Instance Variables

Jonathan Wishcamper

Assignment Definition and General Feedback by Michael Panitz
at Cascadia Community College (http://www.cascadia.edu)

Table of contents:

Quick Summary:

In Java, you can create instance variables, which unlike regular local variables, are able to be used outside of a single method in a class. This allows you to store data in a variable that can be accessed by multiple different methods in an object. While local variables will "disappear" after the method is complete, instance variables will keep their value for the duration of the class that they are in.

When To Use This / Avoid This:

There are several situations where you will want to use this technique. It is good to use if you want a variable that can be accessed throughout the life of your object – for example, if you want to keep track of the robots movements.

In addition, there are several situations where you will want to avoid this technique. If you do not use the variable outside of a single method, it is better to simply declare the variable within the class. This can prevent compile errors if you attempt to use the same variable name in a method without meaning to, and provides better general cleanliness in your program.

Example of Usage:

Let's say you want to make a robot that will count the number of spaces moved, and be able to print that value at any time.

There are three steps:

First, create a new type of robot (in this example, SuperRobot). To do this, put a class declaration above your main class: class SuperRobot extends Robot

Second, declare a variable outside of any methods (in this example, private int spacesMoved = 0;).

Finally, use this variable in any method that you wish within this class. In this example, it is used in the public void move(int n) and public void printMoved() methods, on lines 13 and 19.

Don't pay too much attention to the move(int n) command, just know that it will move the robot however many spaces you specify in the code. Pay more attention to the use of the instance variable spacesMoved.

Line	Code
1.	`import becker.robots.*;`
2.	`class SuperRobot extends Robot`
3.	`{`
4.	`private int spacesMoved = 0;`
5.	
6.	`public SuperRobot(City c, int st, int ave, Direction dir, int num) {`
7.	`super(c, st, ave, dir, num); }`
8.	
9.	`public void move(int n) {`
10.	`while(n > 0) {`
11.	`if(this.frontIsClear()) {`
12.	`this.move();`
13.	`spacesMoved++;`
14.	`n--; }`
15.	`else {`
16.	`n=0; } } }`
17.	
18.	`public void printMoved() {`
19.	`System.out.println("I have moved " + spacesMoved + " Spaces!"); }`
20.	`}`
21.	`public class Instance_Variables_Example extends Object`
22.	`{`
23.	`public static void main(String[] args)`

24.	{
25.	City bothell = new City();
26.	SuperRobot Jon = new SuperRobot(bothell, 1, 1, Direction.EAST, 0);
27.	Jon.move(1);
28.	Jon.printMoved();
29.	Jon.move(4);
30.	Jon.printMoved();
31.	}
32.	}

Example Trace:

In order to go over these details more thoroughly, here is a (partial) trace of the above program, with some additional explanation afterwards:

Line #	Program Statement	Jon St #	Jon Ave #	spacesMoved value	True/ False	n Value	Output
23	public static void main(String[] args)	-	-	-	-	-	
26	SuperRobot Jon = new SuperRobot(bothell, 1, 1, Direction.EAST, 0);	1	1	0	-	-	
27	Jon.move(1);	1	1	0	-	-	
9	public void move(int n) {	1	1	0	-	1	
10	while(n > 0) {	1	1	0	True	1	
11	if(this.frontIsClear()) {	1	1	0	True	1	
12	this.move();	1	2	0	-	1	
13	spacesMoved++;	1	2	1	-	1	
14	n--; }	1	2	1	-	0	
15	else {	1	2	1	False	0	
10	while(n > 0) {	1	2	1	False	0	
28	Jon.printMoved();	1	2	1	-	-	
18	public void printMoved() {	1	2	1	-	-	
19	System.out.println("I have moved " +	1	2	1	-	-	I have moved 1 Spaces!

	spacesMoved + " Spaces!"); }						
29	Jon.move(4);	1	2	1	-	-	
-	… Refer to lines 9-15 in the trace	1	6	5	-	-	
30	Jon.printMoved();	1	6	5	-	-	I have moved 5 Spaces!

You'll notice that the trace starts at beginning of the **main** function, on line 23, like normal. It proceeds normally until line 30, although it's worth noting that line 26 creates a SuperRobot, instead of the normal Robot. The Instance Variable (spacesMoved) is now 0, because it is created as soon as the robot is. As the program runs, spacesMoved gets greater by 1 each time the robot moves, because of line 13. This allows the program to print out the spaces moved at any time.

Syntax Explanation:

It is easy to place the instance variable in the wrong place, which can be disastrous. In the following example, spacesMoved would be reset every time move is called, which would defeat the purpose.

Let's start with the program as it's written here.

Line #	Program Source Code
1.	`import becker.robots.*;`
2.	`class SuperRobot extends Robot`
3.	`{`
4.	`public SuperRobot(City c, int st, int ave, Direction dir, int num) {`
5.	`super(c, st, ave, dir, num); }`
6.	`public void move(int n) {`
7.	`private int spacesMoved = 0;`
8.	`while(n > 0) {`
9.	`if(this.frontIsClear()) {`
10.	`this.move();`
11.	`spacesMoved++;`
12.	`n--; }`
13.	`else {`
14.	`n=0; } } }`

Below, you can see the finished program with the differences highlighted in yellow, so it's easy to see what's been added/changed.

Line #	Program Source Code

```
1.  import becker.robots.*;
2.  class SuperRobot extends Robot
3.  {
4.  private int spacesMoved = 0;
5.  public SuperRobot(City c, int st, int ave, Direction
    dir, int num) {
6.  super(c, st, ave, dir, num); }
7.  public void move(int n) {
8.  while(n > 0) {
9.  if(this.frontIsClear()) {
10. this.move();
11. spacesMoved++;
12. n--; }
13. else {
14. n=0; } } }
```

Help With The Logic:

This is good to use whenever you want to keep track of a number, string, or other variable within an object, and you want to be able to use it within more than one of the methods. The variable is created outside of the methods and is therefore able to be used or changed by every method in the object.

Important Details:

- While it is not required, declaring the variable private helps because this way it is only able to be called within the specific class (SuperRobot, for example). This way it cannot be accidentally called by main or another class you happen to have within the same program. Failing to declare the variable private can lead to confusing and difficult to fix intent errors.

Mistakes People Commonly Make (And How to Fix Them):

Quick Name of Mistake: Placing variable in the wrong place
Detailed Example of Error: Variable placed inside a method instead of outside. See syntax explanation above.
Detailed Example of Fix: See syntax explanation above.

Hyperlinks

Licensing

Plagiarism

If you believe that some or all of this document infringes on your intellectual property (i.e., part or all of this document is copied from something you've written) please immediately contact Mike Panitz at Cascadia Community College (perhaps using the Faculty And Staff Directory at http://www.cascadia.edu/pages/searchtemplate.aspx)