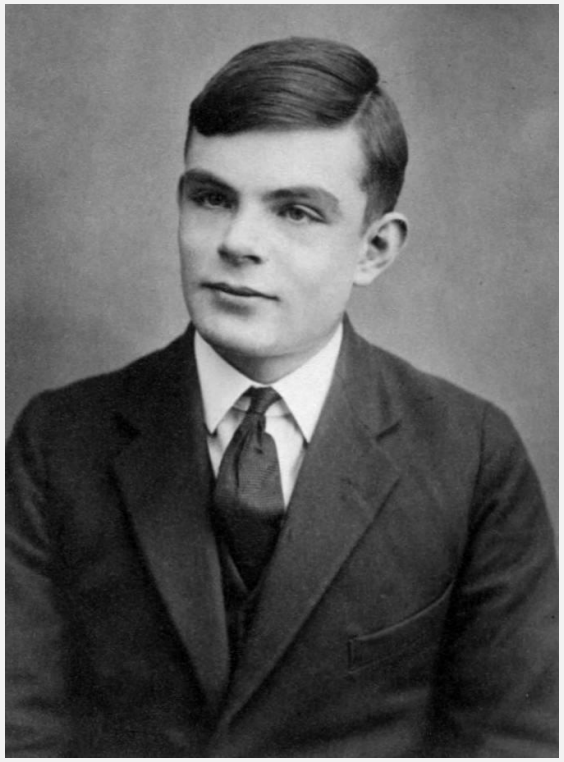
COMPSCI 121: DATA TYPES & BRANCHING

SPRING 20

BYTE FROM COMPUTING HISTORY

Alan Turing, also known as the father of modern computing was a brilliant mathematician and logician. He developed the idea of the modern computer and artificial intelligence.



Read about the Turing Machine.

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:

- New data types
- Conditional branching
- Math methods and operators

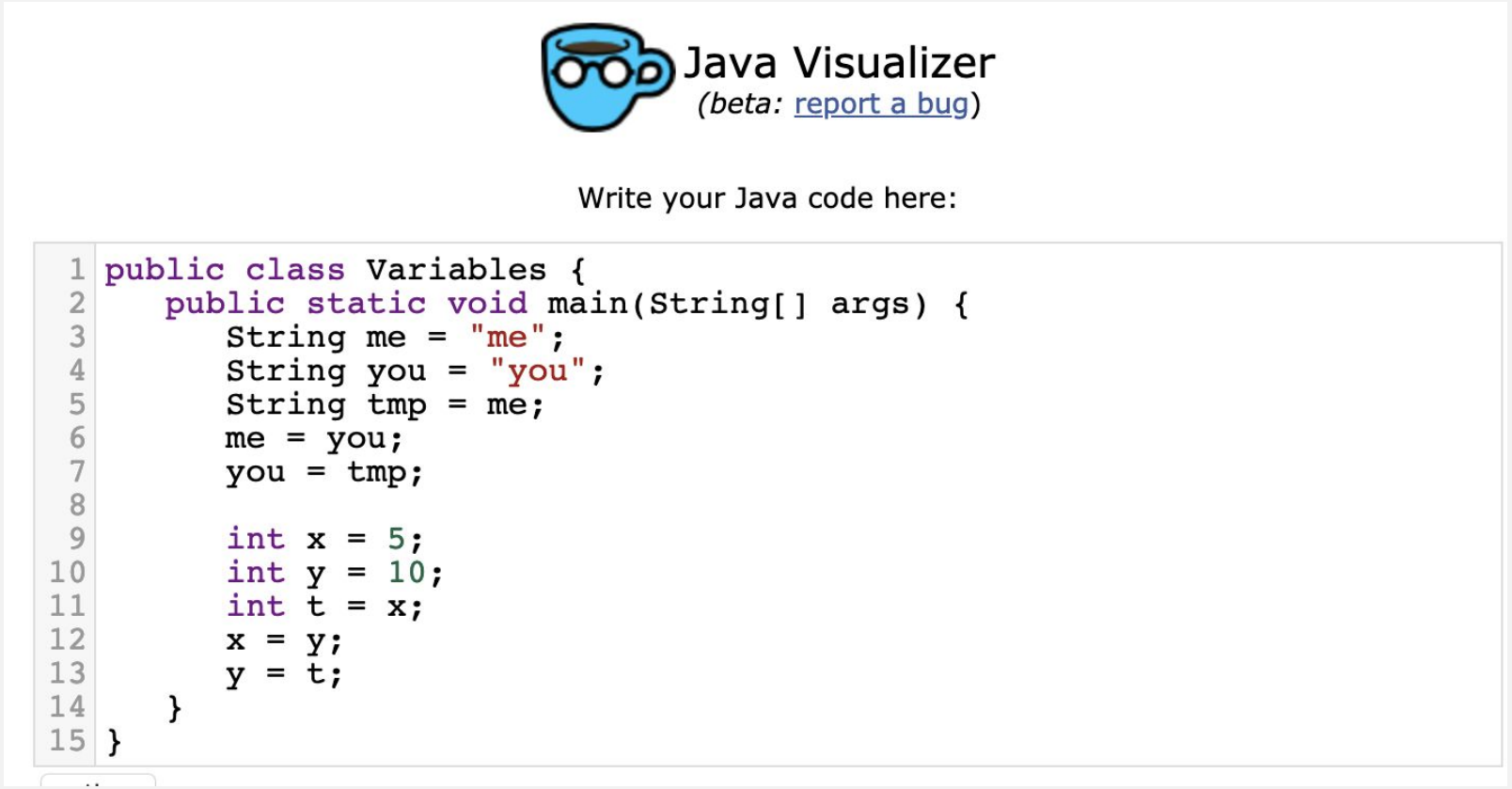
FIRST: LET'S DO SOME REVIEW

Website Java Visualizer:

https://cscircles.cemc.uwaterloo.ca/java_visualize/

Look at Basic Examples:

basic examples | (Default) | Variables |



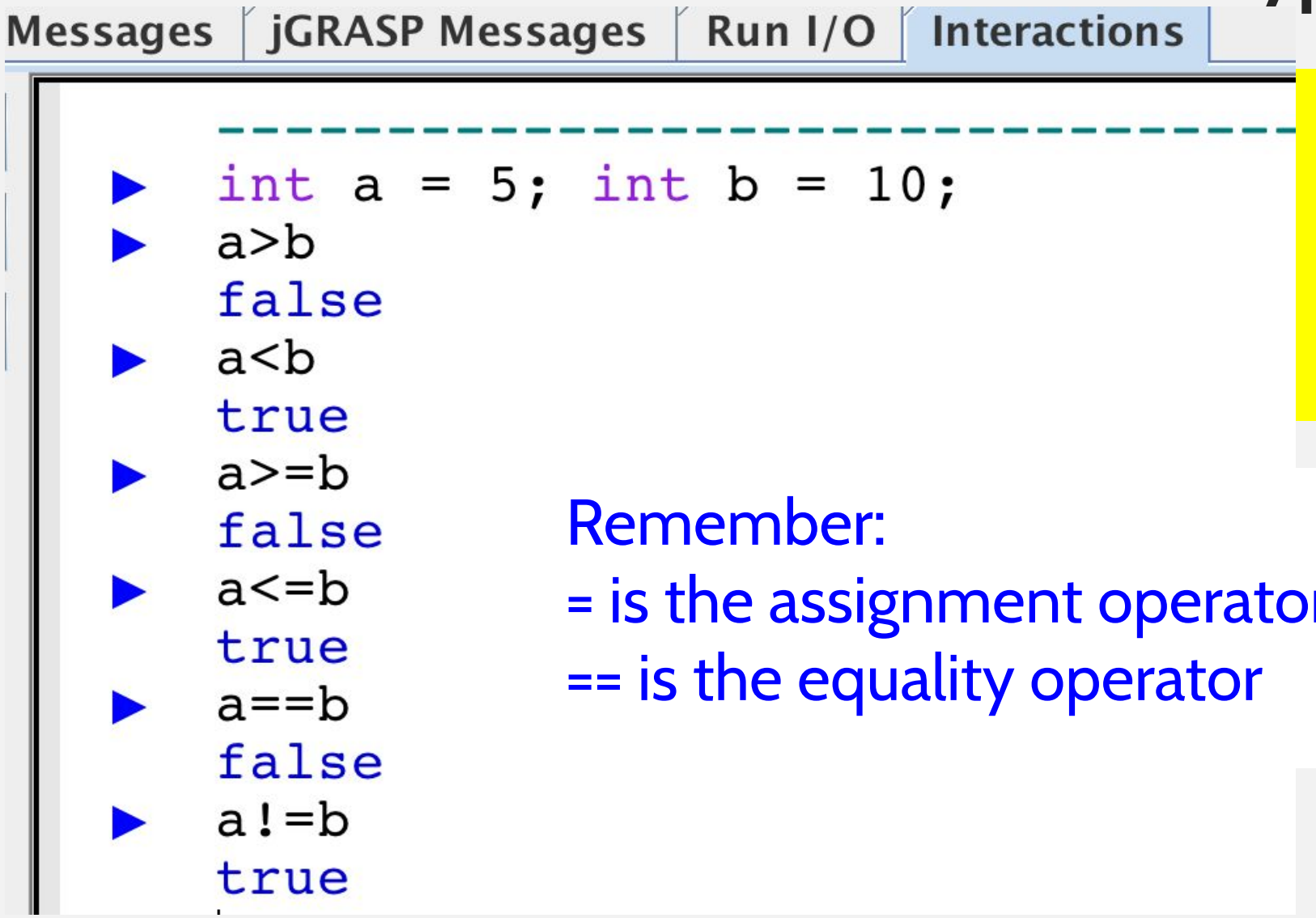
DATA TYPES IN JAVA

- Data is represented as "**types**" in programming.
- In Java, a data type is defined as a **primitive or a class.**
- "**Primitive**" Data types and their values:
  - **Numeric**: **int**, whole numbers, 2, 400, -99

    double, fractional numbers, 7.34, 8.01, -100.00
  - **Character**: **char**, alpha-numeric symbols, 'a', 'G', '&' , '}', '**9**' (notice 9 is not a number but character 9).
  - **Boolean**: **boolean**, true or false (these values are keywords).

RELATIONAL OPERATORS

Check conditions: Result of boolean type: true or false



Remember:
= is the assignment operator
== is the equality operator

Two sides of a relational operator have to be compatible types.

Note:
==
!=

CONDITIONAL IF-ELSE

```
if (<some boolean test>)
     do something;

if (<some boolean test>)
    { do a bunch of things;}

if (<some boolean test>)
    {do a bunch of things;}
else
    {do a bunch of other things;}
```

Relational operators

<
>
<=
>=

Boolean evaluates to TRUE / FALSE

If no curly braces, only the first line after the *if* statement is executed.

CONDITIONALS - STYLE 1

```
if (x > 0) {
    System.out.println("x is positive");
}
else if (x < 0) {
    System.out.println("x is negative");
}
else {
    System.out.println("x is zero");
}
```

What is the answer if x = -2?

See Demo from Java Visualizer:
`ControlFlow`

CONDITIONALS - STYLE 2

```
if (x == 0) {
    System.out.println("x is zero");
}
else {
    if (x > 0) {
        System.out.println("x is positive");
    }
    else {
        System.out.println("x is negative");
    }
}
```

USE OF CONDITIONALS

```
Scanner scan = new Scanner(System.in);
System.out.println("Enter a positive integer");
int number = scan.nextInt();

if (number < 0)
    System.out.println("Number entered isn't positive");

if (number > 0)
    System.out.println(number + " is positive");
else
    System.out.println(number + " is negative or zero");
```

Use conditionals for user error checks!

JAVA ARITHMETIC OPERATORS

- +, -, * behave in standard way.
- Division / is different
- In the absence of parentheses, *, /, have higher precedence than +, -
  - So: `(3 + 5 * 2)is 13`
    `(7 - 4 / 2) is 5`

```
5/3
1
5.0/3
1.6666666666666667
5/3.0
1.6666666666666667
```

See Demo from Java Visualizer:
`CmdLineArgs`

JAVA ARITHMETIC OPERATORS

Modulo mod operator %

With integers:

Gives integer remainder after repeated divisions of the number by the modulus:

number % modulus

```
10      %   7       is 3
21      %   7       is 0
53      %   7       is 4
```

KEEP IN MIND

1. * / % have higher priority and performed first.
2. For same priority, operators are applied from left to right.
3. Integer division always rounds towards zero.
4. When one or more operands are double, Java performs "floating-point" division.

Check the Java™ Tutorials:
https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

MATH OPERATOR RULES

- Dividing by zero gives error

```
Exception in thread "main"
java.lang.ArithmeticException: / by zero
```

- `int`-to-`double` conversion is automatic but `double`-to-`int` conversion may lose precision.
- Use *type casting* to convert a value of one type to another type. e.g.

```
myIntVar = (int)myDoubleVar
myDoubleVar = (double)myIntVar
```

MATH METHODS - EXAMPLE

Static method: Independent of class object

`static double sqrt(double a)`	Returns the correctly rounded positive square root of a `double` value.

sqrt

```
public static double sqrt(double a)
```

Returns the correctly rounded positive square root of a `double` value. Special cases:

- If the argument is NaN or less than zero, then the result is NaN.
- If the argument is positive infinity, then the result is positive infinity.
- If the argument is positive zero or negative zero, then the result is the same as the argument.

Otherwise, the result is the `double` value closest to the true mathematical square root of the argument value.

Parameters:

a - a value.

Returns:

the positive square root of a. If the argument is NaN or less than zero, the result is NaN.

https://docs.oracle.com/javase/10/docs/api/java/lang/Math.html

REVIEW: METHODS AND PARAMETERS

```
static double          max(double a, double b)
                       Returns the greater of two double values.
```

```
double r = Math.max(3.5, 7.1);
```

Return

Arguments

Parameters

From the Java API, Math class `max` method summary.

Parameters refers to the list of variables in a method declaration.
Arguments are the actual values that are passed in when the method is invoked.
When you invoke a method, the *arguments* used **must match** the declaration's *parameters* in **type and order**.

MATH METHOD EXAMPLES

```
double r = Math.max(3.5, 7.1); //7.1
double s = Math.sqrt(2.0);
//1.4142135623730951
double t = Math.sin(.7); //0.6442176872376911
double u = Math.min(3.5, 7.1); //3.5
double v = Math.pow(2,5);  //32.0
```

TO-DO

- Check your iClicker grades in Moodle.
- Exam 1: 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.