CISC 3115 TY3
C06b: Methods and Objects

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Outline

- Passing objects to methods
- Array of objects
- Scope of variables
- Time permitting
  - The this reference variable
  - Immutable class and object

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: Example 1

```
public class TestCircle {

    pubic 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: Example 2

```
public class TestCircle {

    pubic 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

Stack

Activation record for the
`printArea` method
  `int times: 5`
`Circle c:` [reference]

Pass-by-value (here
the value is 5)

Pass-by-value
(here the value is
the reference for
the object)

Activation record for the
`main` method
  `int n: 5`
`myCircle:` [reference]

Heap
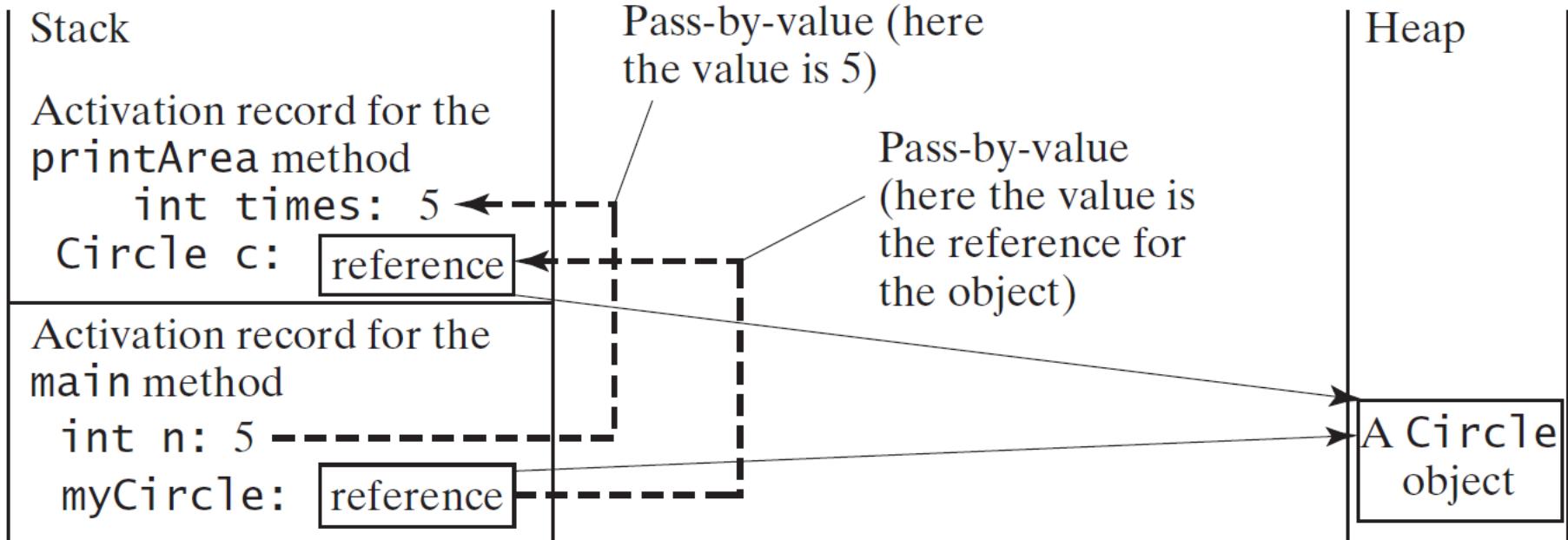
`A Circle`
`object`

Passing Objects to Methods: Example 3: No Side Effect

```java
public class TestCircle {

    pubic 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());

}
```

```java
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.getRadius(c.getRadius() + 1);

    }

}
```

Questions?

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

In-Class Exercise C06b-1: 1/3

- In this exercise, you are to revise the example program.
  - Create a C06b-1 directory in the directory that hosts your weekly programming repository, work at the directory
    - In the end, your program consists of Circle.java, CircleUtils.java, and TestCircle.java.
      - Copy them from the [sample program](#).
    - You will create CircleUtils.java, and revise TestCircle.java
      - See next two slides for details
    - Test the revised program
  - Use git to make a submission

In-Class Exercise C06b-1: 2/3

- CircleUtils.java has a CircleUtils class that consists of 6 methods
    - Copy the two printCircle(…) methods from TestCircle.java to CircleUtils.java, and make them instance methods
    - Add two methods to the CircleUtils class with the following signatures
        - public void doubleRadius(Circle c): double c's readius
        - 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

In-Class Exercise C06b-1: 3/3

- TestCircle.java
  - Remove the two printCircle(…) methods
  - Revise the main methods to demonstrate/test the 6 methods in the CircleUtils class

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];

CUNY | Brooklyn College
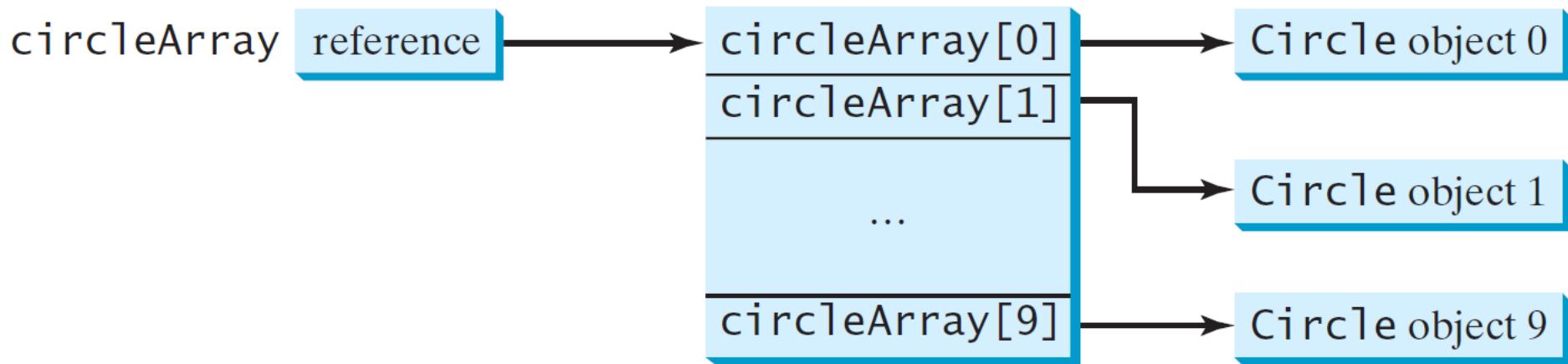
Array of Objects: Example

- 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
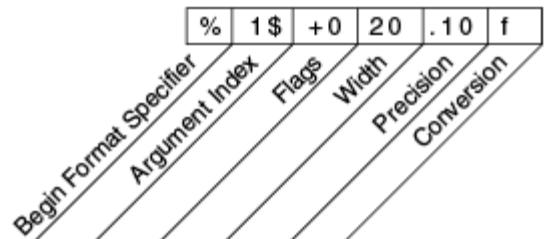
Questions?

- 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

```java
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

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

        System.out.println("y = " + y);
    }
}
```

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.printlln("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