UBC OKANAGAN

COSC 111
Computer Programming I

Introduction to Computers, Programs, and Java

Dr. Firas Moosvi

Acknowledgement: Slides (provided by Dr. Abdallah Mohammed) mainly rely on the materials prepared by Y. D. Liang for the textbook "Introduction to Java Programming, 10th Ed., Pearson Edu. Inc.".

Outline

1) First Java Program and Java Class Anatomy

2) Variables, data types, and assignment

3) Literals

4) Programming Errors

5) Reading input from the user

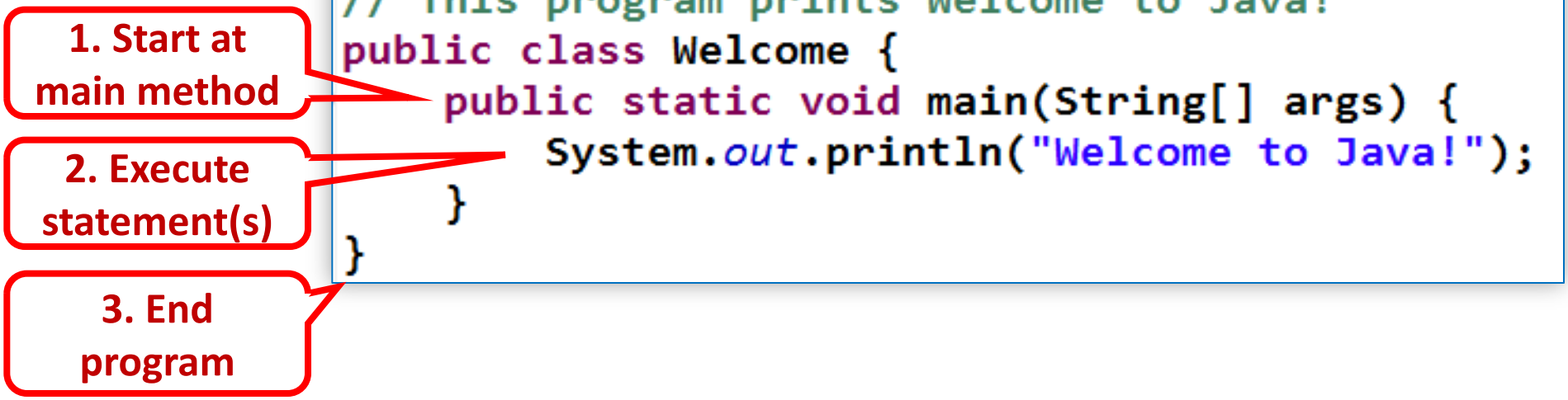
6) Numeric Operations

7) Lecture Activity

First Java Program
and
Java Class Anatomy

Simple Java Program

Animation



Output:

Welcome to Java

Simple Java Program 2

Animation

```
// This program prints Welcome to Java!
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Programming is fun!");
        System.out.println("Fundamentals First");
        System.out.println("Problem Driven");
    }
}
```

Output:

Programming is fun!
Fundamentals First
Problem Driven

Simple Java Program 3

Animation

```
public class ComputeExpression {
    public static void main(String[] args) {
        System.out.println((10.5 + 2 * 3) / (45 - 3.5));
    }
}
```

Output:
0.39759036144578314

Anatomy of a Java Program

Class name

Main method

Statements

Statement terminator

Reserved words

Comments

Blocks

Anatomy of a Java Program

Class name

Main method

Statements

Statement terminator

Reserved words

Comments

Blocks

- Every Java program must have at least one class. Each class has a name. By convention, class names start with an uppercase letter. In this example, the class name is Welcome.

```
// This program prints Welcome to Java!
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

Anatomy of a Java Program

Class name

Main method

Statements

Statement terminator

Reserved words

Comments

Blocks

- Line 2 defines the main method. In order to run a class, the class must contain a method named main. The program is executed from the main method.

```
// This program prints Welcome to Java!
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

Anatomy of a Java Program

Class name

Main method

Statements

Statement terminator

Reserved words

Comments

Blocks

- A statement represents an action or a sequence of actions. The statement System.out.println("Welcome to Java!") in the program in Listing 1.1 is a statement to display the greeting "Welcome to Java!".

```
// This program prints Welcome to Java!
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

Anatomy of a Java Program

Class name

Main method

Statements

Statement terminator

Reserved words

Comments

Blocks

- Every statement in Java ends with a semicolon (;).

```
// This program prints Welcome to Java!
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

Anatomy of a Java Program

Class name
Main method
Statements
Statement terminator
Reserved words
Comments
Blocks

- Reserved words or keywords are words that have a specific meaning to the compiler and cannot be used for other purposes in the program. For example, when the compiler sees the word class, it understands that the word after class is the name for the class.

```
// This program prints Welcome to Java!
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

Anatomy of a Java Program

- Class name
- Main method
- Statements
- Statement terminator
- Reserved words
- **Comments**
- Blocks

Used to document programs and improve their readability. Two types:

- End-of-line comment: it terminates at the end of the line on which it appears.

```
// This is a comment!
```

- Block comment, can be spread over several lines

```
/* This comment. is split over
   multiple lines
*/
```

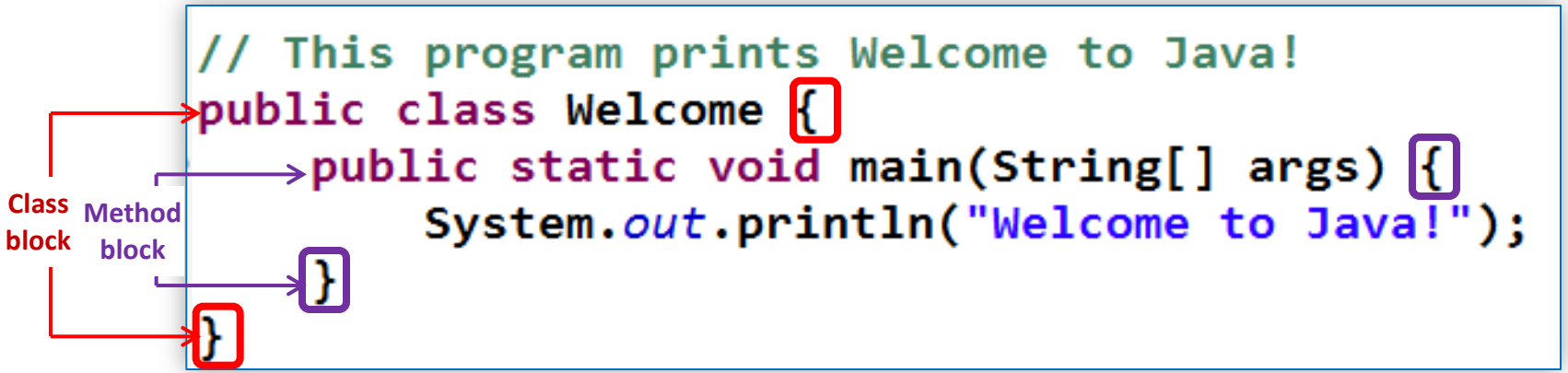
Compiler ignores comments.

```
// This program prints Welcome to Java!
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

Anatomy of a Java Program

Class name
Main method
Statements
Statement terminator
Reserved words
Comments
Blocks

- A pair of braces in a program forms a block that groups components of a program.



Special Symbols

Character	Name	Description
{}	Opening and closing braces	Denotes a block to enclose statements.
()	Opening and closing parentheses	Used with methods.
[]	Opening and closing brackets	Denotes an array.
//	Double slashes	Precedes a comment line.
" "	Opening and closing quotation marks	Enclosing a string (i.e., sequence of characters).
;	Semicolon	Marks the end of a statement.

Special Symbols

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Guidelines

Naming Conventions

- Choose meaningful and descriptive names.
- Class names:
  - Capitalize the first letter of each word in the name. For example, the class name ComputeExpression.

User Proper Indentation and Spacing

- Indentation: Indent two spaces.
- Spacing: Use blank line to separate segments of the code.

Some rules

To program in Java you must follow a set of rules for specifying your commands. This set of rules is called a ***syntax***.

Important general rules of Java syntax:

- Java is ***case-sensitive***.
  - `Main()` is not the same as `main()` or `MAIN()`.
- Java accepts ***free-form layout***.
  - Spaces and line breaks are not important except to separate words.
  - You can have as many words as you want on each line or spread them across multiple lines.
  - However, you should be consistent and follow the programming guidelines given for assignments.
    - It will be easier for you to program and easier for the marker to mark.

More about System.out.println

System.out.println is a Java statement used to print the argument passed to it.

Possible arguments include numbers and strings

If an expression is used, the expression is evaluated first then the result is printed.

```
System.out.println("Hi");
System.out.println("Hi " + "There");
System.out.println(3 + 6);
System.out.println("High " + 5);
System.out.println(2 + 5 + "A");
System.out.println("A" + (2 + 5));
System.out.println("A" + 2 + 5);
```

Output

```
Hi
Hi There
9
High 5
7A
A7
A25
```

More about System.out.println

When we have a mathematical expression that involves (+,-,*,/), multiplication (*) and division(/) have ***higher precedence*** (i.e. computed first) than addition(+) and subtraction(-).

- e.g. 3 + 4 * 2 is the same as 3 + (4 * 2) and equal to 11

If an expression has two operators with the same precedence, then for the above mathematical operators they are evaluated from left to right..

- e.g. 3 + 4 - 2 is the same as (3 + 4) - 2 which becomes 7 - 2

```
System.out.println(2 + 5 * 3 - 1);
System.out.println("A" + (5 - 2) );
System.out.println("A" + 2 - 5);
```

Output

```
16
A3
Error (Why?)
```

Clicker Question

What is the output of the following program

```
public class Q{
   public static void main(String[] args) {
      System.out.println("3 + 4 is: ");
      System.out.println(3+4);
   }
}
```

A. 7 is:
   7

B. 3 + 4 is:
   3+4

C. 7 is:
   3+4

D. 3 + 4 is: 7

E. 3 + 4 is:
   7

Clicker Question

What is the output of the following program

```
public class Q{
   public static void main(String[] args) {
      System.out.println("3 + 4 is " + 3 + 4);
   }
}
```

A. 7 is 7

B. 3 + 4 is 3 + 4

C. 3 + 4 is 7

D. 3 + 4 is 34

E. None of the above

Clicker Question

What is the output of the following program

```
public class Q{
   public static void main(String[] args) {
      System.out.println("3 + 4 is " + (3 + 4));
   }
}
```

A. 7 is 7

B. 3 + 4 is 3 + 4

C. 3 + 4 is 7

D. 3 + 4 is 34

E. None of the above

Clicker Question

What is the output of the following program

```
public class Q{
   public static void main(String[] args) {
      System.out.println("3" + 5);
   }
}
```

A. 8

B. 35

C. Error

D. None of the above

Clicker Question

What is the output of the following program

```
public class Q{
   public static void main(String[] args) {
      System.out.println("Result is " + 5 * 3);
   }
}
```

A. Result is 53

B. Result is 15

C. Result is 5 Result is 5 Result is 5

D. Error

E. None of the above

Variables, data types, and assignment

Variables, data types, and assignment

A variable in java…

- is a location in the computer's memory that is used to store data in a program.
- must be declared with a **name (identifier)** and **type**.

Example 1:

```
int x;                  // declare a variable
x = 5;                  // initialize a variable – what is assignment '='?
int y = 10;             // declare and initialize a variable
System.out.println(x); // print value of x
x = 10;                 // overwrite old value
System.out.println("x " + x);  //what is the output?
```

Example 2:

```
int x = 10, y;          // y has no values yet
y = x;                  // y is 10 now
y = y + 1;              // '=' does not mean equal, it means assignment.
System.out.println("x + y = " + (x + y));     //notice the output
```

Trace a Program Execution

Animation

```
public class ComputeArea {
    /* Main method */
    public static void main(String[] args) {
        double radius;
        double area;

        // Assign a radius
        radius = 20;

        // Compute area
        area = radius * radius * 3.14159;

        // Display results
        System.out.println("The area for the circle of radius "
                           + radius + " is " + area);
    }
}
```

Declare a var

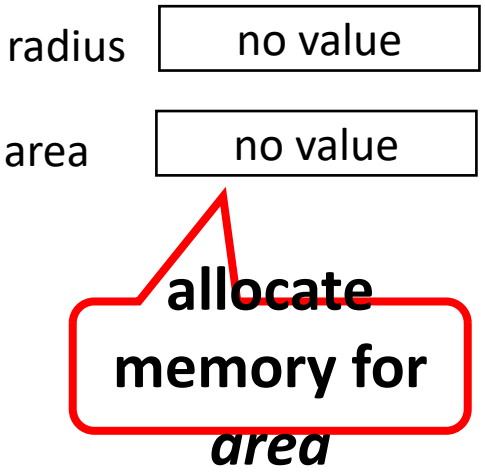
(i.e., allocate memory for the variable)

- Variable **name**: radius
- Variable **type**: real number (double)

radius no value

Trace a Program Execution

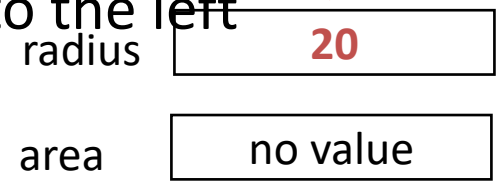
```
public class ComputeArea {
    /* Main method */
    public static void main(String[] args) {
        double radius;
        double area;

        // Assign a radius
        radius = 20;

        // Compute area
        area = radius * radius * 3.14159;

        // Display results
        System.out.println("The area for the circle of radius "
                           + radius + " is " + area);
    }
}
```



Trace a Program Execution

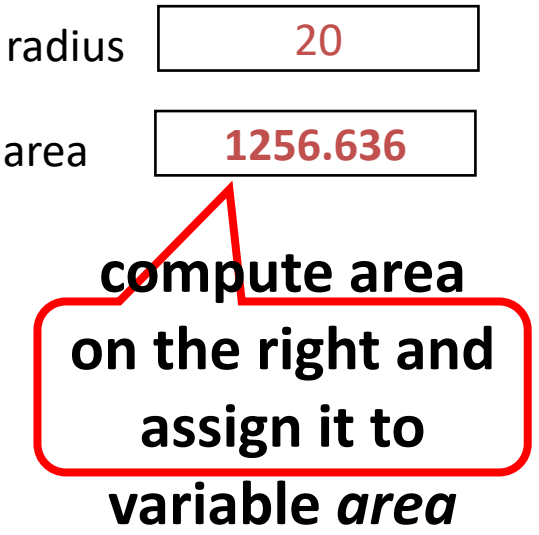
```
public class ComputeArea {
    /* Main method */
    public static void main(String[] args) {
        double radius;
        double area;

        // Assign a radius
        radius = 20;

        // Compute area
        area = radius * radius * 3.14159;

        // Display results
        System.out.println("The area for the circle of radius "
                           + radius + " is " + area);
    }
}
```

The **assignment operator (=)** indicates that value of 20 on the right will be assigned to (**stored in the memory location of**) the variable (radius) to the left



Trace a Program Execution

```
public class ComputeArea {
    /* Main method */
    public static void main(String[] args) {
        double radius;
        double area;

        // Assign a radius
        radius = 20;

        // Compute area
        area = radius * radius * 3.14159;

        // Display results
        System.out.println("The area for the circle of radius "
                           + radius + " is " + area);
    }
}
```

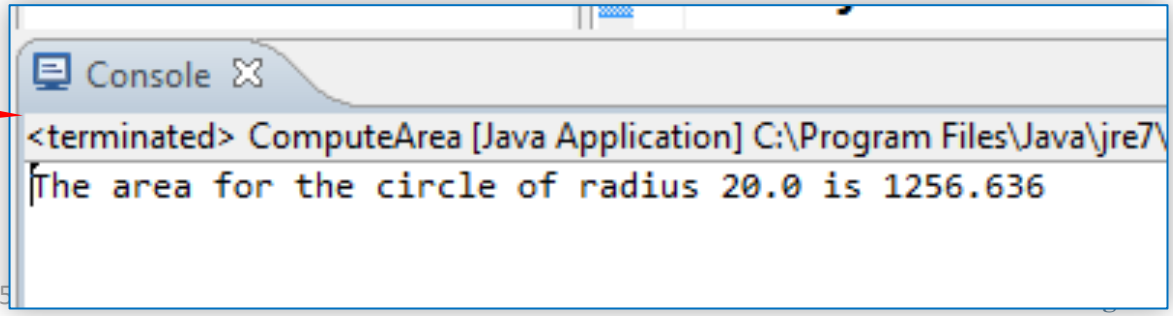


Trace a Program Execution

```
public class ComputeArea {
    /* Main method */
    public static void main(String[] args) {
        double radius;
        double area;

        // Assign a radius
        radius = 20;

        // Compute area
        area = radius * radius * 3.14159;

        // Display results
        System.out.println("The area for the circle of radius "
                           + radius + " is " + area);
    }
}
```

radius	20
area	1256.636

print a message to the console



Variables

Declaring Variables

```
double a;      // Declare a to be a double variable
int x, y;      // Declare x and y to be integer variables
```

Assignment Statements

```
a = 7.1;       // Assign 7.1 to a;
x = 1 + 3;     // assign 4 to x;
y = x + 2;     // assign 6 to y;
```

Declaring and Initializing in One Step

```
double a = 7.1;
int x = 1, y = 2;
```

Identifiers

A variable must be declared before it can be assigned a value.

- declared with a ***name*** and ***type***.

An **identifier** is a sequence of characters that consist of

- letters,
- digits,
- underscores (_), and
- dollar signs ($).

An identifier must start with a letter, an underscore (_), or a dollar sign ($).

- It cannot start with a digit.

An identifier cannot be a reserved word.

- See Appendix A, "Java Keywords," for a list of reserved words.

An identifier cannot be `true`, `false`, or `null`.

An identifier can be of any length.

Primitive Data Types

A variable must be declared before it can be assigned a value.

- declared with a ***name*** and ***type***.

	Java	Size in memory	Range
whole numbers	byte	8 bits	-2^7 to 2^7-1 (-128 to 127)
	short	2 bytes	-2^{15} to $2^{15}-1$ (-32768 to 32767)
	int	4 bytes	-2^{31} to $2^{31}-1$
	long	8 bytes	-2^{63} to $2^{63}-1$
real numbers	float	4 bytes	~1.4E-45 to 3.4E+38 (+ve or -ve)
	double	8 bytes	~4.9E-324 to 1.8E+308 (+ve or -ve)
characters	**char**	2 bytes	e.g. 'a', '1' and '?'
boolean	**boolean**	1 byte	true or false

Literals

Number Literals

A *literal* is a constant value that **appears directly** in the program.

For example, 34, 1000000, and 5.0 are literals in the following statements:

```
int i = 34;

long x = 1000000;

double d = 5.0;
```

Integer Literals

An integer literal can be assigned to an integer variable as long as it can fit into the variable.

- A compilation error would occur if the literal were too large for the variable to hold.
  - For example, the statement byte b = 1000 would cause a compilation error, because 1000 cannot be stored in a variable of the byte type.

An integer literal is assumed to be of the **`int`** type, whose value is between -2^{31} to $2^{31}-1$.

To denote an integer literal of the **`long`** type, append it with the letter L or l.

- L is preferred because l (lowercase L) can easily be confused with 1 (the digit one).

Floating-Point Literals

Floating-point literals are written with a decimal point. By default, a floating-point literal is treated as a **`double`** type value.

- For example, **5.0** is considered a **double** value, not a float value.
- You can make a number a **float** by appending the letter **f or F**, and make a number a **double** by appending the letter **d or D**.
  - For example, you can use 100.2f or 100.2F for a float number, and 100.2d or 100.2D for a double number.

Floating-point literals can also be specified in **scientific notation**.

- For example,
  - 1.23456e+2, same as 1.23456e2, is equivalent to 123.456,
  - 1.23456e-2 is equivalent to 0.0123456.
- E (or e) represents an exponent and it can be either in lowercase or uppercase.

Clicker Question

What is wrong with this code?

```
public class Q{
   public static void main(String[] args) {
      double interestRate = 0.05;
      double interest = interestrate * 45;
   }
}                    // JAVA is CASE SENSITIVE //
```

A. We must print the value of the interest

B. The program is using a variable that is undeclared

C. We must multiply by 45.0, not 45

D. Something else

Clicker Question

Which of the following is a valid Java variable?

A. `aBCde123`

B. `123test`

C. `t_e_s_t!`

D. `my age`

E. `test-123`

Clicker Question

What is printed on the screen?

```
int x, y;

x = 2;
y = 4;
x = y + y / x;
y = x * 5 + 3 * 2;
System.out.println("x:" + x + ", y:"+ y);
```

A. `x:6, y:36`

B. `x:4, y:26`

C. `x:6, y:66`

D. None of the above

Practice

Translate the following simple algorithms into Java code:

Algorithm 1:

- Step 1: Declare a `double` variable named `distance`,
- Step 2: initialize `distance` to 16.5
- Step 3: print `distance` out on the console. The output should look like this:

The distance is 16.5 km

Algorithm 2:

- Step 1: Declare two `int` variables named `x` and `y` and initialize them to 5 and 6
- Step 2: Declare an `int` variable named `sum` and initialize it to the sum of `x` and `y`.
- Step 3: print `sum` on the console. The output should look like this:

The sum of 5 and 6 is 11

Programming Errors

Programming Errors

3 types of errors:

- Syntax Errors
  - Detected by the compiler
  - aka *compilation errors*

```
public class Errors {
    public static   main(String[] args) {
        System.out.println("Welcome to Java");
    }
}
```

- Runtime Errors
  - Causes the program to abort during the runtime.

```
public class Errors {
    public static void main(String[] args) {
        System.out.println(1 / 0);
    }
}
```

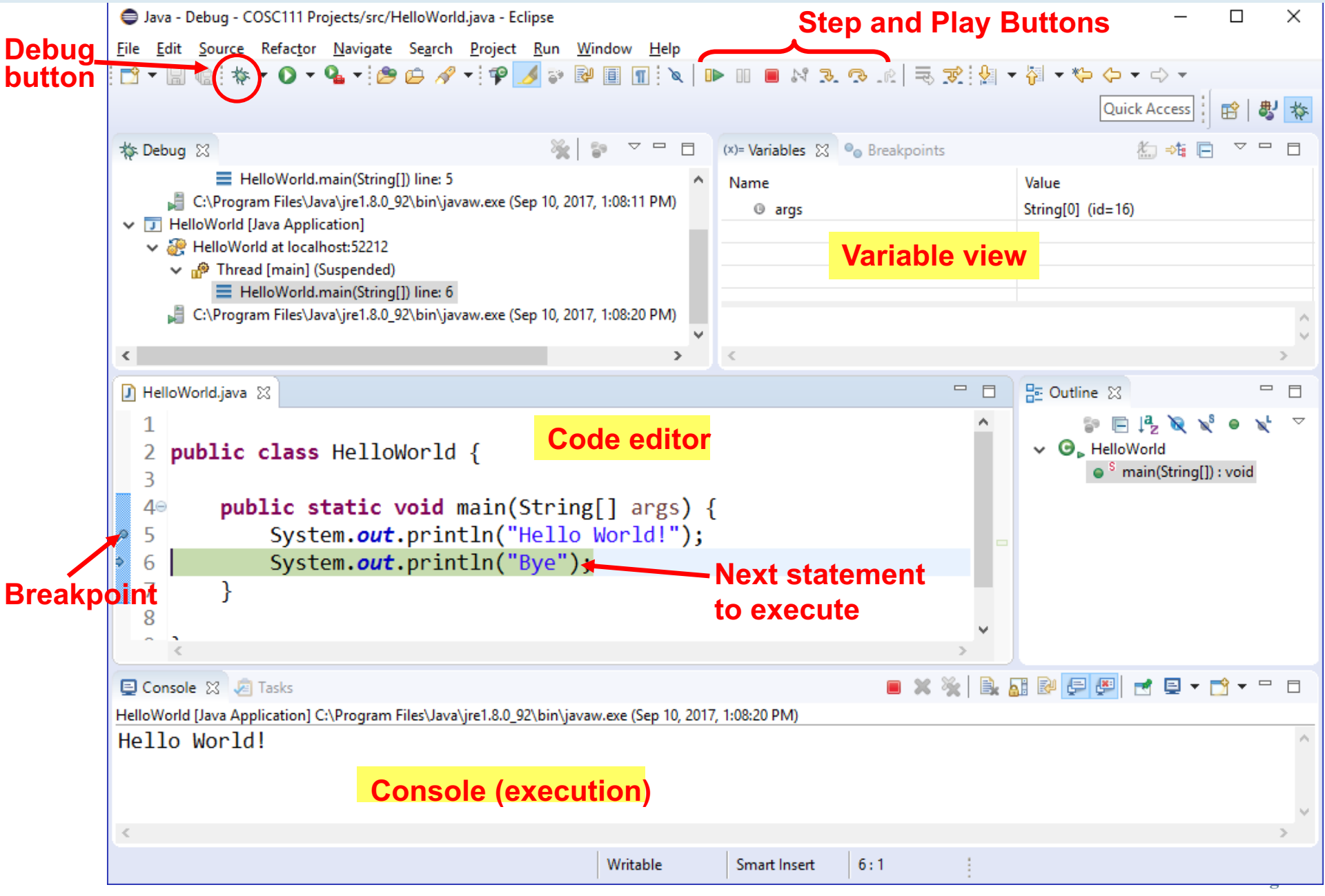
Can't divide by zero

- Logic Errors
  - Produces incorrect result during the runtime
  - no error message is shown

```
public class Errors {
    public static void main(String[] args) {
        System.out.println("35 Celsius in Fahrenheit:");
        System.out.println((9 / 5) * 35 + 32);
    }
}
```

Output is incorrect due to wrong formula

Eclipse: Debugging and Breakpoints



Reading input from the user

Reading Input from the Keyboard

Step1) Create a Scanner object

```
import java.util.Scanner
…
Scanner input = new Scanner(System.in);
        //or var input = new Scanner(System.in) in Java 10
```

Step 2) Use an appropriate method (e.g., nextDouble()) to obtain a double value.

```
System.out.print("Enter a double value: ");
double d = input.nextDouble();
```

Reading Input from the Keyboard

Animation

Rewrite the previous example with reading the radius from the user

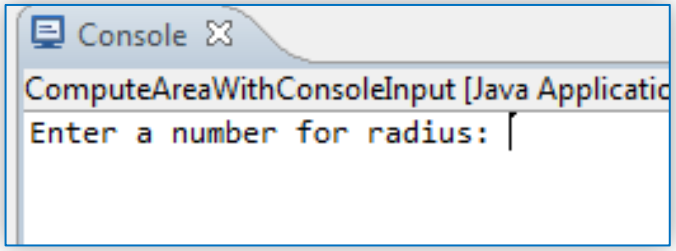
```
import java.util.Scanner; // Scanner is in the java.util package

public class ComputeAreaWithConsoleInput {
    /* Main method */
    public static void main(String[] args) {
        // Create a Scanner object
        Scanner input = new Scanner(System.in);

        // Prompt the user to enter a radius
        System.out.print("Enter a number for radius: ");
        double radius = input.nextDouble();

        // Compute area
        double area = radius * radius * 3.14159;

        // Display results

        System.out.println("The area for the circle of radius "
                           + radius + " is " + area);
    }
}
```

Step 1

Reading Input from the Keyboard

```
import java.util.Scanner; // Scanner is in the java.util package

public class ComputeAreaWithConsoleInput {
    /* Main method */
    public static void main(String[] args) {
        // Create a Scanner object
        Scanner input = new Scanner(System.in);

        // Prompt the user to enter a radius
        System.out.print("Enter a number for radius: ");
        double radius = input.nextDouble();

        // Compute area
        double area = radius * radius * 3.14159;

        // Display results

        System.out.println("The area for the circle of radius "
                           + radius + " is " + area);
    }
}
```

prompt the user for input.

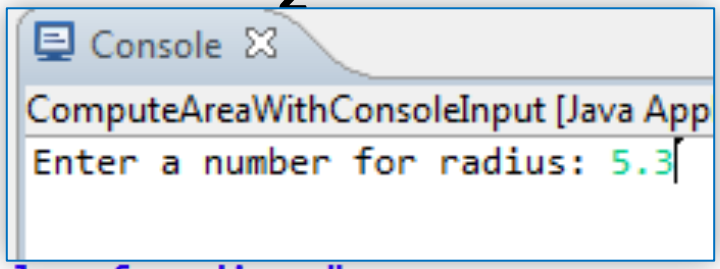


Reading Input from the Keyboard

```
import java.util.Scanner; // Scanner is in the java.util package

public class ComputeAreaWithConsoleInput {
    /* Main method */
    public static void main(String[] args) {
        // Create a Scanner object
        Scanner input = new Scanner(System.in);

        // Prompt the user to enter a radius
        System.out.print("Enter a number for radius: ");
        double radius = input.nextDouble();

        // Compute area
        double area = radius * radius * 3.14159;

        // Display results

        System.out.println("The area for the circle of radius "
                           + radius + " is " + area);
    }
}
```

radius | 5.3

Step 2



Reading Input from the Keyboard

```
import java.util.Scanner; // Scanner is in the java.util package

public class ComputeAreaWithConsoleInput {
    /* Main method */
    public static void main(String[] args) {
        // Create a Scanner object
        Scanner input = new Scanner(System.in);

        // Prompt the user to enter a radius
        System.out.print("Enter a number for radius: ");
        double radius = input.nextDouble();

        // Compute area
        double area = radius * radius * 3.14159;

        // Display results

        System.out.println("The area for the circle of radius "
                           + radius + " is " + area);
    }
}
```

radius 5.3

area 88.2472631

Reading Input from the Keyboard

```
import java.util.Scanner; // Scanner is in the java.util package

public class ComputeAreaWithConsoleInput {
    /* Main method */
    public static void main(String[] args) {
        // Create a Scanner object
        Scanner input = new Scanner(System.in);

        // Prompt the user to enter a radius
        System.out.print("Enter a number for radius: ");
        double radius = input.nextDouble();

        // Compute area
        double area = radius * radius * 3.14159;

        // Display results

        System.out.println("The area for the circle of radius "
                           + radius + " is " + area);
    }
}
```

radius 5.3

area 88.2472631

Console

<terminated> ComputeAreaWithConsoleInput [Java Application] C:\P

```
Enter a number for radius: 5.3
The area for the circle of radius 5.3 is 88.2472631
```

Reading from the Keyboard

Method	Description
`nextByte()`	reads an integer of the `byte` type.
`nextShort()`	reads an integer of the `short` type.
`nextInt()`	reads an integer of the `int` type.
`nextLong()`	reads an integer of the `long` type.
`nextFloat()`	reads a number of the `float` type.
`nextDouble()`	reads a number of the `double` type.
`next()`	reads a 'token' of the `String` type.
`nextLine()`	reads a line of text of the `String` type.

Clicker Question

Assume the following is the complete program (i.e. this is the only code there is). What is the output if the user enters 3 and 4?

```
public class AddTwoNum {
    public static void main(String[] arg){
     Scanner sc = new Scanner(System.in);
     int num1 = sc.nextInt();
     int num2 = sc.nextInt();
     int result = num1 + num2;
     System.out.println(num2 + " + " + num1 + " = " + result);
    }
}
```

A. 3 + 4 = 7

B. 4 + 3 = 7

C. 4 + + + 3 + = + 7

D. num2 + num1 = result

E. Error

Clicker Question

What is the output if the user enters 3 and 4?

```
import java.util.Scanner;
public class AddTwoNum {
    public static void main(String[] arg){
     Scanner sc = new Scanner(System.in);
     int num1 = sc.nextInt();
     int num2 = sc.nextInt();
     int result = num1 + num2;
     System.out.println(num2 + " + " + num1 + " = " + result);
    }
}
```

A. 3 + 4 = 7

B. 4 + 3 = 7

C. 4 + + + 3 + = + 7

D. num2 + num1 = result

E. Error

Practice

- Write a program that asks the user about his/her name and then display a simple greeting message.

- Write a program to read three real numbers (from the user) and display their average.

Tip1: Redundant Input Objects

This code is not wrong, BUT inefficient!

```
Scanner input1 = new Scanner(System.in);
System.out.print("Enter an integer: ");
int v1 = input1.nextInt();

Scanner input2 = new Scanner(System.in);
System.out.print("Enter a double value: ");
double v2 = input2.nextDouble();
```

You should rewrite the above code as follows

```
Scanner input = new Scanner(System.in);
System.out.print("Enter an integer: ");
int v1 = input.nextInt();
System.out.print("Enter a double value: ");
double v2 = input.nextDouble();
```

Tip 2: Reading a Number Followed by a Line

This code has a logic error:

```
Scanner in= new Scanner(System.in);
System.out.println("How old are you?");
int age = in.nextInt();

System.out.println("What is your name?");
String name = in.nextLine();

System.out.println(name + " is " + age + " years old.");
```

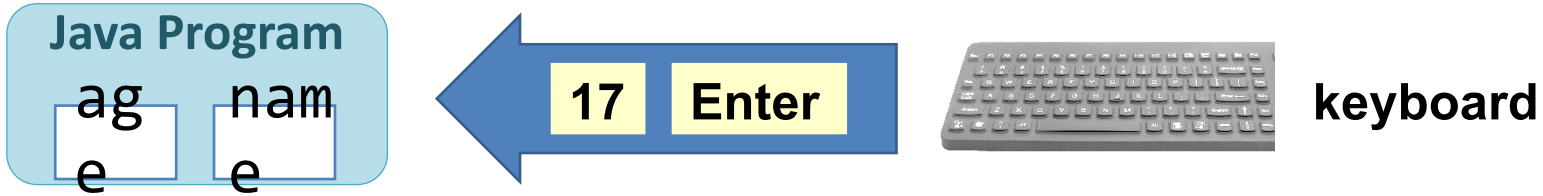
Once you enter your age, you will not get a chance to enter the name and the output will be something like this:

```
How old are you?
12
What is your name?
 is 12 years old.
```

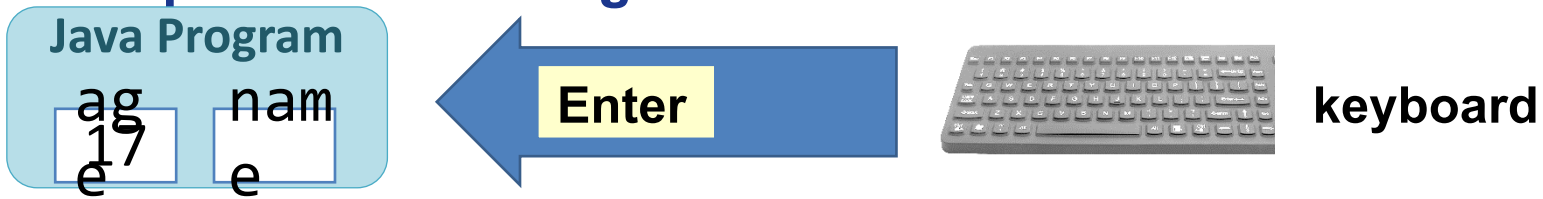
Tip 2: cont'd

Explanation:

- The input received from the user for the age is
  17 followed by "Enter (i.e. new line)"
- Think of the above input as a queue with the values



- The method `nextInt()` reads the value 17 only, removing it from the queue and leaving Enter still in the Queue



- The method nextLine() "thinks" that there is an empty line that is terminated by the remaining Enter – so it immediately reads an empty line into variable name and doesn't wait for the user to enter any more text.

Tip 2: cont'd

How to fix?

Get rid of the extra "Enter" by adding an additional nextLine as shown below.

```
Scanner in= new Scanner(System.in);
System.out.println("How old are you?");
int age = in.nextInt();

in.nextLine()

System.out.println("What is your name?");
String name = in.nextLine();

System.out.println(name + " is " + age + " years old.");
```

Numeric operations

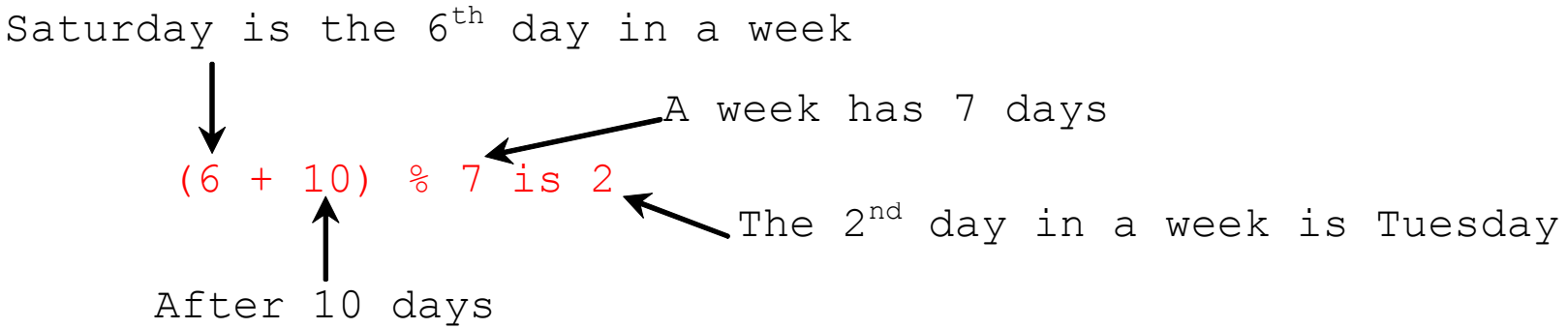
Numeric Operators

Name	Meaning	Example	Result
+	Addition	34 + 1	35
-	Subtraction	34.0 - 0.1	33.9
*	Multiplication	300 * 30	9000
/	Division	1.0 / 2.0	0.5
%	Remainder	20 % 3	2

Remainder operator

Remainder operator **(%)** yields the remainder after division (e.g. 5 % 2 yields 1)

- The remainder is negative only if the dividend is negative.
- Example uses:
  - Can be used to determine whether a number is even or odd.
    - An even number % 2 is always 0 and an odd number % 2 is always 1.
  - Suppose today is Saturday and you and your friends are going to meet in 10 days. What day is in 10 days? You can find that day is Tuesday using the following expression



Exercise

Exercise: Show the result of the following remainders.

- 14 % 6 // 2
- 3 % 0 // Runtime error. Can't divide by zero
- 34 % -5 // 4
- -34 % 5 // -4
- -34 % -5 // -4
- 5 % 1 // 0
- 1 % 5 // 1

Integer Division

- 5 / 2 yields an integer 2
- 5.0 / 2 yields a double value 2.5

Exercise:

1. What is the result of **25 / 4**?

2. How would you rewrite the expression if you wished the result to be a floating-point number?

Practice

Write a program that displays minutes and remaining seconds from seconds entered by the user.

Solution:

Start by writing down the algorithm:

- Step 1: Prompt the user for input. Read seconds from user
- Step 2: Find minutes in seconds. Hint: use /
- Step 3: Find remaining seconds. Hint: use %
- Step 4: Display minutes and seconds.

Problem: Monetary Units

This program lets the user enter the amount in decimal representing dollars and cents (e.g. 17.75) and output a report listing the monetary equivalent in

- single dollars,
- quarters,
- dimes,
- nickels,
- and pennies.

Sample run

```
Enter a monetary amount: 11.56
Your amount 11.56 consists of
 11 dollars
 2 quarters
 0 dimes
 1 nickels
 1 pennies
```

Solution idea:

- Convert the input to cents
  - *11.56 is 1156 cents*
- Use / and % to get the dollars, quarters, etc.
  - E.g.
    - 1156 / 100 is 11 dollars. The remainder is 56 cents.
    - 56/25 = 2 quarters. The remainder is 6 cents
    - etc.

Exponent Operations

The `Math.pow(a, b)` method can be used to compute a^b

```
System.out.println(Math.pow(2, 3)); // Displays 8.0

System.out.println(Math.pow(4, 0.5)); // Displays 2.0

System.out.println(Math.pow(2.5, 2)); // Displays 6.25

System.out.println(Math.pow(2.5, -2)); // Displays 0.16
```

Arithmetic Expressions

$$\frac{3+4x}{5} - \frac{10(y-5)(a+b+c)}{x} + 9\left(\frac{4}{x} + \frac{9+x}{y}\right)$$

is translated to

```
(3+4*x)/5 – 10*(y-5)*(a+b+c)/x + 9*(4/x + (9+x)/y)
```

Practice

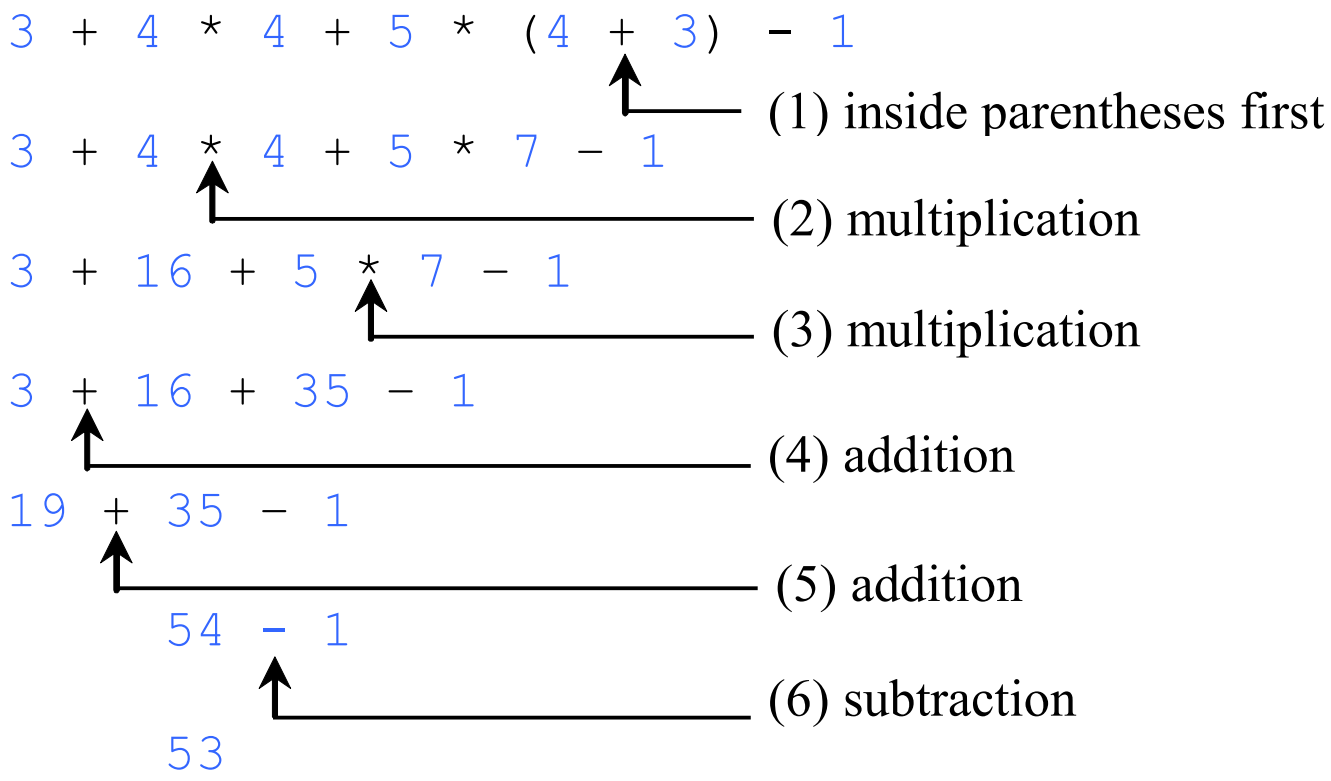
How would you write the following arithmetic expression in Java?

$$\frac{4}{3(r + 34)} - 9(a + bc) + \frac{3 + d(2 + a)}{a + bd}$$

$$5.5 \times (r + 2.5)^{2.5+t}$$

How to Evaluate an Expression

Though Java has its own way to evaluate an expression behind the scene, the result of a Java expression and its corresponding arithmetic expression are the same. Therefore, you can safely apply **the arithmetic rule** for evaluating a Java expression.



Practice

Write a program that converts a Fahrenheit degree given by the user to Celsius using the formula:

$$celsius = (\frac{5}{9})(fahrenheit - 32)$$

Algorithm:

- 1. Prompt the user for the Fahrenheit degree. Store the input in a variable.
- 2. Calculate the Celsius equivalent and store it in a variable
- 3. print out the Celsius degree.

Augmented Assignment Operators

The operators **+**, **-**, *****, **/**, and **%** can be combined with the assignment operator to form augmented operators.

Operator	*Name*	*Example*	*Equivalent*
`+=`	Addition assignment	`i += 8`	`i = i + 8`
`-=`	Subtraction assignment	`i -= 8`	`i = i - 8`
`*=`	Multiplication assignment	`i *= 8`	`i = i * 8`
`/=`	Division assignment	`i /= 8`	`i = i / 8`
`%=`	Remainder assignment	`i %= 8`	`i = i % 8`

Increment and Decrement Operators

The increment operator (++) and decrement operator (– –) are for incrementing and decrementing a variable by 1.

Operator	*Name*	*Description*	*Example (assume i = 1)*
`++var`	preincrement	Increment `var` by `1`, and use the new `var` value in the statement	`int j = ++i;` `// j is 2, i is 2`
`var++`	postincrement	Increment `var` by `1`, but use the original `var` value in the statement	`int j = i++;` `// j is 1, i is 2`
`--var`	predecrement	Decrement `var` by `1`, and use the new `var` value in the statement	`int j = --i;` `// j is 0, i is 0`
`var--`	postdecrement	Decrement `var` by `1`, and use the original `var` value in the statement	`int j = i--;` `// j is 1, i is 0`

Increment and Decrement Operators

```
int i = 10;
int newNum = 10 * i++;
```

Same effect as

```
int newNum = 10 * i;
i = i + 1;
```

```
int i = 10;
int newNum = 10 * (++i);
```

Same effect as

```
i = i + 1;
int newNum = 10 * i;
```

Clicker Question

What is the output?

```
System.out.println("3 * 2 / 4 is " + 3 * 2 / 4);
System.out.println("3 * 2 / 4 is " + 3.0 * 2 / 4);
```

A. 3 * 2 / 4 is 1.5
   3 * 2 / 4 is 1.5

B. 3 * 2 / 4 is 1
   3 * 2 / 4 is 1.5

C. 3 * 2 / 4 is 1.5
   3 * 2 / 4 is 1

D. Error

Clicker Question

What is the output?

```
int x = 6;

System.out.println("x: " + x++);

System.out.println("x: " + ++x);
```

A. x: 6
   x: 7

B. x: 6
   x: 8

C. x: 7
   x: 8

D. Error

Clicker Question

What is the output?

A. 6 7
   7 6

B. 7 6
   7 7

C. 7 7
   7 7

D. 7 7
   7 6

E. 7 7
   6 7

```
int x = 6;
System.out.print(++x);
System.out.println(x);

int y = 6;
System.out.print(y+1);
System.out.println(y);
```

Practice

Show the output of the following code:

```
int a = 6;
int b = a++;
System.out.println(a);
System.out.println(b);
a = 6;
b = ++a;
System.out.println(a);
System.out.println(b);
```