COMPSCI 121: DATA TYPES & BRANCHING

SPRING 20

GOALS FOR TODAY'S CLASS

You are now familiar with Objects and Classes and writing programs to solve a problem.
We now work with more basic "building blocks" of the Java programming language:

- Review conditional statements
- The Random method
- Characters & Strings

REVIEW: CONDITIONAL *IF* STATEMENTS

```
if (alpha > beta)
{
    eta = alpha + 2;
    gamma = alpha + 5;
}
```

```
else
{
    eta = alpha - 1;
    gamma = beta - 1;
}
```

```
int alpha = 2, beta = 1,
delta = 3, eta = 0, gamma = 0;
```

First evaluate `(alpha > beta)`:
(2 > 1) TRUE

If true – follow true branch, if false – do nothing or follow else branch

The condition is TRUE so we execute the true branch:

`eta = alpha + 2 = 2 + 2 = 4`

`gamma = alpha + 5 = 2 + 5 = 7`

Think - Pair -Share

```
int alpha = 2;
int beta = 1,
int delta = 3,
int eta = 0,
int gamma = 0;
```

Evaluate these statements and determine the value of all variables used.

```
if (alpha > beta) {
    eta = alpha + 2;
    gamma = alpha + 5;
}
else {
    eta = alpha -1;
    gamma = beta - 1;
}

if (alpha > delta)
    gamma = alpha + 5;
else
    gamma = beta + 5;
eta = beta + 2;
```

EXPLANATION

For *if* statement, determine whether true or false

First we evaluate `(alpha > delta)`:

(2 > 3) FALSE

If true – follow true branch, if false – do nothing or follow else branch

The condition is FALSE so we follow the else branch.

`gamma = beta + 5 = 1 + 5 = 6`

Next sequential statement is always executed:

`eta = beta + 2 = 1 + 2 = 3`

```
if (alpha > beta)
{
    eta = alpha + 2;
    gamma = alpha + 5;
}
else
{
    eta = alpha - 1;
    gamma = beta - 1;
}
```

```
if (alpha > delta)
    gamma = alpha + 5;
else
    gamma = beta + 5;
```

```
eta = beta + 2;
```

EXPLANATION

For *if* statement, determine whether true or false

First we evaluate `(alpha > delta)`:

(2 > 3) FALSE

If true – follow true branch, if false – do nothing or follow else branch

The condition is FALSE so we follow the else branch.

`gamma = beta + 5 = 1 + 5 = 6`

Next sequential statement is always executed:

`eta = beta + 2 = 1 + 2 = 3`

```
if (alpha > beta)
{
    eta = alpha + 2;
    gamma = alpha + 5;
}
else
{
    eta = alpha - 1;
    gamma = beta - 1;
}
if (alpha > delta)
    gamma = alpha + 5;
else
    gamma = beta + 5;
eta = beta + 2;
```

```
int alpha = 2;
int beta = 1,
int delta = 3,
int eta = 0,
int gamma = 0;
```

alpha = 2 : int

beta = 1 : int

delta = 3 : int

eta = 3 : int

gamma = 6 : int

CLICKER QUESTION 1

```
if (omega > kappa)
{
    if (alpha > delta)
        eta = 5;
    else
        eta = 4;
}
else {
    if (alpha < delta)
        eta = 3;
    else
        eta = 2;
}
```

Evaluate the statements and determine the value of `eta` given:

```
int alpha = 2, delta = 3,
eta = 0;
double omega = 2.5, kappa
= 3.0;
```

A. 2

B. 3

C. 4

D. 5

CLICKER QUESTION 1 ANSWER

```
if (omega > kappa)
{
    if (alpha > delta)
        eta = 5;
    else
        eta = 4;
}
else
    if (alpha < delta)
        eta = 3;
    else
        eta = 2;
```

FALSE

TRUE

Evaluate the statements and determine the value of `eta` given:

```
int alpha = 2, delta = 3,
eta = 0;
double omega = 2.5, kappa =
3.0;
```

A. 2

B. 3

C. 4

D. 5

alpha = 2 : int
delta = 3 : int
eta = 3 : int
omega = 2.5 : double
kappa = 3.0 : double

THE RANDOM CLASS

To generate random numbers- e.g., to simulate real-world situation, gaming, etc. see `DiceRoll.java`

```
import java.util.Scanner;
import java.util.Random;

public class DiceRoll {
   public static void main (String [] args) {

      Random randGen = new Random();
      System.out.println(randGen.nextInt());
```

Import statement

New instance

call `nextInt()` method

```
 ----jGRASP exec: java DiceRoll
1531554058

 ----jGRASP: operation complete.

 ----jGRASP exec: java DiceRoll
-2143417819
```

Returns a random number

THE RANDOM CLASS

To generate random numbers- e.g., to simulate real-world situation, gaming, etc. see `DiceRoll.java`

```
import java.util.Random;

public class DiceRoll {
   public static void main (String [] args) {

      Random randGen = new Random();
      System.out.println(randGen.nextInt(10));
      System.out.println(randGen.nextInt(10));
      System.out.println(randGen.nextInt(10));
```

Returns 10 possible values from 0 to 9

OTHER USES OF nextInt METHOD

```
System.out.println(randGen.nextInt(6)+ 10);
```

Generates any 6 values starting at 10.

```
Random randGen = new Random();
System.out.println(randGen.nextInt(6)+ 10);
System.out.println(randGen.nextInt(6)+ 10);
System.out.println(randGen.nextInt(6)+ 10);
System.out.println(randGen.nextInt(6)+ 10);
System.out.println(randGen.nextInt(6)+ 10);
System.out.println(randGen.nextInt(6)+ 10);
```

12	11	12
12	12	15
14	12	13
10	14	12
12	12	13
12	15	10

OTHER USES OF nextInt METHOD

```
Random randGen = new Random();
Scanner scnr = new Scanner(System.in);
int seedVal;

System.out.print("Enter seed value: ");
seedVal = scnr.nextInt();
randGen.setSeed(seedVal);
System.out.println(randGen.nextInt(10));
System.out.println(randGen.nextInt(10));
```

declare int variable

Set the seed

With a specific seed, each program run will yield the same sequence of pseudo-random numbers.

```
Enter seed value: 5
7
2
```

```
Enter seed value: 5
7
2
```

CLICKER QUESTION #2

Which of the following will work like flipping a coin and getting us 0 and 1 (heads/tail)?

A. `randGen.nextInt(0);`

B. `randGen.nextInt(1);`

C. `randGen.nextInt(2);`

D. `randGen.nextInt(3);`

CLICKER QUESTION #2 ANSWER

Which of the following will work like flipping a coin?

A. `randGen.nextInt(0);`

Error: bound must be positive

B. `randGen.nextInt(1);`

Gives 0;

C. `randGen.nextInt(2);`

Gives 0/1 (heads/tails)

D. `randGen.nextInt(3);`

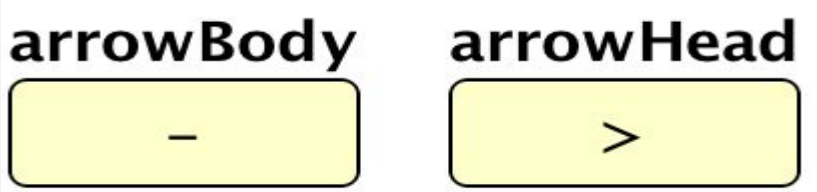
Gives 0, 1, or 2

CHARACTER DATA TYPE

```java
public class CharArrow {
   public static void main (String [] args) {
      char arrowBody;
      char arrowHead;

      arrowBody = '-';
      arrowHead = '>';

      System.out.println(arrowHead);
      System.out.println("" + arrowBody + arrowBody + arrowBody + arrowHead)

      arrowBody = 'o';

      System.out.println("" + arrowBody + arrowBody + arrowBody + arrowHead)
   }
}
```

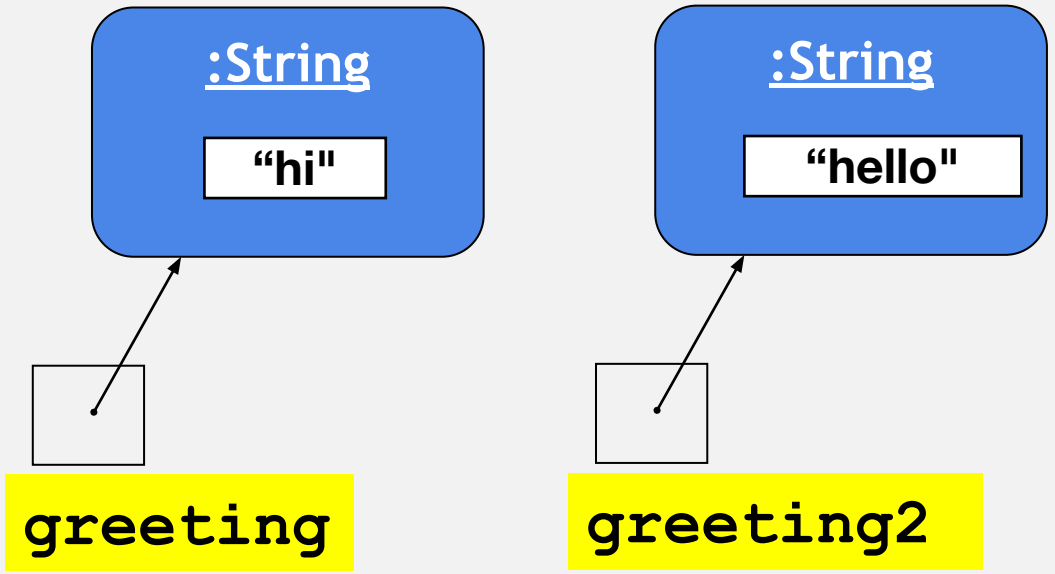
Dec	Char
64	@
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M

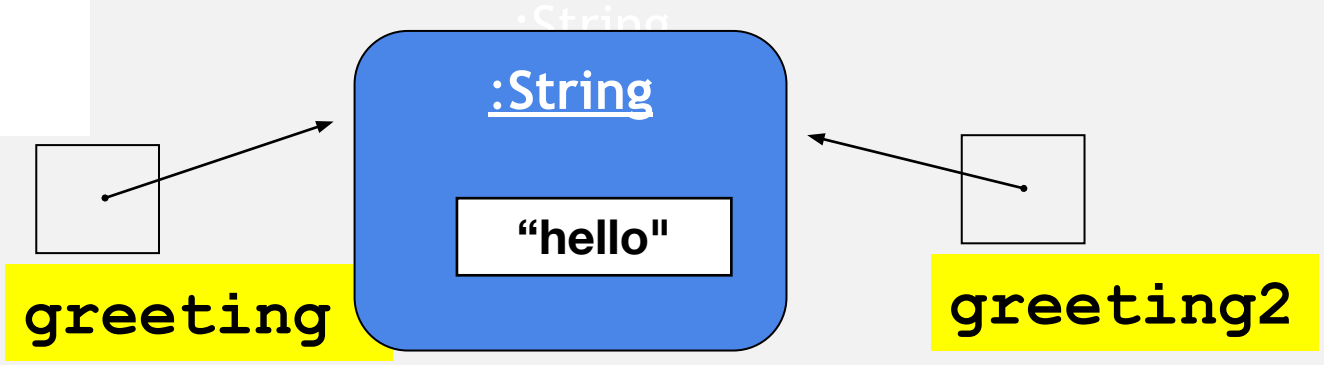
```
>
--->
ooo>
```



INTRODUCTION TO REFERENCES

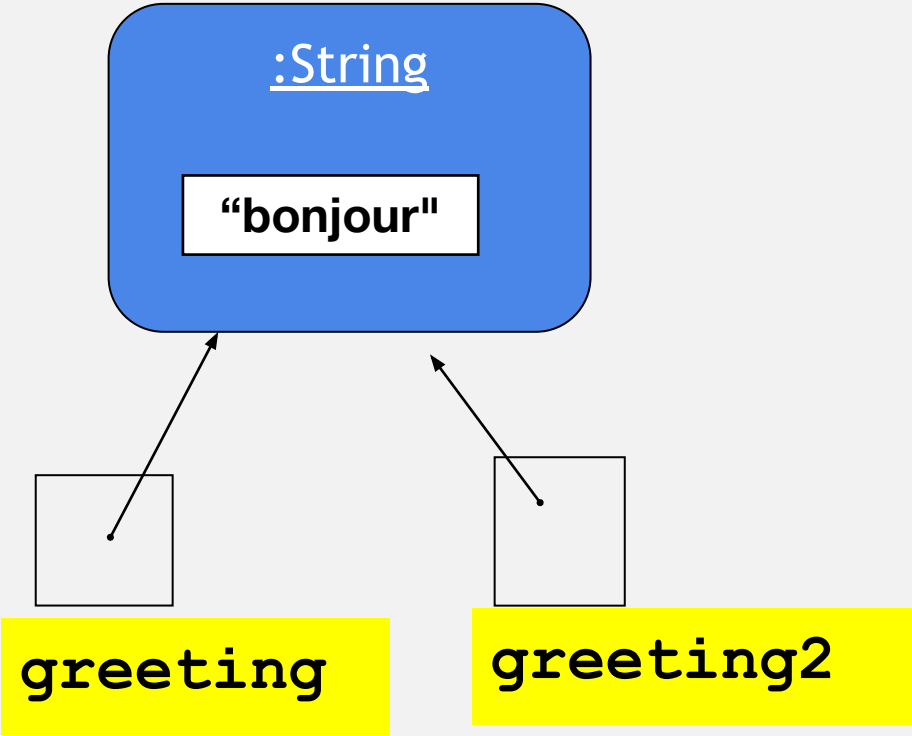
```
String greeting = new
String("hi");

String greeting2 = new
String("hello");

greeting = greeting2;
```





REFERENCES WITH STRING LITERALS

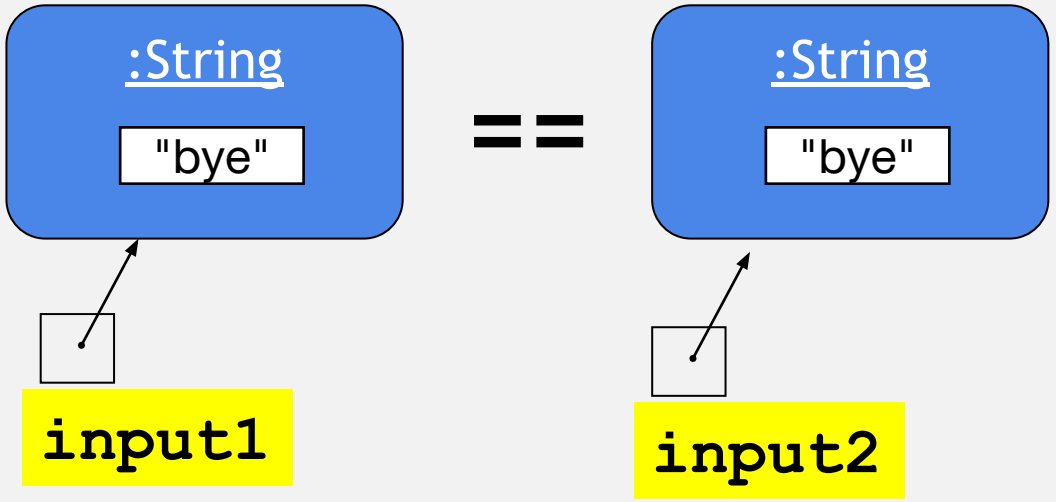
```
String greeting = "bonjour";
String greeting2 = "bonjour";
```



A Literal value is the actual value,
as opposed to a variable which refers to a value.

IDENTITY vs EQUALITY (STRINGS)

```
String input1 = new String("bye");
String input2 = new String("bye");
```



== tests identity

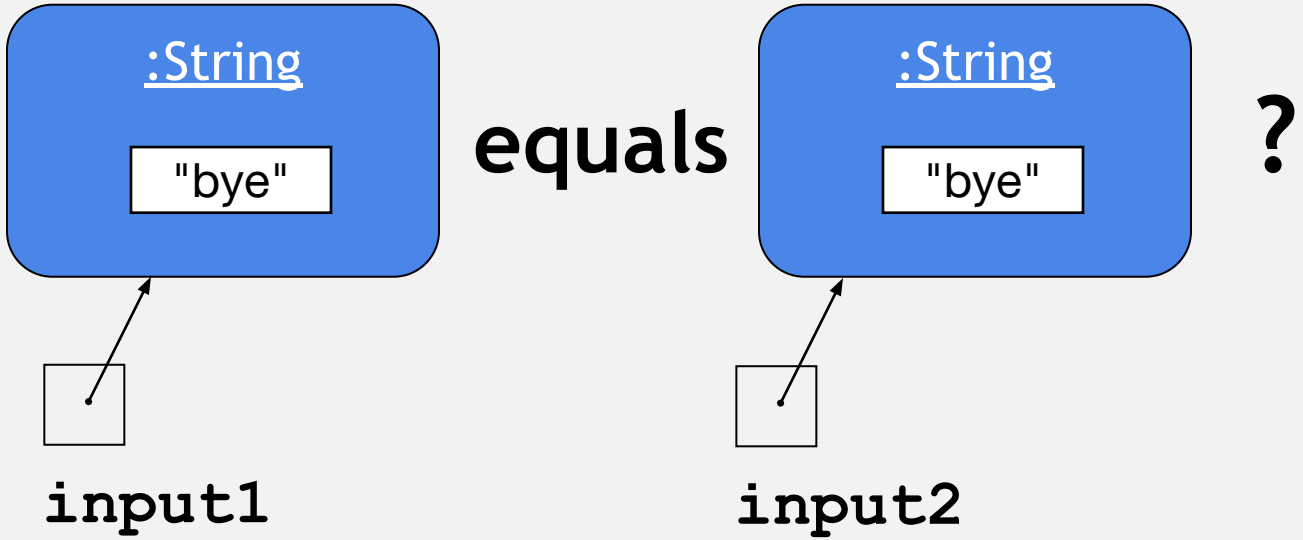
input1==input2

false!

```
input1 --> "bye" (obj 152 : java.lang.String)  java.lang.String
input2 --> "bye" (obj 154 : java.lang.String)  java.lang.String
```

== is not true here (different objects)

IDENTITY VS EQUALITY (STRINGS)



`input1.`*`equals`*`(input2)`

TRUE!

`equals` tests equality

Only use "==" to test if whole numbers or characters are equal. For String use the method *`equals`*

EQUALITY OPERATORS: SUMMARY

Checks whether two operands' values are the same (==) or different (!=).

Evaluates to a Boolean value: TRUE or FALSE.

Also useful for checking expressions e.g.

```
int numItems = 3;
if (numItems == 3) {
    numItems = numItems + 1;
}
```

CLICKER QUESTION #3

```
String name1 = "Grace Hopper";
String name2 = new String("Grace Hopper");
```

Evaluate: name1 == name2

A. True
B. False
C. Maybe
D. Don't know

CLICKER QUESTION #3 ANSWER

```
String name1 = "Grace Hopper";
String name2 = new String("Grace Hopper");
```

Evaluate: name1 == name2

A. True
B. False
C. Maybe
D. Don't know

Evaluate:
`name1.equals(name2)`
Ans: TRUE

```
name1 --> "Grace Hopper" (obj 134 : java.lang.String)  java.lang.String
name2 --> "Grace Hopper" (obj 143 : java.lang.String)  java.lang.String
```

REVIEW: SCANNER CLASS METHODS

```
Scanner console = new Scanner(System.in);
```

Scanner object

means: input from keyboard

`console.nextInt()` --- looking for int value

`console.nextDouble()` --- looking for double value

`console.next()` --- looking for String value

`console.nextLine()` --- looking for a whole line

https://docs.oracle.com/javase/10/docs/api/java/util/Scanner.html

CLICKER QUESTION #4

```
Scanner sc = new Scanner(System.in);
System.out.println("Enter first phrase");
```

Which choice retrieves all that is typed?

A. `String phrase = sc.nextInt();`
B. `String phrase = sc.nextLine();`
C. `String phrase = sc.nextPhrase;`
D. `String phrase = sc.nextLine;`
E. `String phrase = sc.nextString();`

CLICKER QUESTION #4 ANSWER

```
Scanner sc = new Scanner(System.in);
System.out.println("Enter first phrase");
```

Which choice retrieves all that is typed?

A. `String phrase = sc.nextInt();` For ints

B. `String phrase = sc.nextLine();`

C. `String phrase = sc.nextPhrase;` No such method

D. `String phrase = sc.nextLine;` Missing ()

E. `String phrase = sc.nextString();` No such method

TO-DO

- Check your iClicker grades in Moodle.
- Study zyBook chapter 1-3 (content for exam. Does not include Optional sections and Strings). Do the online practice exam and worksheets.
- Communicate with us using only Moodle forum or Piazza.
- Start Project 2 early - seek help in office hours.
- Start zyBooks chapter 4 exercises.
- **Read ALL the exam instructions sent to you.**

EXAM IMPORTANT POINTS

- **Charge your laptop battery.**
- Come to your lab room (don’t come to another lab) early .
- Have your ID out for check in.
- Silence and stow your phone.
- Stay in your seat until exam time ends even if you finish early.
- Click “Submit Quiz” when you finish the quiz.