



BIRZEIT UNIVERSITY

Objects & Classes

Liang, Introduction to Java Programming and Data Structures,
Twelfth Edition, (c) 2020 Pearson Education, Inc. All rights reserved.





By: Mamoun Nawahdah (Ph.D.)
2022/2023

Objects and Classes

- ❖ An object has both a **state** and **behavior**.
- ❖ The **state** defines the object, and the **behavior** defines what the object does.
- ❖ **Classes** are constructs that define objects of the same type.
- ❖ A Java class uses **variables** to define data fields and **methods** to define behaviors.
- ❖ Additionally, a class provides a special type of methods, known as **constructors**, which are invoked to construct objects from the class.



Circle Class

```

class Circle {
    /** The radius of this circle */
    double radius = 1.0;

    /** Construct a circle object */
    Circle() {
    }

    /** Construct a circle object */
    Circle(double newRadius) {
        radius = newRadius;
    }

    /** Return the area of this circle */
    double getArea() {
        return radius * radius * 3.14159;
    }
}

```

Data field

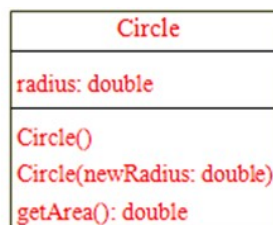
Constructors

Method

3

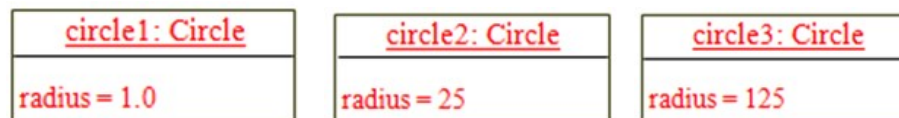
UML Class Diagram

UML Class Diagram



Class name

Data fields

Constructors and
methods

UML notation for objects

4

Constructors

- ❖ Constructors are a **special kind of methods** that are invoked to construct objects.

```
Circle() {  
}  
  
Circle(double newRadius) {  
    radius = newRadius;  
}
```



5

Default Constructor

- ❖ A class may be defined **without** constructors.
- ❖ In this case, a **no-arg constructor** with an empty body is **implicitly** declared in the class.
- ❖ This constructor, called a **default constructor**, is provided **automatically** **ONLY IF** *no constructors are explicitly defined in the class.*



6

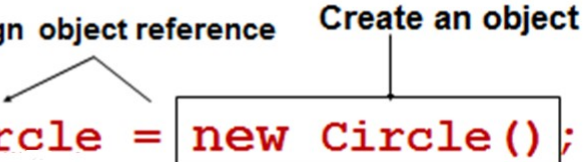
Declaring/Creating Objects in a Single Step

ClassName objectRefVar = new ClassName();

Example:

Assign object reference Create an object

Circle myCircle = new Circle();



7

Accessing Object's Members

❖ Referencing the object's data:

objectRefVar.data

e.g., **myCircle.radius**

❖ Invoking the object's method:

objectRefVar.methodName(arguments)

e.g., **myCircle.getArea()**



8

Default Value for a Data Field

❖ The default value of a data field is:

null for a *reference* type

0 for a *numeric* type

false for a *boolean* type

'\u0000' for a *char* type

❖ However, **Java assigns NO default value to a local variable inside a method.**



9

Instance (**Object**) Variables, and Methods

❖ **Instance variables** belong to a specific instance (**object**).

❖ **Instance methods** are invoked by an instance (**object**) of the class.



10

Static Variables, Constants, and Methods

- ❖ **Static variables** are **shared** by all the instances (objects) of the class.
- ❖ **Static methods** are not tied to a specific instance (object).
- ❖ **Static constants** are **final** variables shared by all the instances (objects) of the class.
- ❖ To declare static **variables**, **constants**, and **methods**, use the **static** modifier.



11

Static Variable

- ❖ It is a variable which belongs to the **class** and not to the **instance (object)**.
- ❖ Static variables are **initialized only once**, at the start of the execution.
 - Static variables will be initialized first, before the initialization of any instance variables.
- ❖ A **single copy** to be shared by all instances of the class.
- ❖ A static variable can be **accessed directly** by the **class name** and doesn't need any object.



Syntax : **<class-name>.<static-variable-name>**

Static Method

- ❖ It is a method which **belongs to the class** and **not** to the instance (**object**).
- ❖ A **static method can access only static data**. It can not access non-static data (instance variables).
- ❖ A **static method can call only other static methods** and can not call a non-static method from it.
- ❖ A static method can be **accessed directly** by the **class name** and doesn't need any object.

Syntax : **<class-name>.<static-method-name>(..)**

- ❖ A static method cannot refer to **"this"** or **"super"** keywords in anyway.



Note: main method is static, since it must be accessible for an application to run, before any instantiation takes place.

Visibility Modifiers

- ❖ **By default** (when no visibility modifiers are used), the **class**, **variable**, or **method** can be accessed by any class in the same **package**.

👉 **public** modifier: The **class**, **data**, or **method** is visible to any class in any package.

👉 **private** modifier: The **data** or **method** can be accessed only by the declaring class.

- ❖ The **get** and **set** methods are used to read and modify private properties.



14

Why Data Fields Should Be private?

- ❖ To protect data.
- ❖ To make code easy to maintain.



15

Example of Data Field Encapsulation

The - sign indicates
private modifier

Circle	
-radius: double	
-numberOfObjects: int	
+Circle()	
+Circle(radius: double)	
+getRadius(): double	
+setRadius(radius: double): void	
+getNumberOfObject(): int	
+getArea(): double	

The radius of this circle (default: 1.0).

The number of circle objects created.

Constructs a default circle object.

Constructs a circle object with the specified radius.

Returns the radius of this circle.

Sets a new radius for this circle.

Returns the number of circle objects created.

Returns the area of this circle.



16

Overloading Methods/Constructors

- ❖ In a class, there can be **several methods with the same name**. However they **must** have **different signature**.
- ❖ The signature of a method is comprised of its **name**, its **parameter types** and the **order of its parameter**.
- ❖ The signature of a method is **not** comprised of its **return type** nor **its visibility** nor its **thrown exceptions**.



Passing Objects to Methods

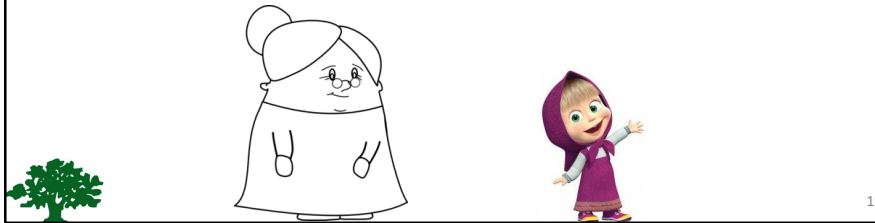
- ❖ **Passing by value** for primitive type value:
 - The **value** is passed to the parameter
- ❖ **Passing by reference** for reference type value:
 - The value is the **reference** to the object



18

Immutable Objects and Classes

❖ If the contents of an object (instance) **can't** be changed once the object is created, the object is called an ***immutable object*** and its class is called an ***immutable class***.



What Class is Immutable?

- ❖ For a class to be immutable:
 - It must mark all data fields **private**.
 - Provide **no set** methods.
 - No **get** methods that would return a reference to a mutable data field object.



Scope of Variables

- ❖ The scope of **instance** (object) and **static** variables is the entire class. They can be declared anywhere inside a class.
- ❖ The scope of a **local** variable starts from its declaration and continues to the end of the block that contains the variable.
- ❖ A local variable **must** be initialized explicitly before it can be used.



21

The **this** Keyword

- ❖ The **this** keyword is the name of a **reference** that refers to an **object itself**.
- ❖ One common use of the **this** keyword is reference a class's *hidden data fields*.
- ❖ Another common use of the **this** keyword to enable a **constructor** to invoke another **constructor** of the same class.



22

Calling **Overloaded** Constructor

```
public class Circle {  
    private double radius;  
  
    public Circle(double radius) {  
        this.radius = radius;  
    }  
  
    public Circle() {  
        this(1.0);  
    }  
  
    public double getArea() {  
        return this.radius * this.radius * Math.PI;  
    }  
}
```

this must be explicitly used to reference the data field radius of the object being constructed

this is used to invoke another constructor

Every instance variable belongs to an instance represented by **this**, which is normally omitted



23