CISC 3115

Design Methods with Objects

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Outline

- Design Methods with Objects
  - Passing objects to methods
  - Returning objects
- 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 {
   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: Ex 2

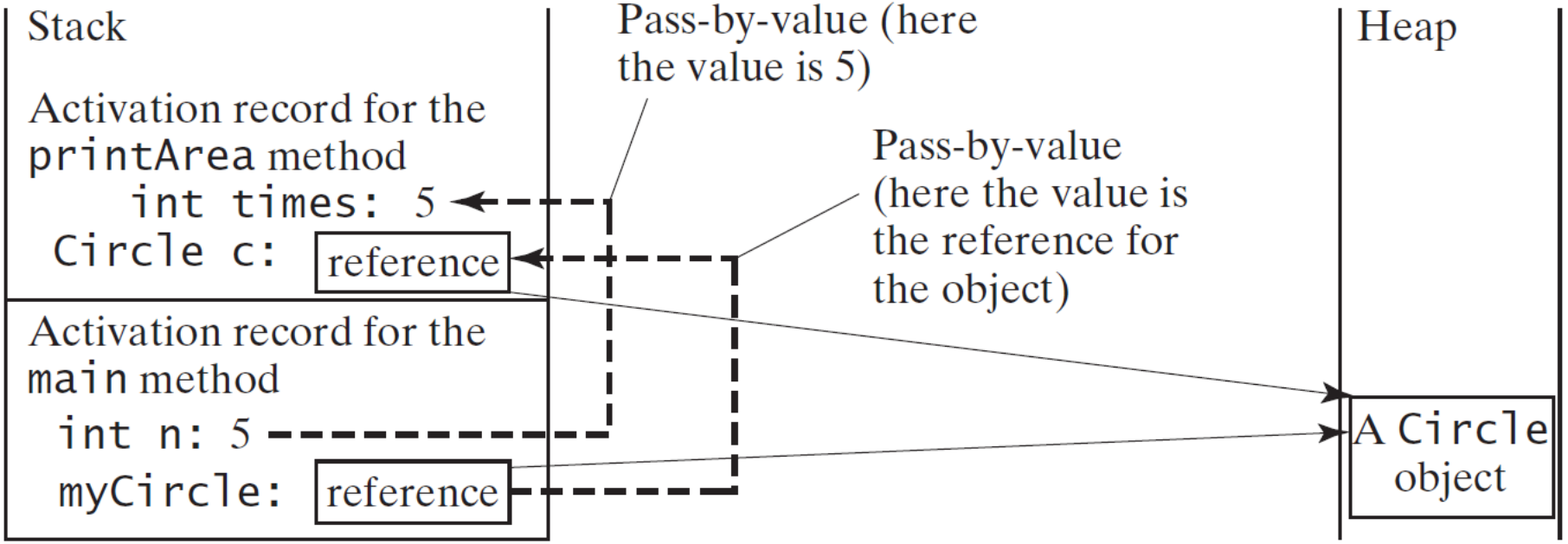
```
public class TestCircle {
  pubic static void main(String[] args) {
        Circle myCircle = new Circle (1.0);
        int n = 5;
        printCircle(myCircle, n);
        printCircle(myCircle);
  }
}
```

```
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



Passing Objects to Methods: Ex 3: Side Effect?

```
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 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.
  - Suggestion: make a directory for this exercise to organize your work
  - 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
    - In the end, your program should consist of three classes, Circle.java, CircleUtils.java, and TestCircle.java (or CircleClient.java).

Exercise (continued, 2 of 3)

- The CircleUtils class (in file CircleUtils.java) has 6 methods
  - Move the two printCircle(…) methods from TestCircle.java/CircleClient.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
    - public void doubleRadius(Circle c)
      - It doubles c's radius
    - public void doubleArea(Circle c)
      - It doubles c's area by setting c's radius to an appropriate value
    - public Circle getNewCircleDoubleRadius(Circle c)
      - It returns a Circle object whose radius is twice of c's, but without changing c
    - public Circle getNewCircleDoubleArea(Circle c)
      - It returns a Circle object whose area is twice of c's, but without changing c

Exercise (continued, 3 of 3)

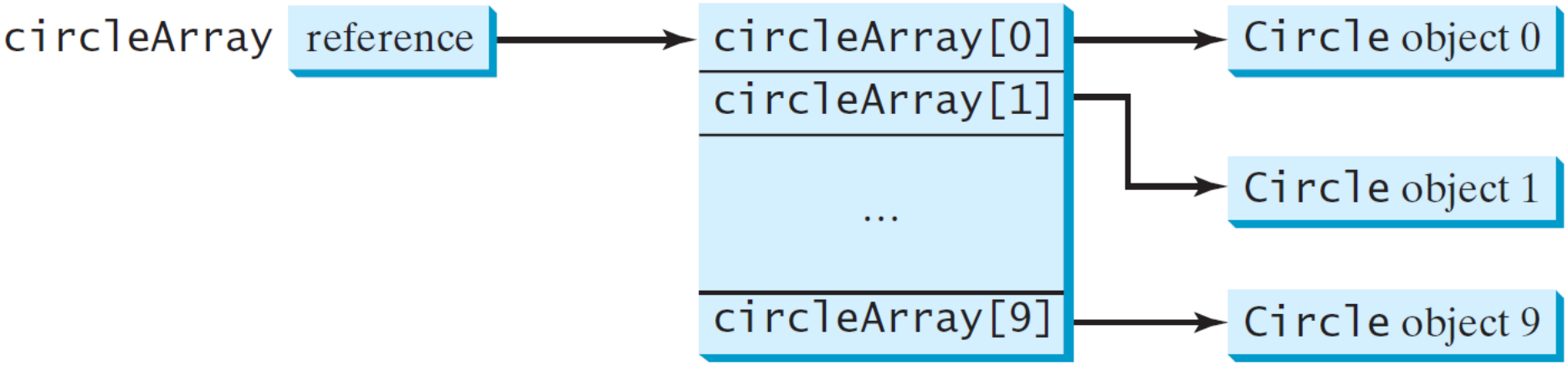
- Revise TestCircle.java/CircleClient.java
  - Revise the main methods in TestCircle.java/ CircleClient.java to demonstrate the 6 methods in the CircleUtils class, i.e., write necessary code to invoke the 6 methods in the previous slide

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.

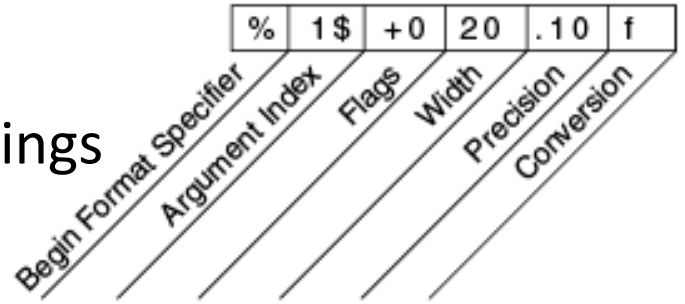
Two-level Referencing

- 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.
  - Format the output using format strings
    - Hint: using Java format strings
    - String.format("%+020.10f %+020.10f", Math.PI, -Math.E)
    - String.format("%2$+020.10f %1$+020.10f", Math.PI, -Math.E)



```
public class Circle {
  private double radius = 1.0;

  public Circle() {
  }

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

  public Circle(Circle c) {
    radius = c.radius;
  }

  public double getRadius() {
    return radius;
  }

  public void setRadius(double newRadius) {
    radius = newRadius;
  }

  public double getArea() {
    return radius * radius * Math.PI;
  }
}
```

```
public class TotalArea {
  public static void main(String[] args) {
    Circle[] circleArray;
    circleArray = createCircleArray(5);
    printCircleArray(circleArray);
  }

  public static Circle[] createCircleArray(int n) {
    Circle[] circleArray = new Circle[n];
    for (int i=0; i<n; i++) {
      circleArray[i] = new Circle(Math.random() * 100);
    }
    return circleArray;
  }

  public static void printCircleArray(Circle[] circleArray) {
    System.out.printf("%-30s%-15s\n", "Radius", "Area");
    for (int i=0; i<circleArray.length; i++) {
      System.out.printf("%-30f%-15f\n", circleArray[i].getRadius(), circleArray[i].getArea());
    }
    System.out.printf("%-30s%-15f\n", "The total area of the circles is ", sumArea(circleArray));
  }

  public static double sumArea(Circle[] circleArray) {
    double sum = 0;
    for (int i=0; i<circleArray.length; i++) {
      sum += circleArray[i].getArea();
    }
    return sum;
  }
}
```

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

what is the scope of each variable?

```
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;

  public void p() {
    int x = 1;
    System.out.println("x = " + x);

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

Are these x's two variables or a single variable?

Scope of Variable: Example 3

What is the output of the program?

```
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