

CISC 3115 TY2

Methods and Objects

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Notice

- The slides are subject to change.

Outline

- Passing objects to methods
- Array of objects
- Scope of variables

Passing Objects to Methods

- Java passes arguments to methods by their values
- Primitive type argument
 - Passing by value for primitive type argument
- (Object) reference type
 - Passing by value for reference type argument
 - The value is the reference to the object

Passing Objects to Methods: Ex 1

```
public class TestCircle {  
    public static void main(String[] args) {  
        double small = 5.0, big = 25.0;  
        Circle c1 = new Circle (small);  
        printCircle(c1);  
        c1. setRadius(big);  
        printCircle(c1);  
    }  
}  
  
public static void printCircle(Circle c) {  
    System.out.println("The area of the circle of radius " + c.getRadius() + " is " + c.getArea());  
}
```

Passing Objects to Methods: Ex 2

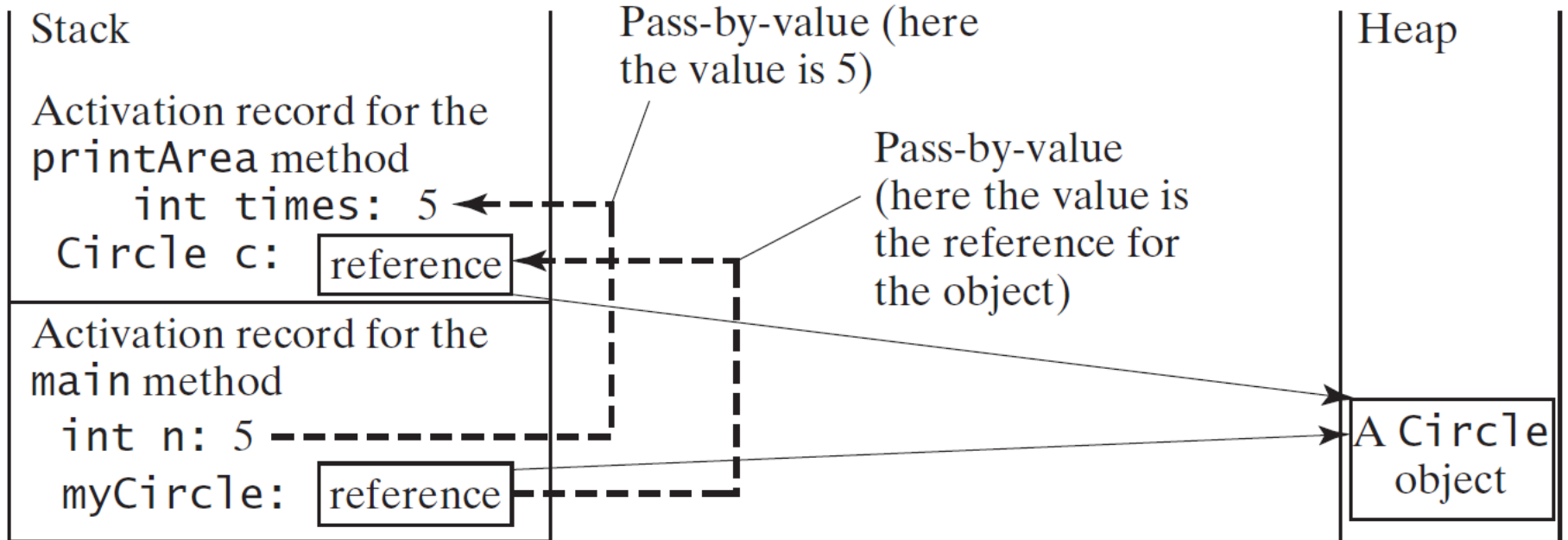
```
public class TestCircle {  
    public static void main(String[] args) {  
        Circle myCircle = new Circle (1.0);  
        int n = 5;  
        printCircle(myCircle, n);  
        printCircle(myCircle);  
    }  
}
```

```
public static void printCircle(Circle c) {  
    System.out.println("The area of the circle of  
radius " + c.getRadius() + " is " + c.getArea());  
}
```

```
public static void printCircle(Circle c, int times) {  
    System.out.println("Radius \t\t Area");  
    for (int i = 0; i < times; i++) {  
        System.out.println(c.getRadius() + "\t\t" +  
c.getArea());  
        c.getRadius(c.getRadius() + 1);  
    }  
}
```

Pass-by-Value

- Primitive type and reference type



Passing Objects to Methods: Ex 3: Side Effect?

```
public class TestCircle {  
    public static void main(String[] args) {  
        Circle myCircle = new Circle (1.0);  
        int n = 5;  
        printCircle(myCircle, n);  
        printCircle(myCircle);  
    }  
}  
  
public static void printCircle(Circle c) {  
    System.out.println("The area of the circle of  
radius " + c.getRadius() + " is " + c.getArea());  
}
```

```
public static void printCircle(Circle circle, int  
times) {  
    System.out.println("Radius \t\t Area");  
    Circle c = new Circle(circle);  
    for (int i = 0; i < times; i++) {  
        System.out.println(c.getRadius() + "\t\t" +  
c.getArea());  
        c.setRadius(c.getRadius() + 1);  
    }  
}
```


Questions?

- Pass parameters to methods
 - Primitive type values
 - Reference type values
 - Is there any side effects? Is the side effect desired or undesired?

Exercise (to be continued, 1 of 3)

- In this exercise, you are to revise previous Java classes you wrote.
 - Make a directory for this exercise (e.g., ex02 or CircleUtils), work on the directory for this exercise
 - In the end, your program consists of Circle.java, CircleUtils.java, and TestCircle.java (or CircleClient.java).
 - You will create CircleUtils.java, and revise TestCircle.java (or CircleClient.java)
 - See next two slides for how you should write or revise these two classes

Exercise (continued, 2 of 3)

- The CircleUtils class (in file CircleUtils.java) that consists of 6 methods
 - Move the two printCircle(...) methods from TestCircle.java to CircleUtils.java, and make them *instance methods*. These two methods are discussed in this lecture
 - Add four new methods to the CircleUtils class with the following signatures
 - `public void doubleRadius(Circle c):` double c's radius
 - `public void doubleArea(Circle c):` double c's area by setting c's radius to an appropriate value
 - `public Circle getNewCircleDoubleRadius(Circle c):` returns a Circle object whose radius is twice of c's, but without changing c
 - `public Circle getNewCircleDoubleArea(Circle c):` returns a Circle object whose area is twice of c's, but without changing c

Exercise (continued, 3 of 3)

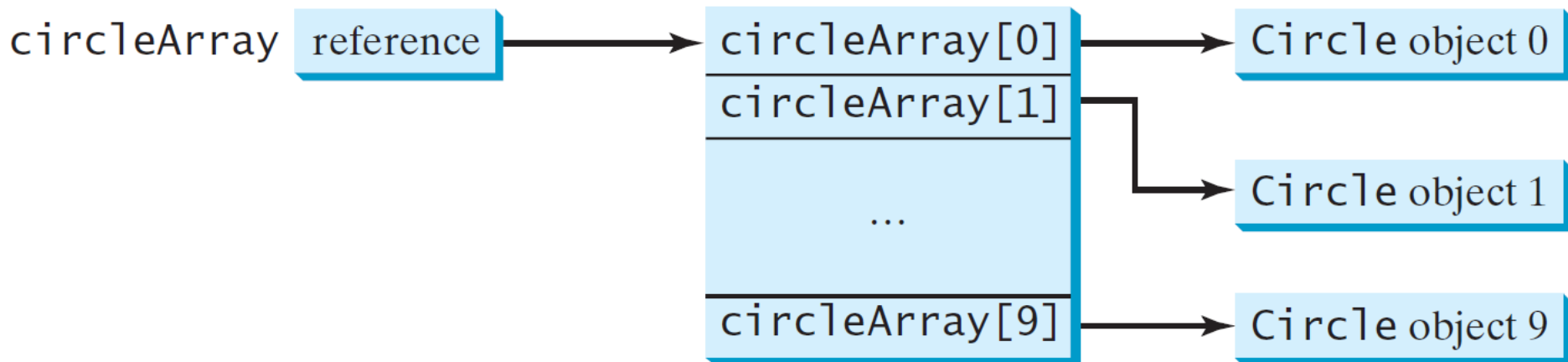
- Revise TestCircle.java
 - Remove the two printCircle(...) methods (you have done this in previous slide if you “moved” instead of “copying” them)
 - Revise the main methods in TestCircle.java (or CircleClient.java) to demonstrate the 6 methods in the CircleUtils class
- Submit the work as a journal entry

Array and Array of Objects

- What is an array? How do we create an array in Java?
- Create an array of objects
 - `Circle[] circleArray = new Circle[10];`
- An array of objects is actually an array of reference variables.
 - So invoking `circleArray[1].getArea()` involves two levels of referencing.
 - `circleArray` references to the entire array.
 - `circleArray[1]` references to a `Circle` object.

Two-level Referencing

- An array is in effect an object
- `Circle[] circleArray = new Circle[10];`



Array of Objects: Example

- Compute areas of an array of Circles
 - Create an array of Circle objects
 - Print the array in a method to which the array is passed as an argument.
 - See it in the [sample program](#).
 - Formatted output (see [here](#) and [here](#))
 - Array length

%	1\$	+0	20	.10	f
Begin Format Specifier	Argument Index	Flags	Width	Precision	Conversion

Questions?

- What is an array?
- How do we create an array?
- How do we create an array of objects?
- How do we pass an array of objects to a method
- How do we get the length of an array?
- How do we reference to each object referenced by an array of objects?

Scope of Variables

- Scope: the part of the program where the variable can be referenced.
- Instance and static variables
 - The scope is the entire class. They can be declared anywhere inside a class.
 - There is one exception: when a data field is initialized based on a reference to another data field
- Local variable
 - The scope 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.

Scope of Variable: Example 1

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

```
public class F {  
    private int i;  
    private int j = i + 1;  
}
```

Scope of Variable: Example 2

```
public class F {  
    private int x = 0;  
    private int y = 0;  
  
    public void p() {  
        int x = 1;  
  
        System.out.println("x = " + x);  
  
        System.out.println("y = " + y);  
    }  
}
```

Two variables

Scope of Variable: Example 3

- What is the output?

```
public class TestScope {  
    public static void main(String[] args) {  
        int i = 2; int k = 3;  
        {  
            int j = 3;  
            System.out.println("i + j is " + i + j);  
        }  
        k = i + j;  
        System.out.println("k is " + k);  
        System.out.println("j is " + j);  
    }  
}
```

```
private static int i = 0;  
private static int j = 0;  
}
```

Questions?

- Concept of scope of variables
- Scope of local, instance, and static variables