

Object-Oriented Programming with Java **Classes**



Quan Thanh Tho, Ph.D.

CSE – HCMUT

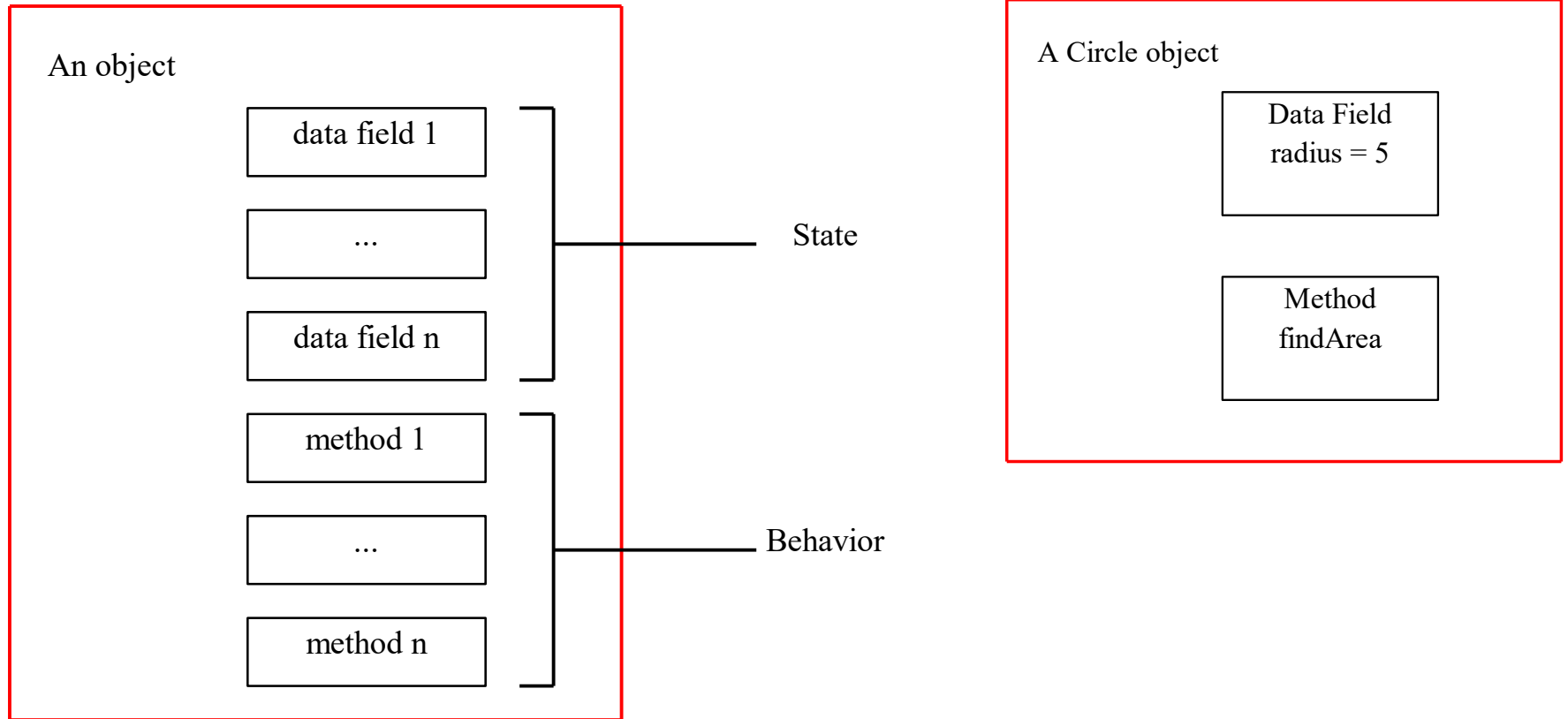
qttho@cse.hcmut.edu.vn

Outlines

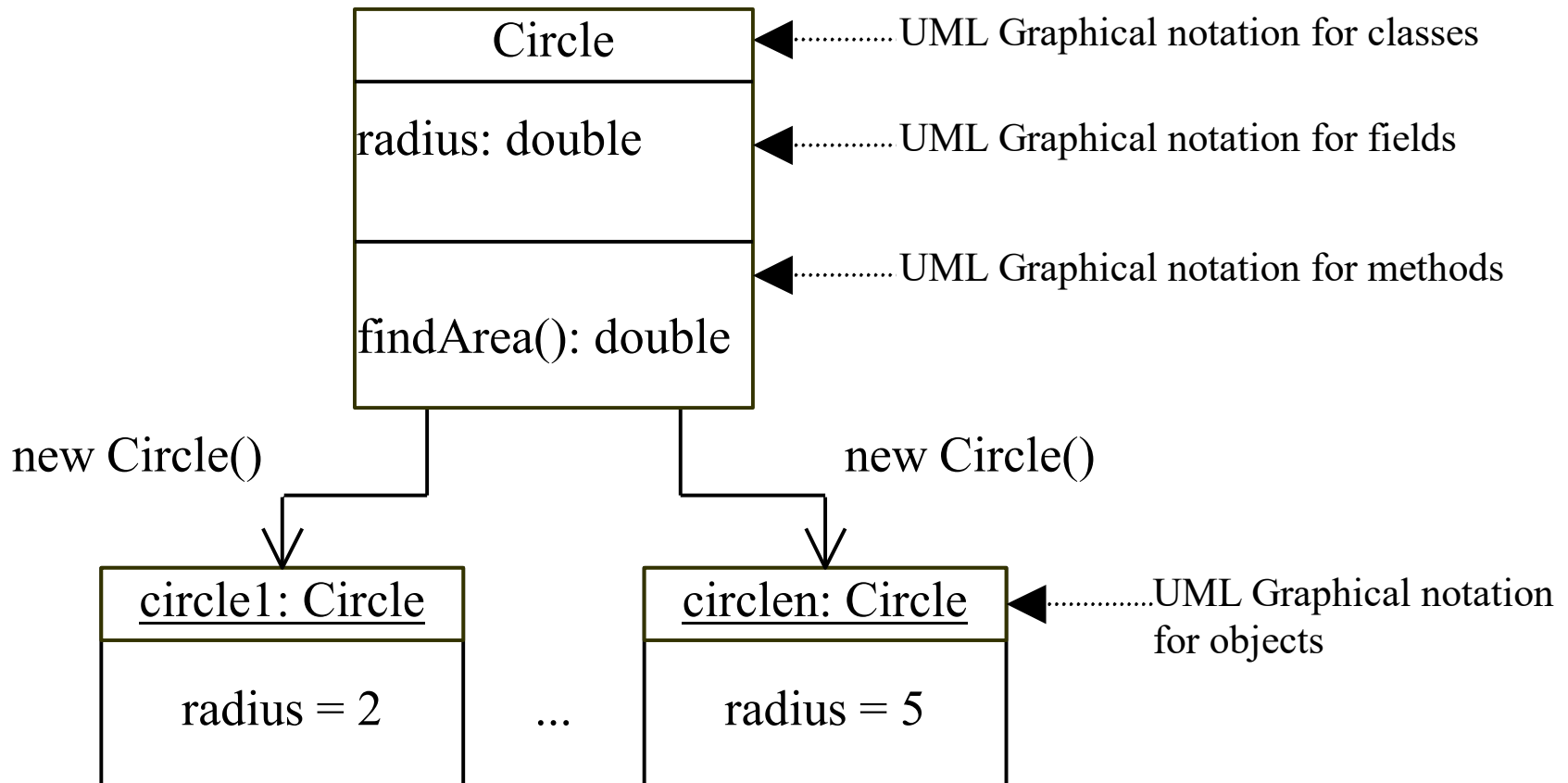


- OO Programming Concepts
- Creating Objects and Object Reference Variables
 - Differences between primitive data type and object type
 - Automatic garbage collection
- Constructors
- Modifiers (`public`, `private` and `static`)
- Instance and Class Variables and Methods
- Scope of Variables
- Use the `this` Keyword
- Case Studies (`Mortgage` class and `Count` class)

OO Programming Concepts



Class and Objects



Class Declaration

```
class Circle
{
    double radius = 1.0;

    double findArea()
    {
        return radius*radius*3.14159;
    }
}
```

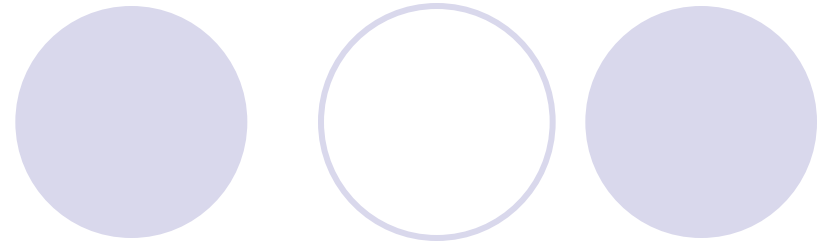
Declaring Object Reference Variables

```
ClassName objectName;
```

Example:

```
Circle myCircle;
```

Creating Objects



```
objectName = new ClassName();
```

Example:

```
myCircle = new Circle();
```

The object reference is assigned to the object reference variable.

Declaring/Creating Objects in a Single Step

Four decorative circles are positioned at the top of the slide. From left to right: a solid light purple circle, a hollow light purple circle, a solid light purple circle, and a hollow light purple circle.

```
ClassName objectName = new  
    ClassName();
```

Example:

```
Circle myCircle = new Circle();
```


Differences between variables of primitive Data types and object types

Primitive type

int i = 1

i

1

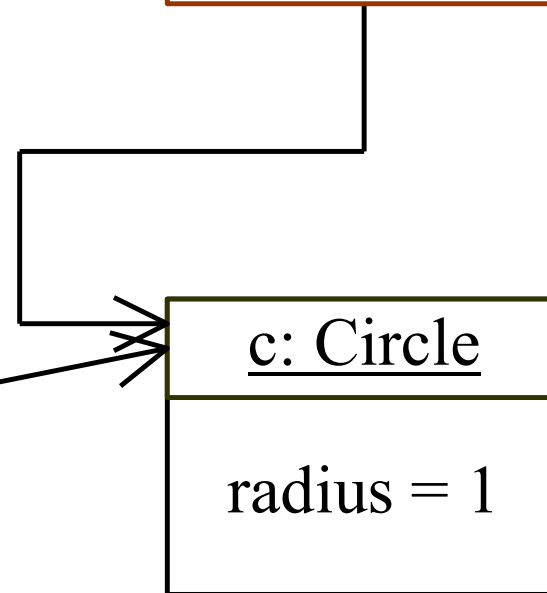
Object type

Circle c

c

reference

Created using
new Circle()



Copying Variables of Primitive Data Types and Object Types

Primitive type assignment
 $i = j$

Object type assignment
 $c1 = c2$

Before:

i 1

j 2

After:

i 2

j 2

Before:

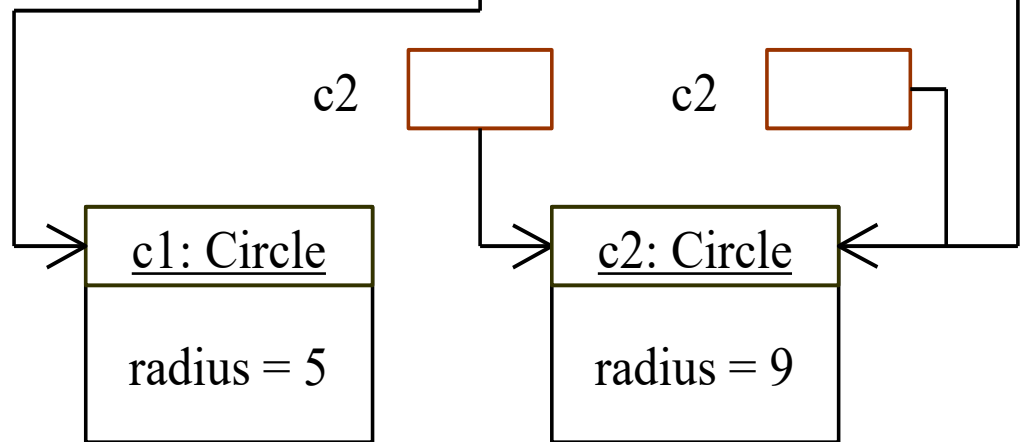
c1

c2

After:

c1

c2



Garbage Collection



As shown in the previous figure, after the assignment statement `c1 = c2`, `c1` points to the same object referenced by `c2`. The object previously referenced by `c1` is no longer useful. This object is known as garbage. Garbage is automatically collected by JVM.

Garbage Collection, cont



TIP: If you know that an object is no longer needed, you can explicitly assign null to a reference variable for the object. The Java VM will automatically collect the space if the object is not referenced by any variable.

Accessing Objects

- Referencing the object's data:

`objectName.data`

`myCircle.radius`

- Invoking the object's method:

`objectName.method`

`myCircle.findArea()`



Example 6.1 Using Objects

- Objective: Demonstrate creating objects, accessing data, and using methods.

[TestCircle](#)

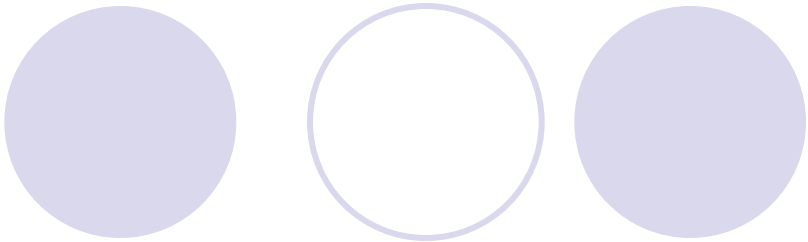
Run

Constructors

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

```
Circle()
{
    radius = 1.0;
}
```

```
myCircle = new Circle(5.0);
```



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

Constructors, cont.

A constructor with no parameters is referred to as a *default constructor*.

- Constructors must have the same name as the class itself.
- Constructors do not have a return type—not even void.
- Constructors are invoked using the new operator when an object is created. Constructors play the role of initializing objects.


Example 6.2 Using Constructors

- Objective: Demonstrate the role of constructors and use them to create objects.

[TestCircleWithConstructors](#)

Run

Visibility Modifiers and Accessor Methods



By default, the class, variable, or data can be accessed by any class in the same package.

- `public`

The class, data, or method is visible to any class in any package.

- `private`

The data or methods can be accessed only by the declaring class.

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

Example 6.3

Using the `private` Modifier and Accessor Methods

In this example, private data are used for the radius and the accessor methods `getRadius` and `setRadius` are provided for the clients to retrieve and modify the radius.

[TestCircleWithAccessors](#)

Run

Passing Objects to Methods

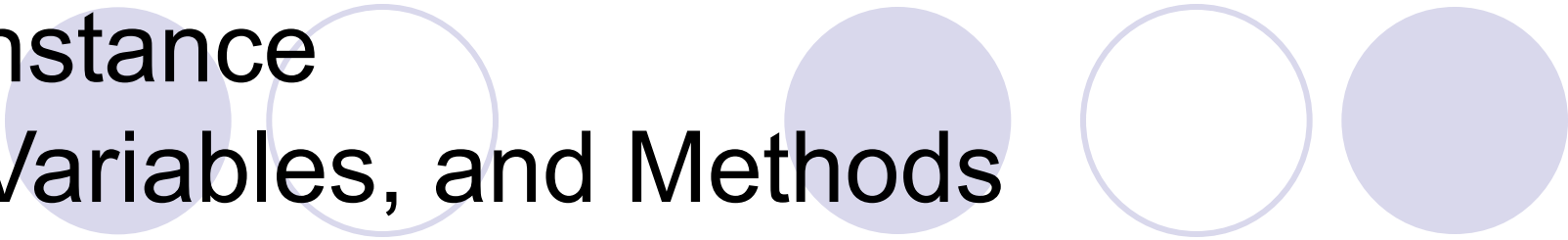
- Passing by value (the value is the reference to the object)

Example 6.3 Passing Objects as Arguments

[TestPassingObject](#)

Run

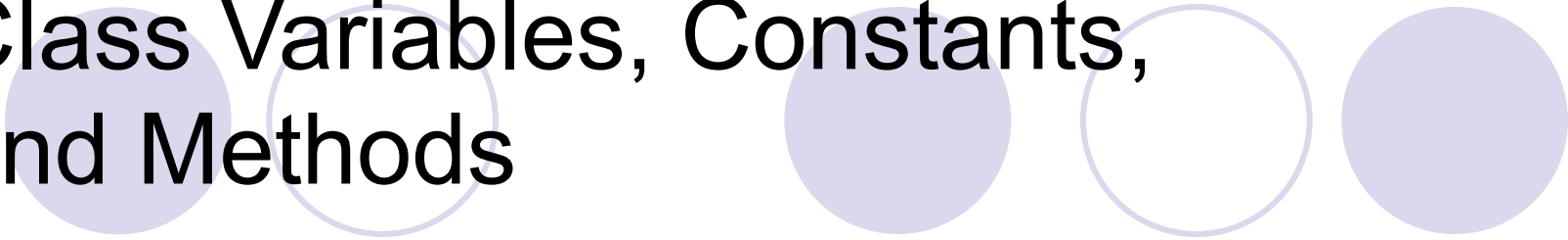
Instance Variables, and Methods



Instance variables belong to a specific instance.

Instance methods are invoked by an instance of the class.

Class Variables, Constants, and Methods



Class variables are shared by all the instances of the class.

Class methods are not tied to a specific object.

Class constants are final variables shared by all the instances of the class.

Class Variables, Constants, and Methods, cont.



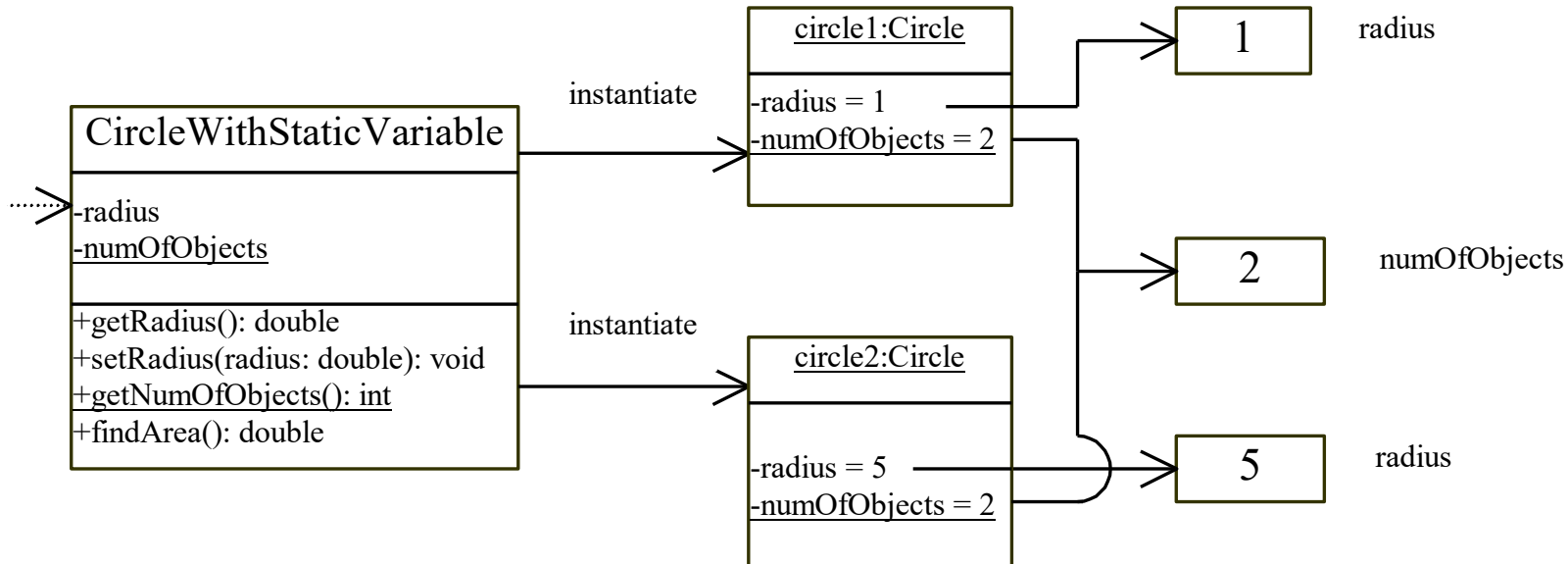
To declare class variables, constants, and methods, use the static modifier.

Class Variables, Constants, and Methods, cont.

UML Notation:

- +: public variables or methods
- : private variables or methods
- underline: static variables or methods

radius is an instance variable, and numObjects is a class variable



Example 6.5

Using Instance and Class Variables and Method

Objective: Demonstrate the roles of instance and class variables and their uses. This example adds a class variable numObjects to track the number of Circle objects created.

[Test CircleWithStaticVariable](#)

Run

Scope of Variables



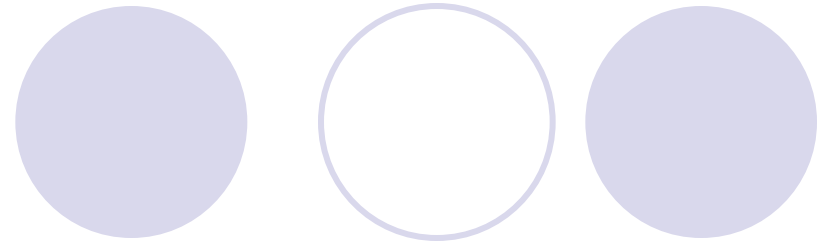
- The scope of instance and class variables is the entire class. They can be declared anywhere inside a class.
- The scope of a local variable starts from its declaration and continues to the end of the block that contains the variable. A local variable must be declared before it can be used.



The Keyword `this`

- Use `this` to refer to the current object.
- Use `this` to invoke other constructors of the object.

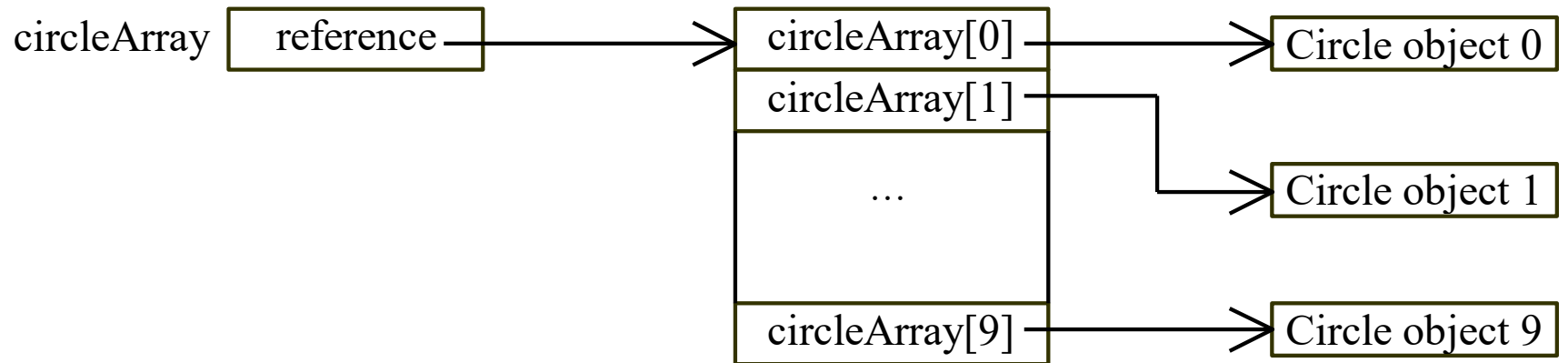
Array of Objects



```
Circle[] circleArray = new  
Circle[10];
```

An array of objects is actually an *array of reference variables*. So invoking `circleArray[1].findArea()` involves two levels of referencing as shown in the next figure. `circleArray` references to the entire array. `circleArray[1]` references to a `Circle` object.

Array of Objects, cont.



Array of Objects, cont.

Example 6.6: Summarizing the areas of the circles

Demonstrate the roles of instance and class variables and their uses. This example adds a class variable `numOfObjects` to track the number of `Circle` objects created.

TotalArea

Run

The title 'Class Abstraction' is positioned on the left side of the slide. To its right, there are five circles arranged horizontally. The first circle is solid light purple. The second circle is a light purple outline. The third circle is solid light purple. The fourth circle is a light purple outline. The fifth circle is solid light purple.

Class Abstraction

Class abstraction means to separate class implementation from the use of the class. The creator of the class provides a description of the class and let the user know how the class can be used. The user of the class does not need to know how the class is implemented. The detail of implementation is encapsulated and hidden from the user.

Example 6.7 The Mortgage Class

Mortgage
<pre>-annualInterestRate: double -numOfYears: int -loanAmount: double</pre>
<pre>+Mortgage() +Mortgage(annualInterestRate: double, numOfYears: int, loanAmount: double) +getAnnualInterestRate(): double +getNumOfYears(): int +getLoanAmount(): double +setAnnualInterestRate(annualInterestRate: double): void +setNumOfYears(numOfYears: int): void +setLoanAmount(loanAmount: double): void +monthlyPayment(): double +totalPayment(): double</pre>

Mortgage

TestMortgageClass

Run

Example 6.8 The Count Class

Count
<div>-count: int</div> <div><u>-numOfCounts: int</u></div>
<div>+Count()</div> <div>+getCount(): int</div> <div>+setCount(count: int): void</div> <div><u>+getNumOfCounts(): int</u></div> <div>+clear(): void</div> <div>+increment(): void</div> <div>+decrement(): void</div>

Count

TestCount

Run



Java API and Core Java classes

- `java.lang`

Contains core Java classes, such as numeric classes, strings, and objects. This package is implicitly imported to every Java program.

- `java.awt`

Contains classes for graphics.

- `java.applet`

Contains classes for supporting applets.

Java API and Core Java classes, cont.

- `java.io`

Contains classes for input and output streams and files.

- `java.util`

Contains many utilities, such as date.

- `java.net`

Contains classes for supporting network communications.



Java API and Core Java classes, cont.

- `java.awt.image`

Contains classes for managing bitmap images.

- `java.awt.peer`

Platform-specific GUI implementation.

- Others:

`java.sql`

`java.rmi`