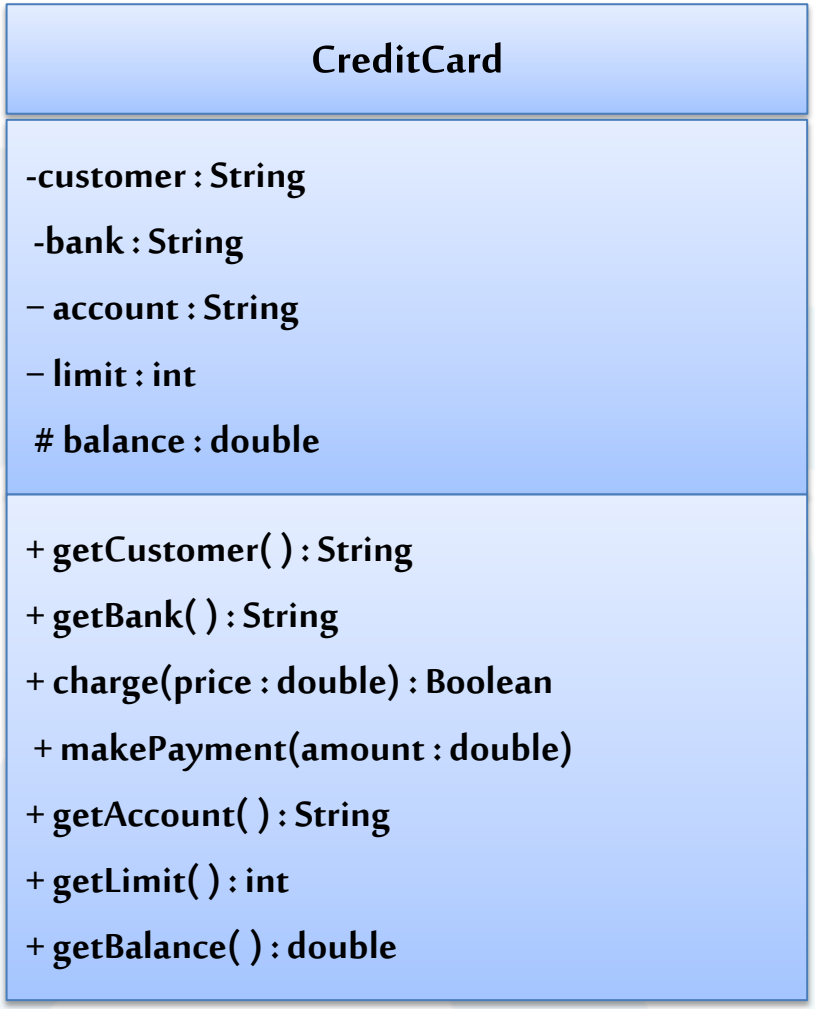
Rectangle box1 = new Rectangle(); Rectangle box2 = new Rectangle(15.0, 10.0);

Class test

```
public class RectangleTest {
        public static void main(String[] args) {
        Rectangle r1= new Rectangle();
        System.out.println("length= "+ r1.getLength() + "width= "+ r1.getWidth());
        r1.setLength(11);
        r1.setWidth(22);
        System.out.println("length= "+ r1.getLength() + "width= "+ r1.getWidth());
         //If the height and width are made public, we can directly access them
        //1 System.out.println("length= "+ r1.length + "width= "+ r1.width);
        Rectangle r2= new Rectangle(30, 20);
        System.out.println("length= "+ r2.getLength() + "width= "+ r2.getWidth());
        System.out.println("perimeter= "+ r2.perRectangle() + "\narea= "+ r2.areaRectangle());
        //1 System.out.println("length= "+ r2.length + "width= "+ r2.width);
    }
}
```

Examples 2


Accont
– name : String
– balance : double
«constructor» Account(name : String, balance: double)
+ deposit(depositAmount : double) : void
+ setName(name : String) : void
+ getBalance() : double
+ getName() : String


CreditCard
-customer : String
-bank : String
– account : String
– limit : int
balance : double
+ getCustomer( ) : String
+ getBank( ) : String
+ charge(price : double) : Boolean
+ makePayment(amount : double)
+ getAccount( ) : String
+ getLimit( ) : int
+ getBalance( ) : double

Uninitialized Local Reference Variables

- **Reference variables can be declared without being initialized.**

Rectangle box;

- **This statement does not create a** Rectangle **object, so it is an uninitialized local reference variable.**

- **A local reference variable must reference an object before it can be used, otherwise a compiler error will occur**.

box = new Rectangle();

Displaying with print, println

Performing output 1

- Performing Output with System.out.print

```
System.out.print("Welcome to Java Programming!");
```

- Where System is the class name, it is declared as final. The out is an instance of the System class and is of type PrintStream. Its access specifiers are public and final.
- Class System is part of package java.lang. Notice that class System will not be imported with an import declaration at the beginning of the program.

- توجد ثلاث نسخ من منهج للطباعة print:

1- المنهج print() يوجه الكمبيوتر لتنفيذ إجراء - عرض الأحرف الموجودة بين علامات الاقتباس المزدوجة (لا يتم عرض علامات الاقتباس نفسها) وتعتبر كبارامتر وحيد، وإذا وجد مابين القوسين أكثر من بارامتر مثل أعداد ونصوص وتعابير وجب استخدام (+) لقسرها أو لضمها لبعضها، وسيتم ترك مؤشر الاخراج حيث انتهت الطباعة.

2- منهج println()، عندما تكتمل مهمة المنهج System.out.println()، فإنه ينقل مؤشر الاخراج حيث انتهت الطباعة إلى بداية السطر التالي.

Performing output 2

Escape sequence

- Escape

sequence	Description
\n	Newline. Position the screen cursor at the beginning of the next line.
\t	Horizontal tab. Move the screen cursor to the next tab stop.
\r	Carriage return. Position the screen cursor at the beginning of the current line—do not advance to the next line. Any characters output after the carriage return overwrite the characters previously output on that line.
\\	Backslash. Used to print a backslash character.
\"	Double quote. Used to print a double-quote character. For example, System.out.println("\"in quotes\""); displays "in quotes".

Some common escape sequences.

Displaying with printf

Performing output 3

- **The** System.out.printf **method (f means** "formatted"**) displays formatted data.**

```
System.out.printf("%s%n%s%n", "Welcome to", "Java Programming!");
//System.out.printf("%4d %, 20.2f\n %n", year, amount);//left justified %-20s
```

3- منهج printf() يحتاج المنهج إلى ثلاث وسطاء أو بارامترات، يتم وضعها في القائمة. تبدأ محددات التنسيق بعلامة النسبة المئوية (%) متبوعة بحرف يمثل نوع البيانات ثم فاصمة، تفصل البيانات عن بعضها البعض بفواصل (يمكن للبيانات ان تتضمن تعبير).
(مثلاً: محدد التنسيق %s هو عنصر يدل لسلسلة بحروف صغيرة ، %S سلسلة بحروف كبيرة، %d عنصر يدل لقيمة عدد صحيح، %f عنصر يدل لقيمة حقيقية،

محدد التنسيق %,20.2f : حيث % علامة التنسيق، الفاصلة (,) يشير إلى أنه يجب إخراج قيمة النقطة العائمة بفاصل تجميع كل ثلاث ارقام بمجموعة ومؤلفه من 20 مرتبة ، مرتبتين عشريتين، غياب الإشارة قبل 20 أي موجبة وبالتالي الهامش يميني) .

- لذا فإن هذا المثال الأول يستبدل "Welcome to " مع %s الأول والانتقال للسطر التالي بفعل %n وبعدها يستبدل "Java Programming! "مع %s الثانيه ثم الانتقال للسطر التالي من %n الثانية، ويمكن استخدام \n بدل %n مع printf.
- لايمكن استخدام %n للانتقال للسطر التالي بدل \n في وسطاء System.out.print(); أو System.out.println();
- ملاحظة: سيتم تغطية هذه الحالات في تمرين اخر المحاضرة.

Performing input 1

Input with input.nextInt();

```
import java.util.Scanner; // program uses class Scanner
// create a Scanner to obtain input from the command window
 int number ;
```

- تصريح الاستيراد الذي يساعد المترجم على تحديد صنف يتم استخدامه في هذا البرنامج. يشير إلى أن البرنامج يستخدم صنف Scanner المحددة مسبقًا من الحزمة المسماة **java.util** .

- يحجز للمتغير موقع في ذاكرة الكمبيوتر حيث يمكن تخزين القيم لاستخدامها لاحقًا في أحد البرامج. يجب التصريح عن المتغيرات باسم ونوع قبل استخدامها. يتيح اسم المتغير للبرنامج الوصول إلى القيمة الموجودة في الذاكرة.

Performing input 2

new Scanner(System.in);

Scanner input = new Scanner(System.in);

in is basically the instance of ***InputStream*** from ***java.lang*** package.

- إعلان عن كائن بالاسم (input) ونوع (Scanner) للكائن المستخدم في هذه التعليمة. يسمح **Scanner** للبرنامج بقراءة البيانات لاستخدامها في البرنامج.

يمكن أن تأتي البيانات من عدة مصادر، مثلاً من المستخدم عبر لوحة المفاتيح أو ملف على القرص. قبل استخدام Scanner، يجب أن تقوم بإنشائه وتحديد مصدر البيانات.

يجب تهيئة الكائن input من النمط Scanner من خلال منادات الباني العبارة في الطرف اليميني من إشارة النسب.

- new Scanner(System.in); يستخدم هذا التعبير الكلمة الأساسية new لإنشاء كائن Scanner في ذاكرة الحاسب من خلال منادات المنهج الباني للصنف Scanner مع ارسال بارامتر من النوع System.in له، ليقرأ المدخلات التي كتبها المستخدم على لوحة المفاتيح كونها وحدة الإدخال القياسية.

number = input.nextInt();

Performing input 3

```
System.out.print("Enter integer number: "); // prompt read int number from user
number = input.nextInt(); // read Double number, line nextDouble(), nextLine()
```

- يستخدم المنهج nextInt من الكائن input لإدخال عدد صحيح من المستخدم عبر لوحة المفاتيح. في هذه المرحلة، ينتظر البرنامج أن يقوم المستخدم بكتابة الرقم والضغط على مفتاح Enter لإرسال الرقم إلى المكان المحجوز للمتغير number المعرف كعدد صحيح قبل استخدامه.
- نضع نتيجة استدعاء الأسلوب nextInt() في المتغير number باستخدام عامل التخصيص = .
- عامل التخصيص= يسمى العامل الثنائي لأنه يحتوي على معاملين ونتيجة لاستدعاء الأسلوب input.nextInt(). المتغير والقيمة التي ستنسب له.

يستخدم input .nextDouble() لإدخال عدد حقيقي و input.nextFloat() لإدخال عدد , float و عند الإدخال يضاف لنهاية العدد f و nextLine لادخال شريط محرفي

و;char s= input.next().charAt(0) لإدخال واسناد المحرف الأول للمتغير s.

;char str2=in.next().charAt(3)سيتم إدخال أربعة محارف وإسناد المحرف الرابع للمتعير str2.

The Default Constructor

- عند إنشاء كائن سيتم نداء المنهج الباني constructor بشكل تلقائي.
- إذا لم يتم كتابة الباني من قبل المبرمج، ستقوم Java بتوفير provides واحد عند إجراء المطابقة للصنف، والباني المنشئ من Java يعرف بالباني الافتراضي.
  - سيتم الاسناد لكل الحقول العددية بما فيها المحرفية القيمة الصفر.
  - سيتم الاسناد لكل الحقول المنطقية إلى false.
  - سيتم الاسناد لكل الحقول المرجعية إلى null.
- الباني الافتراضي default constructor يكون بدون وسطاء.no-arg ويستخدم للتجهيز بالقيم الافتراضية.
- الباني الافتراضي default constructor لن يتم تنفيذه من قبل Java إذا كان هناك بانياً آخراً مكتوباً ولم تتم كتابته.
- يمكننا كتابة الباني الخاص بنا بدون وسطاء no-arg وفق أحد الحالتين في الأولى يضع القيم الافتراضية 0.0, 0.0 في الثانية يضع القيم 15.0, 10.0

```
public Rectangle(){}
}
Rectangle box1 = new Rectangle();
```

OR

```
public Rectangle() { length = 15.0;width = 10.0;
```

عندها يمكن أن نشتق كائن

Overloading Methods and Constructors

- قد يكون لطريقتين أو أكثر في الصنف نفس الاسم طالما أن قوائم المتغيرات الممرة الخاصة بهم مختلفة نوعاً أو ترتيباً أو كماً أو اكثر من واحده مما سبق.
- يطلق عليه طريقة التحميل الزائد Overloading، وهذا ينطبق أيضا على constructors.
- أسلوب التحميل الزائد مهم لأنك في بعض الأحيان تحتاج إلى عدة طرق مختلفة الإجراء وبنفس الاسم.

```
public int add(int num1, int num2)
{ int sum = num1 + num2; return sum; }

public String add (String str1, String str2)
{ String combined = str1 + str2; return combined; }
```

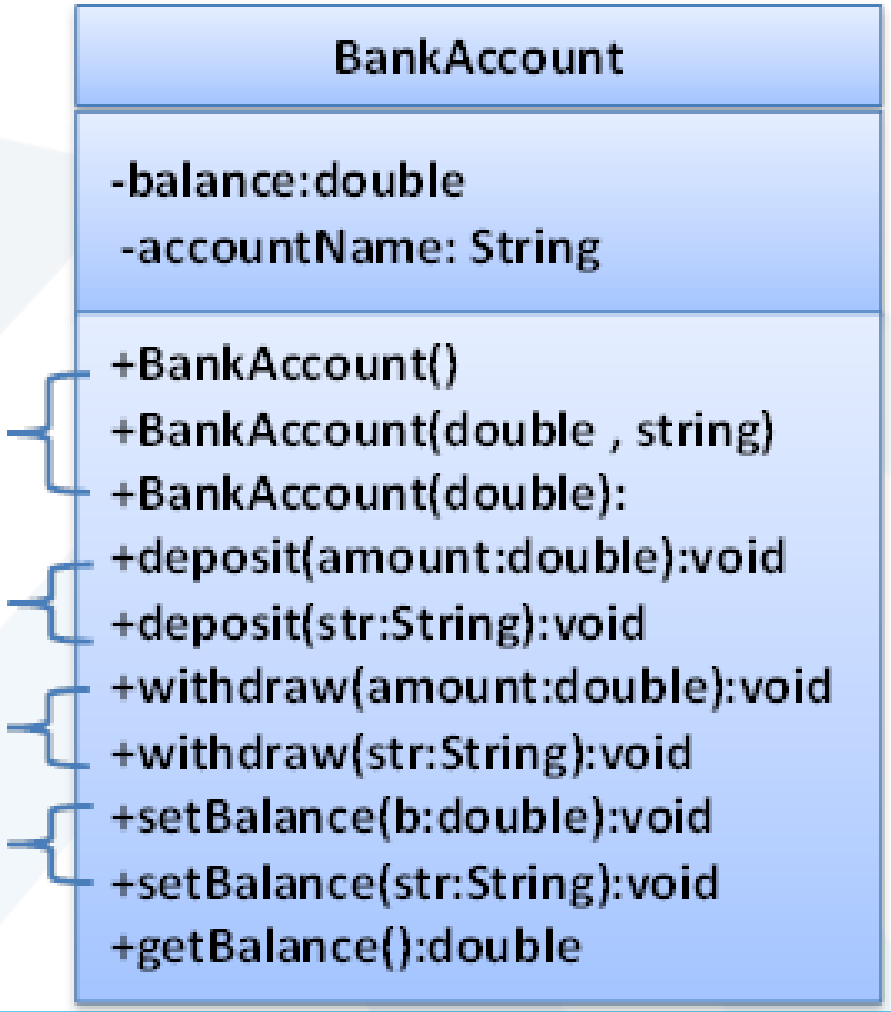
The BankAccount Example

Overloaded Constructors

Overloaded deposit methods

Overloaded withdraw methods

Overloaded setBalance methods



Static Class Members

- متى نحتاج لاعضاء الصنف (مناهج وحقول) أن تكون static ثابته؟
- عندما نرغب بأن تتبع هذه الاعضاء للصنف وليس للكائن فقط، أي عندما يكون المنهج خدمي utility classes (مثل توابع الرياضيات غير المعرفة في صنف Math) من مكتبة Java القياسية.
وعندما يكون الحقل تابع لكل الكائنات (مثلاً عد الكائنات المشتقه من صنف) أي حقل مشرك.
- عندما نحتاج مناهج لتشغيلها ولانرغب بتخزين الكائنات، أي نحن لانرغب بتخزين معطيات عن هذه الكائنات فلا داعي لانشائها، ونوفر أماكن في الذاكرة كانت ستخصص للكائنات.
- لاستدعاء طريقة static أو استخدام حقل معطيات static يتم استخدام اسم الصنف، بدلاً من اسم الكائن.
مثال استدعاء منهج الجزر التربيعي من الصنف Math وإعطائها القمة 25.

- **Example:**

```
double val = Math.sqrt(25.0);
```

Static Fields

- يتم الإعلان عن حقول الصنف بأنها static باستخدام الكلمة المفتاحية static بين محدد الوصول ونوع الحقل.
- يمكن تعريف scope من النمط static كما يلي.

```
class Test {
        static int i;
        static
                { i=100;        float f=13.22f;        String s= "Adam";        }
```

- إن ترتيب خطوات الترجمة تكون:

1-تجهيز initializationالمتغيرات من نمط static .

2- تجهيز initializationالمتغيرات من نمط Dynamic .

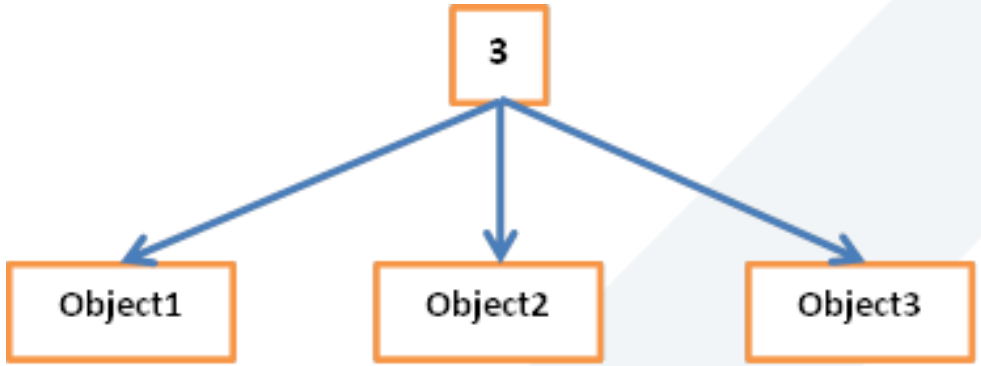
3- مناداة الباني constructer.

Static Fields

- لايمكن تعريف متحول staticضمن الطرق بل محصوربالاصناف.

- يتم تهيئة الحقول الثابتة الأولية إلى 0 إذا لم يتم إجراء تهيئة لها initialization.
- مثال لعد الكائنات المشتقة من صنف نضع ;private static int instanceCount = 0ضمن الصنف ونزيد هذا الحقل بمقدار1 في كل باني للصنف أو أن نبدأ بـ 500 وننقص واحد في كل باني.

- instanceCount field(static).



Static Methods

- يتم الإعلان عن static Methods من خلال وضع الكلمة المفتاحية static بين معدِّل الوصول ونوع القيمة المعاده.

```
public static int getInstanceCount()        { return instanceCount; }
```

- عندما يحتوي صنف على طريقة static، لاداعي لإنشاء مثيل للصنف من أجل استدعاء الطريقة، بفرض أن الطريقة *add* ضمن الصنف *Calc*.

```
System.out.println(Calc.add(121, 112));
```

- تعتبر الطرق static ملائمة لأنها قد يتم استدعاؤها على مستوى الصنف.

- الحقول والطرق static تابعة للصنف وبالتالي في الحالة العامة لا تتواصل الطرق static مع الحقول أو الطرق غير static أي مع **instance** لإنها تابعة للكائنات،بل فقط مع الحقول static.
- يمكن الوصول إليها من الكائنات ولكن ذلك غير محبب.
- تستطيع الطرق غير static التواصل مع الحقول static.

Static Methods

```
public class Calc
{
    private static int instanceCount = 0;
    private int a;          private int b;
    public Calc()           {instanceCount++;}
    public Calc(int x, int y) {    instanceCount++;          a=x;              b=y;     }
    public static  int add(int aa, int bb) {          return aa+bb;    }
    public int sub()                {          return a-b;}
    public int mult()               {return a*b;}
    public static int getInstanceCount()          { return instanceCount;}
}
```

Static Methods

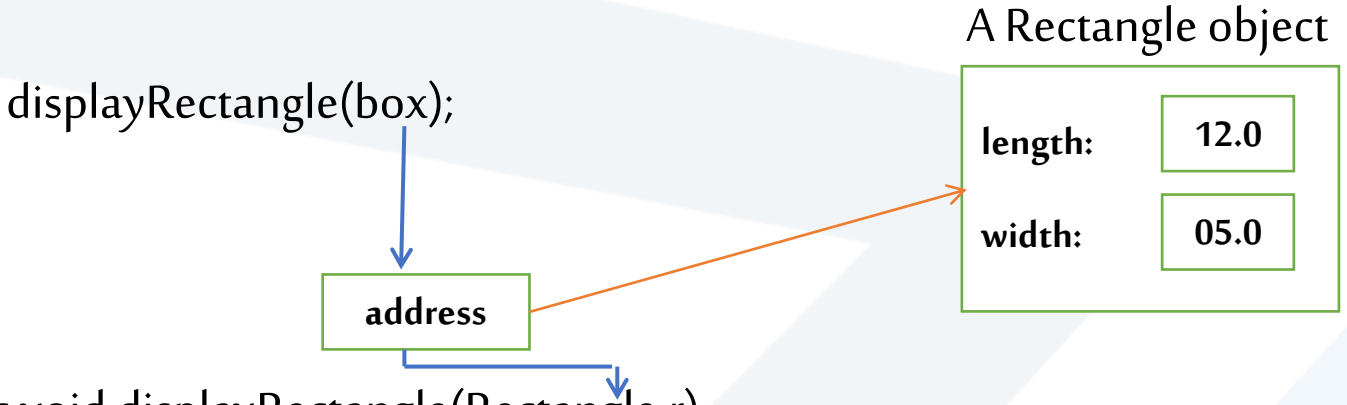
```
public class CalcTest {
    public static void main(String[] args) {
        Calc c1 =new Calc(10,20);Calc c0 =new Calc(); Calc c2 =new Calc(23,17);Calc c3 =new Calc();
        Calc c4 =new Calc(); System.out.println("th mult   is0 = "+c0.mult());
        System.out.println("th mult   is = "+c1.mult());
        System.out.println("th sub   is = "+c2.sub());
         System.out.println(Calc.add(111,222));
        System.out.println("the number of object = "+Calc.getInstanceCount());
        int year=2020; double amount=2334443.446;
        System.out.printf("%4d %, 20.2f\n %n", year, amount);
        System.out.printf("%4d %, 20.2f\n %n", year*100, amount*2);
        System.out.println(year+year);  }//end main
}//end class CalaTest
```

```
th mult   is0 = 0
th mult   is = 200
th sub   is = 6
333
the number of object = 5
2020       2,334,443.45


202000       4,668,886.89


4040
```

Passing Objects as Arguments

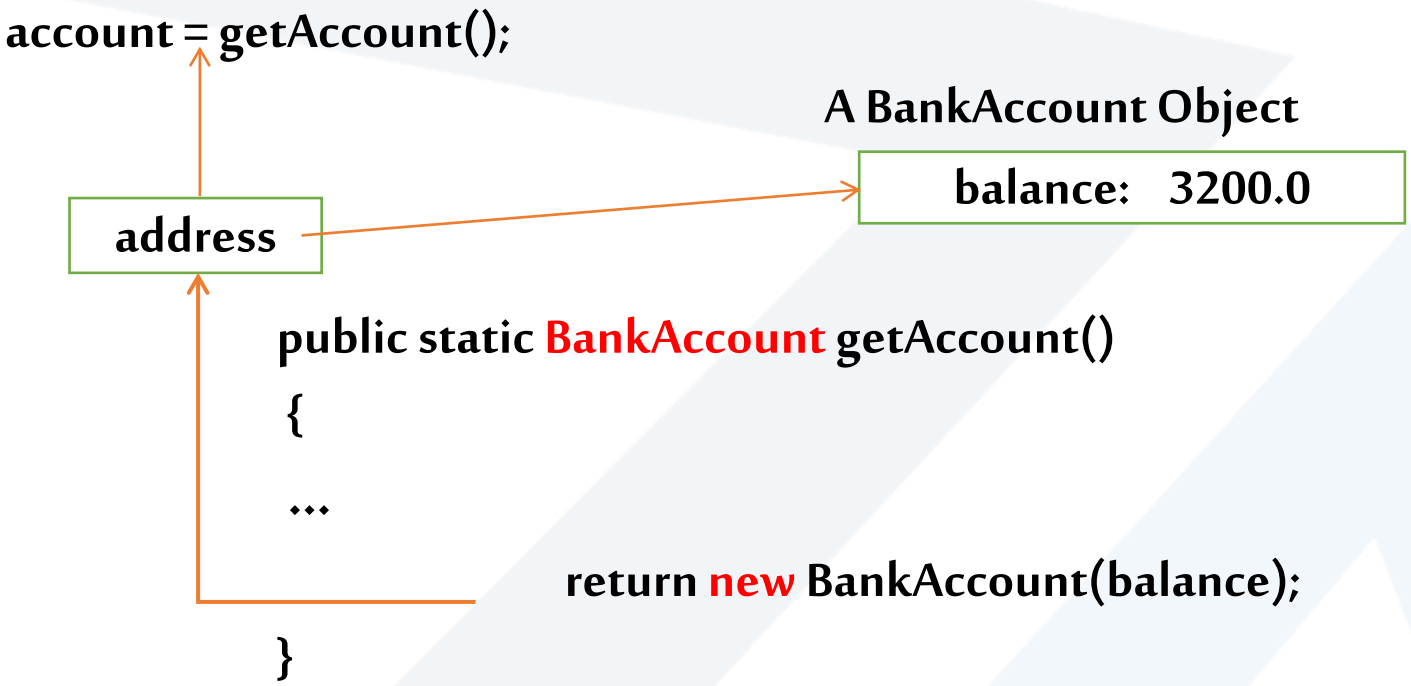
- يمكن تمرير الكائنات إلى الأساليب كوسيطات.
- يقوم Java بتمرير جميع البارامترات للنمط الاولي primitive data type بالقيمة.
- عند تمرير كائن كوسيط ، يتم تمرير قيمة المتغير المرجعي له (العنوان) لا يتم تمرير نسخة من الكائن .
- المتغير المرجعي reference type هو عنوان الكائن في الذاكرة.
- عندما تتلقى طريقة متغير مرجعي كوسيطة ، فإن أية تعديل للكائن ضمن الطريقة تعديل محتويات الكائن المشار إليه بواسطة المتغير، لإن العمل يتم على نفس المكان المحجوز في الذاكرة.

Passing Objects as Arguments




```
public static void displayRectangle(Rectangle r)
{
 // Display the length and width.

System.out.println("Length: " + r.getLength() + " Width: " +r.getWidth());

 }
```

Returning Objects From Methods



Using The == operators with objects

- **If we try the following:**

```
Rectangle r1 = new Rectangle(10,50);
Rectangle r2 = new Rectangle(10,50);
if (r1 == r2) // This is a mistake
        System.out.println("The objects are the same.");
Else
        System.out.println("The objects are not the same.");
```

سيتم طباعة :

The objects are not the same.

only the addresses of the objects are compared.

Methods That Copy Objects

- هناك طريقتان لنسخ كائن.

- لا يمكنك استخدام عامل النسب لنسخ ما يشيرله المراجع (محتوى الكائن) بشكل مباشربل يتم من خلال.

1- نسخة مرجعية فقط: هذا ببساطة هو نسخ عنوان كائن إلى متغير مرجعي لكائن آخر.

2- نسخة عميقة Deep copy يتضمن ذلك إنشاء مثيل جديد للفئة ونسخ القيم من كائن إلى كائن أخر.

الحالة الاولى :

```
Rectangle r1 = new Rectangle(100,50);
Rectangle r2 = new Rectangle(100,50);
r2=r1;
if (r1 == r2)     System.out.println("The objects are the same.");
Else              System.out.println("The objects are not the same.");
```

سيتم طباعة :

```
The objects are the same.
```

لو تم التعديل على إحداهما سينطبق على الاخر نظراً لإن الاثنان يؤشران لنفس المكان ضمن الذاكرة:

```
r2.setLength(66);        System.out.println(r1.getLength()); → 66
```

Methods That Copy Objects

- **A copy constructor accepts an existing object of the same class and clones it** **الحالة الثانية:**

ويمكن أن نحمل الباني بشكل زائد بارسال قيم الحقول وإسنادها أو بارسال كائن له مثل r2 ونسخ حقول المرسل لحقول المبني:

```
public Rectangle () { System.out.println("default constructer");} // end constructor with out argument "default constructor"
public Rectangle (double l, double w) {length=l;width = w;
        System.out.println("constructer with argument");} // end constructor with argument
```

Rectangle r1 = new Rectangle(100,50);

```
public Rectangle(Rectangle r2)
{ length = r2.length; width = r2. width ; }   // end Create copy constructor
```

Rectangle r2 = new Rectangle(r1);

OR

```
Rectangle  r1= new Rectangle(13, 9);          // Create r3, a copy of r1
Rectangle  r3 = Rectangle (r1);
```

انتهت المحاضرة الثانية من الأسبوع 1