Lecture 26:

Reference types
Passing objects to methods
Array of objects

Apr 3 2015
Variables with primitive types vs. Variables with object types

<table>
<thead>
<tr>
<th>Primitive type</th>
<th>int i = 1</th>
<th>i</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object type</td>
<td>Circle c</td>
<td>c</td>
<td>reference</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>c: Circle</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>radius = 1</td>
</tr>
</tbody>
</table>

Created using new Circle()
Copying variables with primitive types vs.
Copying variables with object types

**Primitive type assignment** \( i = j \)

<table>
<thead>
<tr>
<th>Before:</th>
<th>After:</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i )</td>
<td>( i )</td>
</tr>
<tr>
<td>( j )</td>
<td>( j )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Before:</th>
<th>After:</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c1 )</td>
<td>( c1 )</td>
</tr>
<tr>
<td>( c2 )</td>
<td>( c2 )</td>
</tr>
</tbody>
</table>

**Object type assignment** \( c1 = c2 \)

Before:

- \( c1: \text{Circle} \), radius = 5
- \( c2: \text{Circle} \), radius = 9

After:

- \( c1: \text{Circle} \), radius = 5
- \( c2: \text{Circle} \), radius = 9

The diagram shows that when copying variables with primitive types, the values are copied directly, resulting in the same values for both variables. However, when copying variables with object types, the objects themselves are not copied, and the values are not modified accordingly.
Garbage Collection

• As shown in the previous figure, after the assignment statement \( c_1 \ = \ c_2 \), \( c_1 \) points to the same object referenced by \( c_2 \).

• The object previously referenced by \( c_1 \) is no longer referenced. This object is known as garbage.

• Garbage is automatically collected by JVM.
Passing objects to methods

Passing an object to a method is to pass the reference (a reference to the object) to the parameter variable of the method

- Passing by value for primitive type value
  - the value is passed to the parameter

- Passing by value for reference type value
  - the value is the reference to the object

(This is the exact same idea as when we pass an array to a method!)
Passing objects to methods, cont.

Stack

Space required for the printAreas method
int times: 5
Circle c: reference

Space required for the main method
int n: 5
myCircle: reference

Pass by value (here the value is 5)
Pass by value (here the value is the reference for the object)

Heap

A circle object
public class TestPassObj{
    public static void main(String[] args){
        CirclePrivate c1 = new CirclePrivate();
        int n = 5;
        printArea(c1, n);
        System.out.println("\n Radius = " + c1.getRadius() + " no of Circle obj = " + c1.getNumObj() + " n = " + n);
    }
    public static void printArea(CirclePrivate cir, int times){
        while(times >= 3){
            System.out.println(cir.getRadius() );
            cir.setRadius(cir.getRadius() + 1);
            times--; }
    }
}

1.0 2.0 3.0
Radius = 4 no of Circle obj = 1 n = 5
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 as shown in the next figure:

– circleArray references to the entire array
– circleArray[1] references to a Circle object
Array of objects continued
Let’s see all this in action…

See

• Card.java
• TestCard1.java
• DeckOfCards.java
• TestDeck.java

This example is explained in much detail on
http://www.mathcs.emory.edu/~cheung/Courses/170/Syllabus/syl.html

Under “Data Abstraction”