 

**Advanced Placement**

**Computer Science**

[**Shenendehowa HS**](http://www.shenet.org/shen-high-school/)[**mr Hanley**](http://mrhanleyc.com/)

**Unit 3: Primitives, References & Objects**

**Lesson: Parameter Passing\_FILLEDOUT**

***Last Updated:*** *10/17/2017*

Lesson: Parameter Passing Mechanisms

*Last Updated: 100/11001/1100*

Parameter Passing :

Primitives and References are passed by Value

This means that the copy of the value is passed from the caller to the parameter

**Example 1: passing a primitive:**

public class Example1 {

public static double findAvg(double a, double b, double c) {

double avg = (a + b + c) /3;

a = a + 1; //attempt at a side effect

return avg;

}

public static void main(String[] args) {

double n1 = 10, n2 = 20, n3 = 32;

double result = findAvg(n1, n2, n3);

System.out.println(“For the #’s” + n1 + “\t” + n2 + “\t” + n3 + “\t” + “Average = “ + result);

}

}

In the example above, the values 10, 20 and 32 are passed into the parameters a, b and c.

The pass by value mechanism means that a gets 10, b gets 20 and gets the value 32.

The average is calculated and then we change a.

Does this create a side effect?No because a now has nothing to do with the variable in the main n1.

Therefore the output is :

For the #’s 10 20 32 Average = 20.6666667

**Example 2: passing a reference:**

public class Student { //constructor not shown

public String name;

public double test1, test2, test3;

}

public class Example2 {

public static double findStuAvg(Student s) {

double avg = (s.test1+s.test2+s.test3)/3;

s.test1 -= 10; //attempt at a side effect

return avg;

}

public static void main(String[] args) {

Student s1 = new Student (“Johnny Rocket” , 100, 87, 91);

double result = findStuAvg(s1);

System.out.println(“S1” + s1.test1 + “\t” + s1.test2 + “\t” + s1.test3 + “\t” + “Average = “ + result);

}

}

References are variables that contain the address of other variables.

Objects of the Student class are controlled by reference variables.

**1290**

In this case,

**Johnny Rocket**

**100**

**87**

**91**

s1

**1290**

As you can see from the above example, the reference s1 “points” to the object at memory address 1290, which is the Student object containing Johnny Rocket and the 3 grades and other variables not shown.(1290 is purely arbitrary and will change each time you run the program)

When we pass s1 to the method findStuAvg, java uses pass by value. This means that the memory address 1290 gets past to the reference variable s.

Thus we have

**1290**

**Johnny Rocket**

**100**

**87**

**91**

**Johnny Rocket**

90

**87**

**91**

s1

**1290**

**1290**

s

Now that we passed the memory address of the object, both s1 and s are aliases for the object containing Johnny Rocket. They are two different pathways to the same object.

So after finding the average, then we subtract 10 from s.test1. This will CHANGE the object so test1 is 10 points lower. This is known as a side effect since we are altering a variable that was passed to the method.

Thus the output is

s1 100 s2 87 s3 91 Average = 92.66667

Yet the test1 after the method returns is actually 90

Side Effects can be helpful if you want to return multiple values from a method