I don’t feel comfortable doing video. I want to do audio, but my computer doesn’t come with built in microphone. I am happy to do audio on school computer that comes with microphone.

Level 3: Comparing Strings-Lesson 9

There are many cases you need to compare string to each other. Using simple equality operator as you do with primitive don’t work the way you expect. I declare two variables.

String s1=” Hello”

String s2= “Hello”

They both have the same value, string “Hello”

I will add simple conditional code. Now I will compare using double equal or equality operator. It works great with primitive numbers, bool, and character. With string it might appears that its woks also.

If(s1==s3) {

System.out.println(“They match”)

} else {

System.out.println (“They don’t match”

}

I got back they match. After all the two strings are identical. But the reason for that match isn’t what you think they are.

Let’s declare another string s3=” hello”. Now compare it with s1 and s3.

If(s1==s3) {

System.out.println(“They match”)

} else {

System.out.println (“They don’t match”

}

Clearly don’t match because one is upper case(s1), and one is lower case(s3). Now try this. I declare string variable named p1 and I give it a value of hello followed by space character. Then I declare variable named p2 and I will give it a value of world.

P1=” Hello “

P2=” World”

Then I create a variable p4 and I create that by concatenating p1 plus p2. Now I have string of hello world.

P4=p1+p2

Now I create another string p5. This would be an entire string “Hello world “. Now those two strings p4 and p5 should match. Now when I executed conditional logic,

If(p4==p5) {

System.out.println(“They match”)

} else {

System.out.println (“They don’t match”)

}

Now I got they don’t match. What is going here?

In the first example when I declared s1, it is registered in memory by the compiler. Then when I declared s2, instead of creating a new object, the compiler looked into the existing string and found a match. Instead of creating another object, it points to the second reference variable s2 to the original object s1. This is ok because strings are immutable, once string object is created, it’s value can’t be changed. It is safe, to point the second string s2 to the first string s1. After all they have same values.

When comparing s1 with s3, the compiler didn’t find the match, so it created new object and the entire string in s3 is lower case. So, it return “They don’t match”.

Now we come to last example, p1 and p2. They are their own object. The compiler asked to put those values together. So p4 is created as another object and p5 is completely different object. The complier doesn’t know p4 match p5. When it tries to match values, it returns “They don’t match”. To accurately compare string in java, we should use string classes equals or equalsignorecase methods. Let’s call equals method first.

If (p4. Equals(p5)) {

System.out.println(“They match”)

} else {

System.out.println (“They don’t match”)

}

This time, it returns “They match”. The lesson here that is when we want to compare strings to each other, to always get accurate result use equals or equalsignorecase methods. Don’t use == operator. The == operator compare reference not content but equals or equalsignorecase methods compare the contents of the string does not compare reference. So, always use equals or equalsignorecase methods when comparing strings.

Level 3: Triangle-Lesson 6

In this lesson I need to show the sum of two sides of a triangle is always greater than the third side. Now consider a right-angle triangle with sidea, sideb and sidec with dimension of 3,4, and 5 respectively.

1. sideA + sideB > sideC (3+4 = 7 >5)
2. sideA + sideC > sideB (3+5 = 8 >4)
3. sideB + sideC > sideA(4+5 = 9 >3)

For any combination, the sum of two sides of a triangle is always greater than the 3rd sides.

public static void main (String[] args) {  
 *//write your code here* Scanner input = new Scanner(System.*in*);  
 int sideA = input.nextInt();  
 int sideB = input.nextInt();  
 int sideC = input.nextInt();  
 sideA=3;  
 sideB=4;  
 sideC=5;

*// sideA, SideB and SideC are passed as arguments to isTriangle method which has*

*// parameters sideA, sideB and sideC.*

*// It returns true if the triangle exists, false if it doesn't.* if (*isTriangle*(sideA, sideB, sideC)) {  
 System.*out*.println(*TRIANGLE\_EXISTS*);  
 } else {  
 System.*out*.println(*TRIANGLE\_DOES\_NOT\_EXIST*);  
 }  
}

public static boolean isTriangle(int sideA, int sideB, int sideC) {  
 return sideA + sideB > sideC && sideA + sideC > sideB && sideB + sideC > sideA;  
}

In all three cases, the isTriangle method return “*TRIANGLE\_EXISTS*” for right angle triangle. But if (sideA=1, sideB=1 and sideC=3). Because any combination of the sum of two sides of a triangle is not always greater than the 3rd sides. It should return “*TRIANGLE\_DOES\_NOT\_EXIST.*” Because this is not a valid triangle and it failed in the first case.

Case 1: sideA + sideB > sideC (1+1 = 2 >3) // not a valid triangle

Case2: sideA + sideC > sideB (1+3 = 4 >1)

Case3: sideB + sideC > sideA(1+3 = 4 >1)