Chapter 2
Elementary Programming
COSC1046

Ping Luo

Office Hour

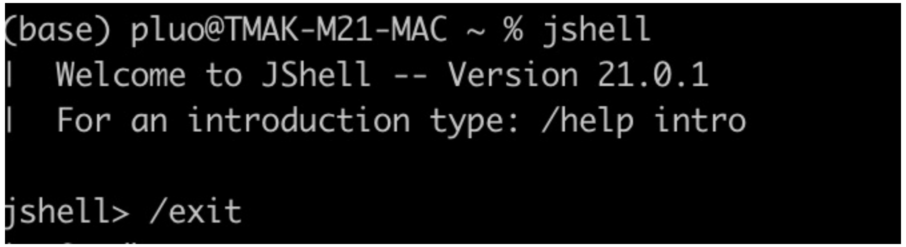
- Tuesday, A-605, 2:30-3:00pm

Objectives

- We will learn Java primitive data types and related subjects, such as variables, constants, data types, operators, expressions, and input and output.

JShell

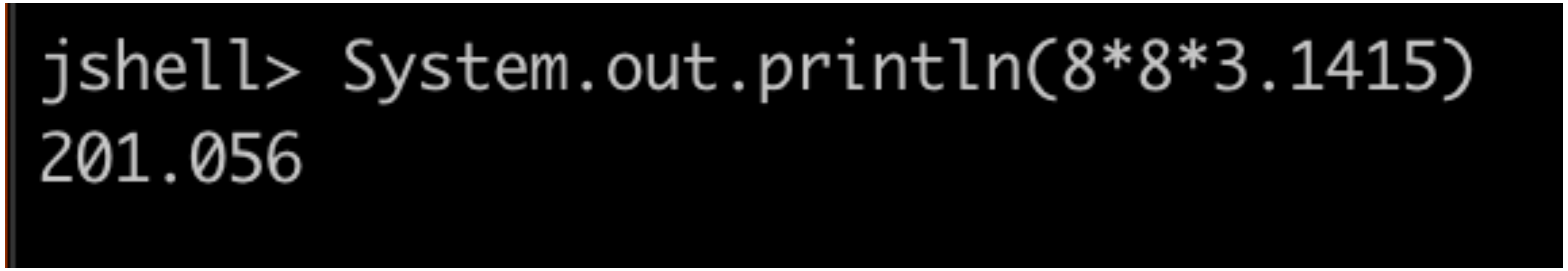
- JShell is a command line interactive tool introduced in Java 9.
- JShell enables you to type a single Java statement and get it executed to see the result right away without having to write a complete class. This feature is commonly known as REPL (Read-Evaluate-Print Loop), which evaluates expressions and executes statements as they are entered and shows the result immediately.



```
(base) pluo@TMAK-M21-MAC ~ % jshell
|  Welcome to JShell -- Version 21.0.1
|  For an introduction type: /help intro

jshell> /exit
```

Elementary Programming

- Computing the Area of a Circle



```
jshell> System.out.println(8*8*3.1415)
201.056
```

Input the radius of the circle: 8
The area of the circle is: 201.056

Variables

- A variable is a specific type of identifier that represents a storage location in the computer's memory. It is used to store and manipulate data.
- Variables have a data type that specifies what kind of values they can hold (e.g., integer, floating-point, string).
- The data stored in a variable can be changed during the execution of the program.

Declaring Variables

- We need to declare variables before using them.

- int x;   // Declare x to be an
-                           // integer variable;
- double radius;   // Declare radius to
-                           // be a double variable;
- char a;   // Declare a to be a
-                           // character variable;

Variables

- Once variable is declared, the area can be computed as follows:

```
radius = 1.0;
area = radius * radius * 3.14159;
System.out.println("The area is “ + area + " for radius "+radius);
```

Declaring and Initializing in One Step

- int x = 1;
- double d = 1.4;

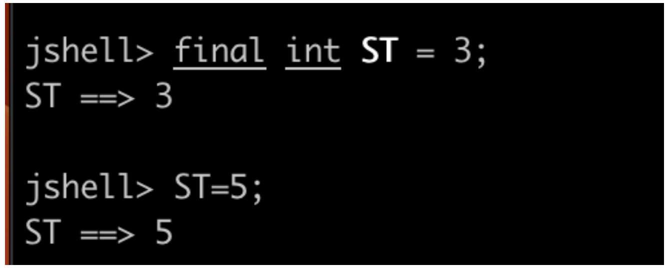
Copyright © 2024 by Ping Luo, Brampton, Ontario

Constants

- Constants cannot be changed once declared

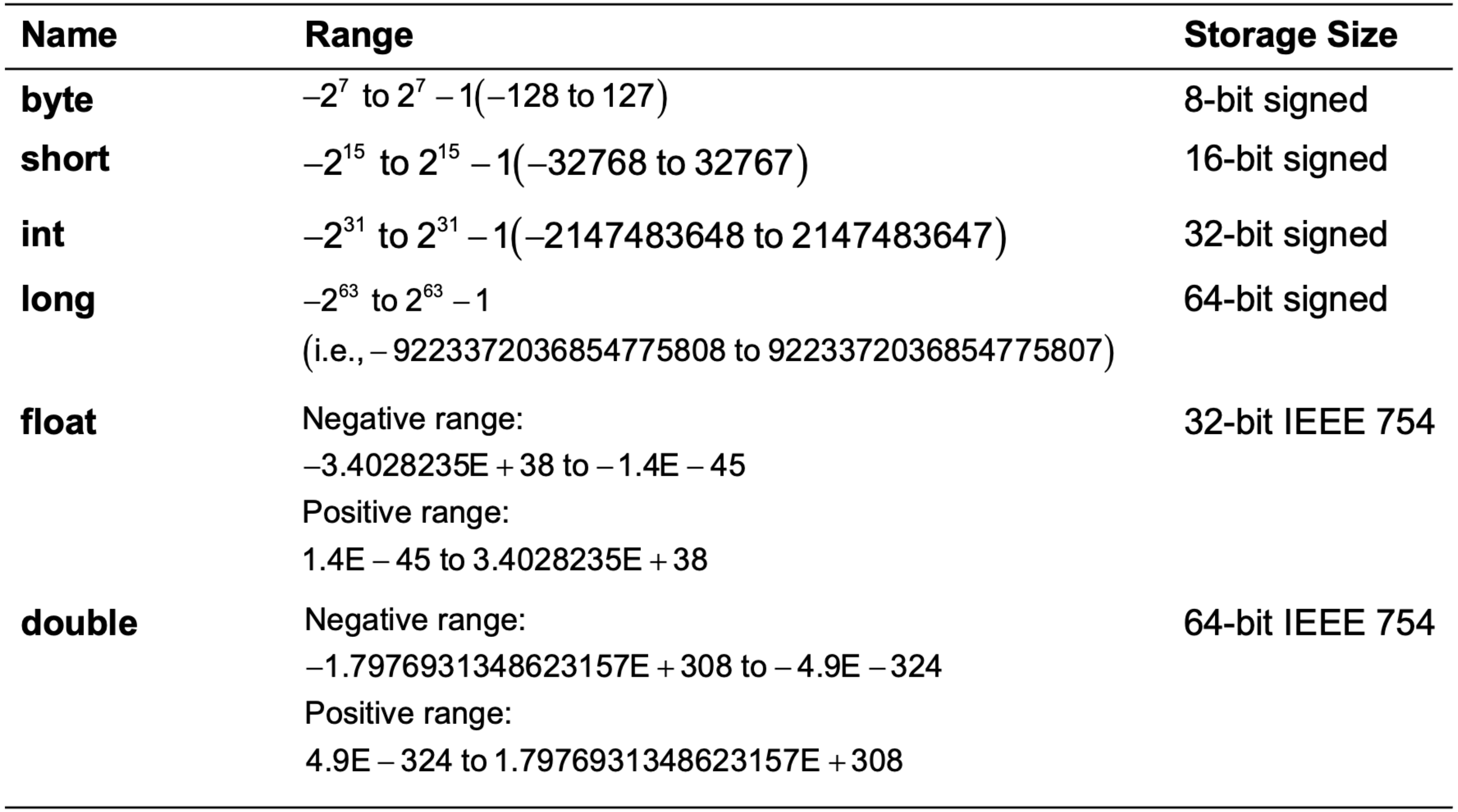
```
final datatype CONSTANTNAME = VALUE;
final double PI = 3.14159;
final int SIZE = 3;
```

Jshell doesn’t Support Constants



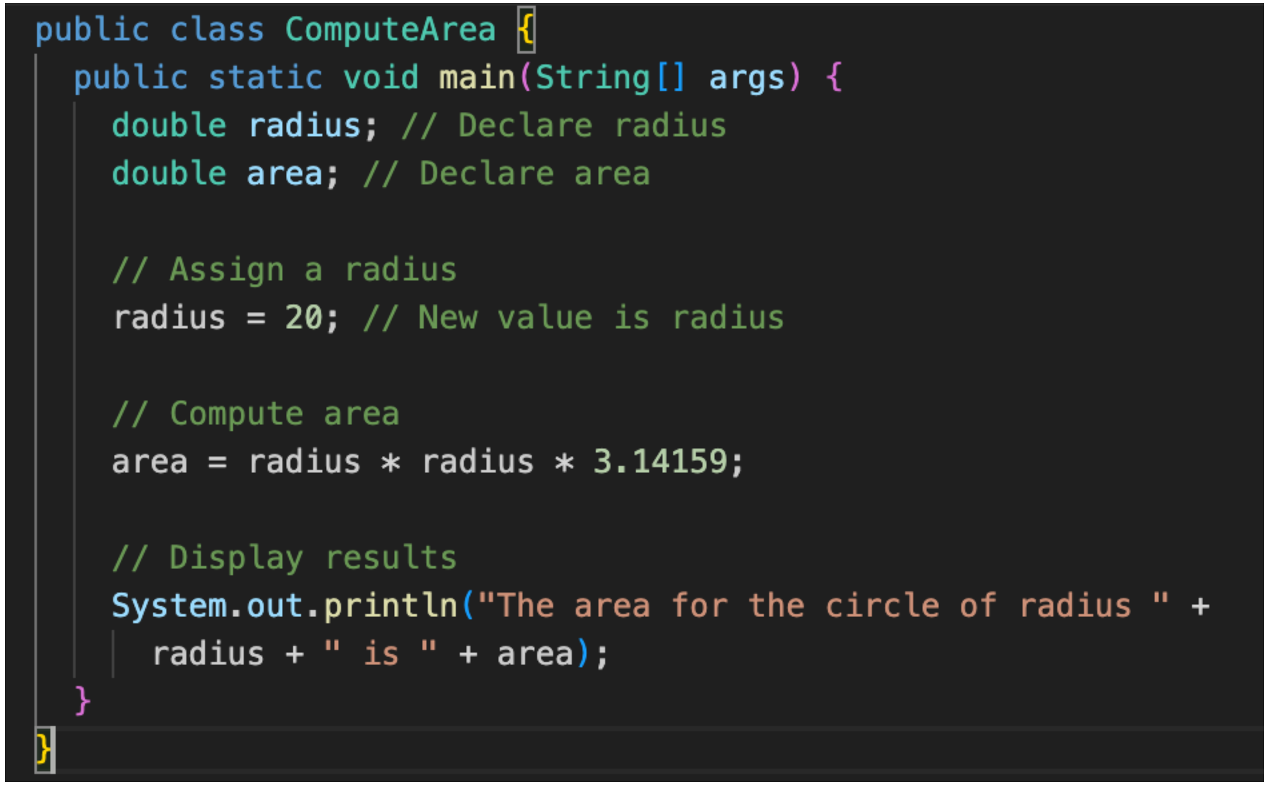
```
jshell> final int ST = 3;
ST ==> 3

jshell> ST=5;
ST ==> 5
```

Numerical Data Types



Name	Range	Storage Size
byte	-2^7 to $2^7-1(-128 \text{ to } 127)$	8-bit signed
short	-2^{15} to $2^{15}-1(-32768 \text{ to } 32767)$	16-bit signed
int	-2^{31} to $2^{31}-1(-2147483648 \text{ to } 2147483647)$	32-bit signed
long	-2^{63} to $2^{63}-1$ $(\text{i.e.}, -9223372036854775808 \text{ to } 9223372036854775807)$	64-bit signed
float	Negative range: $-3.4028235\text{E}+38$ to $-1.4\text{E}-45$ Positive range: $1.4\text{E}-45$ to $3.4028235\text{E}+38$	32-bit IEEE 754
double	Negative range: $-1.7976931348623157\text{E}+308$ to $-4.9\text{E}-324$ Positive range: $4.9\text{E}-324$ to $1.7976931348623157\text{E}+308$	64-bit IEEE 754

Problem: Compute Area of a Circle

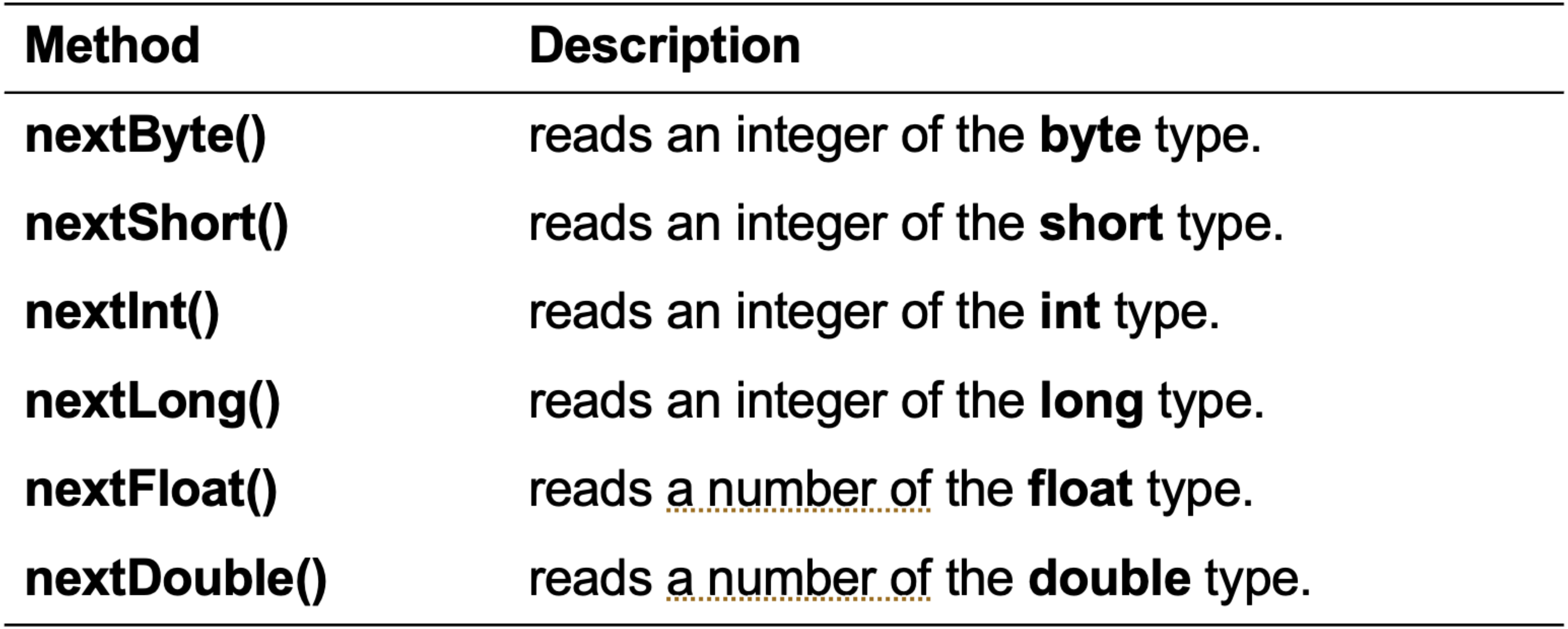


```java
public class ComputeArea {
  public static void main(String[] args) {
    double radius; // Declare radius
    double area; // Declare area

    // Assign a radius
    radius = 20; // New value is radius

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

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

Reading Input From the Keyboard

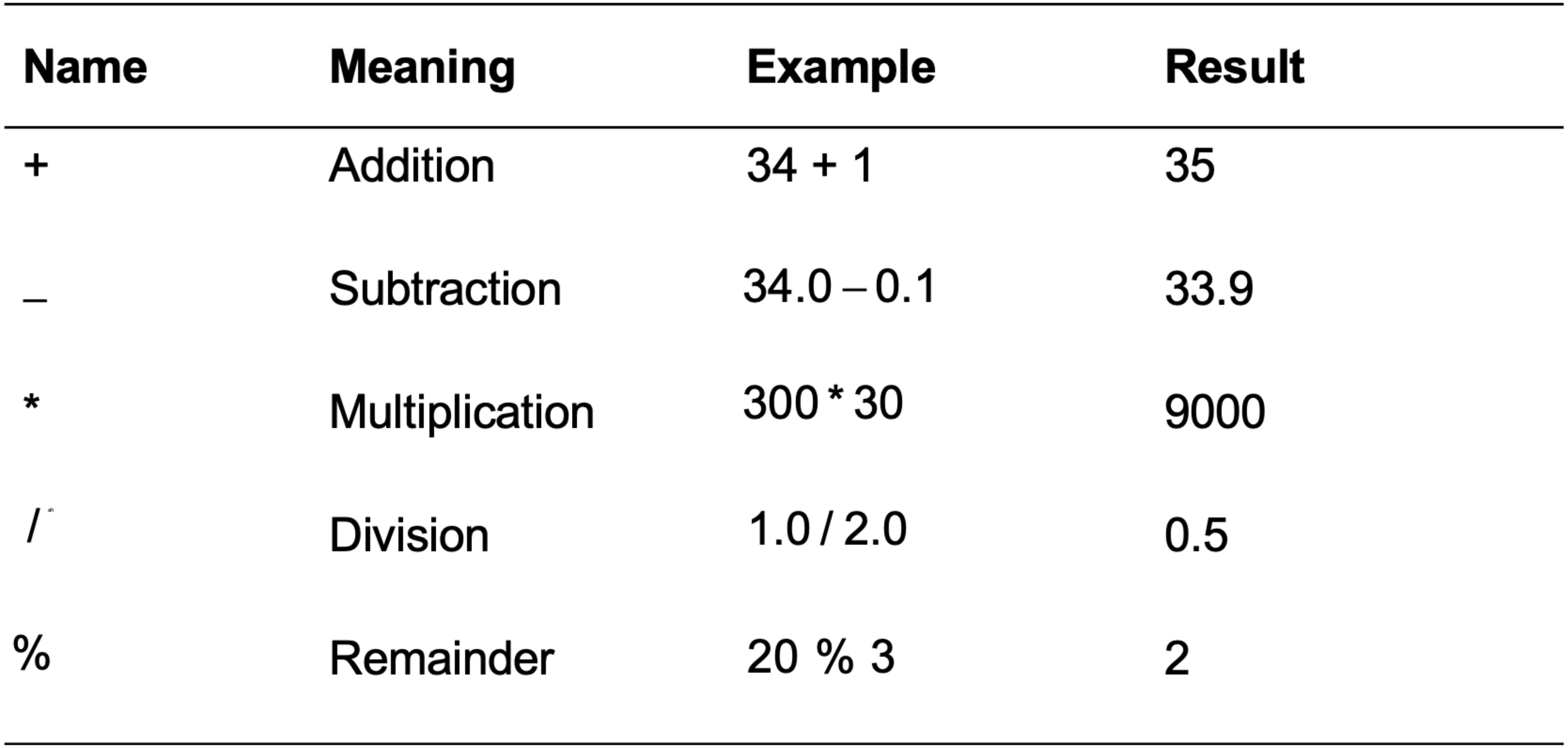
- Import Scanner
  - import java.util.Scanner;
- Create a Scanner object
  - Scanner input = new Scanner(System.in);
- Use nextDouble() to obtain a double value.
  - double d = intpu.nextDouble();

Reading Numbers 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.

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

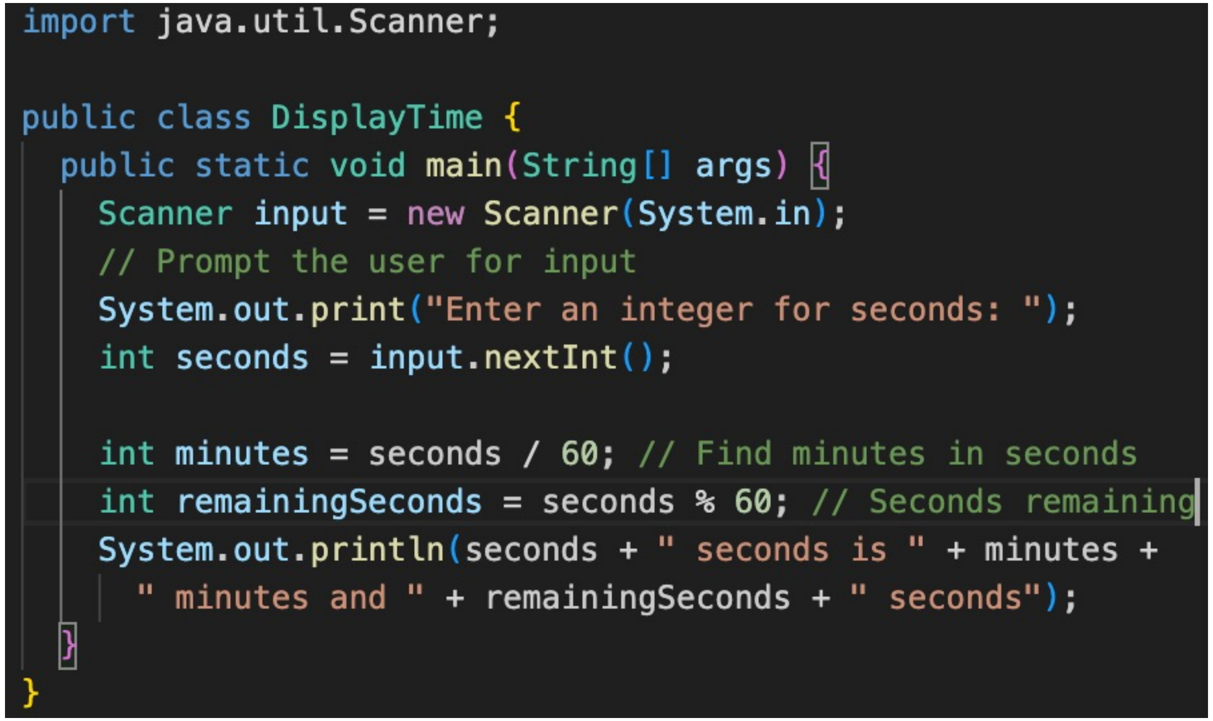
Integer Division

- 7/2 yields an integer 3;
- 7.0/2 yields a double value 3.5;
- 7%2 yields 1 (the remainder of the division).

Exponent Operations

- 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

Problem: Displaying Time



```java
import java.util.Scanner;

public class DisplayTime {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        // Prompt the user for input
        System.out.print("Enter an integer for seconds: ");
        int seconds = input.nextInt();

        int minutes = seconds / 60; // Find minutes in seconds
        int remainingSeconds = seconds % 60; // Seconds remaining
        System.out.println(seconds + " seconds is " + minutes +
          " minutes and " + remainingSeconds + " seconds");
    }
}
```

Note

- Calculations involving floating-point numbers are approximated because these numbers are not stored with complete accuracy.
- System.out.println(1.0 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1);
- displays 0.5000000000000001, not 0.5, and
- System.out.println(1.0 - 0.9);
- displays 0.09999999999999998, not 0.1.
- Integers are stored precisely. Therefore, calculations with integers yield a precise integer result.

Number Literals

- A literal is a constant value that appears directly in the program. For example, 34, 1,000,000, 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.
- 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.
- You can use 100.2f or 100.2F for a float number, and 100.2d or 100.2D for a double number.

Double versus Float

- The double type values are more accurate than the float type values.
- For example, System.out.println("1.0 / 3.0 is " + 1.0 / 3.0);

$$1.0 / 3.0 \text{ is } 0.\underbrace{3333333333333333}_{\text{16 digits}}$$

- System.out.println("1.0F / 3.0F is " + 1.0F / 3.0F);

$$1.0F / 3.0F \text{ is } 0.\underbrace{3333333}_{\text{7 digits}}4$$

Scientific Notation

- Floating-point literals can also be specified in scientific notation, for example, 1.23456e+2, same as 1.23456e2, is equivalent to 123.456, and 1.23456e-2 is equivalent to 0.0123456.

- E (or e) represents an exponent and it can be either in lowercase or uppercase.

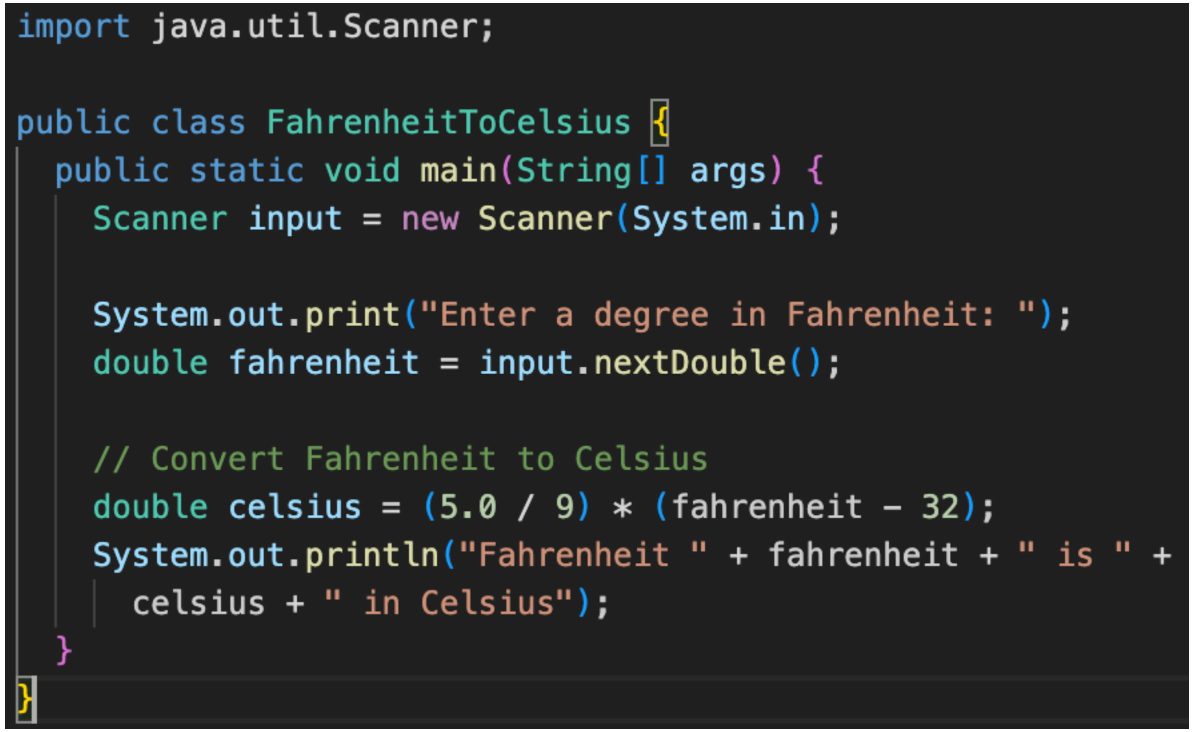
Arithmetic Expressions

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

is translated to

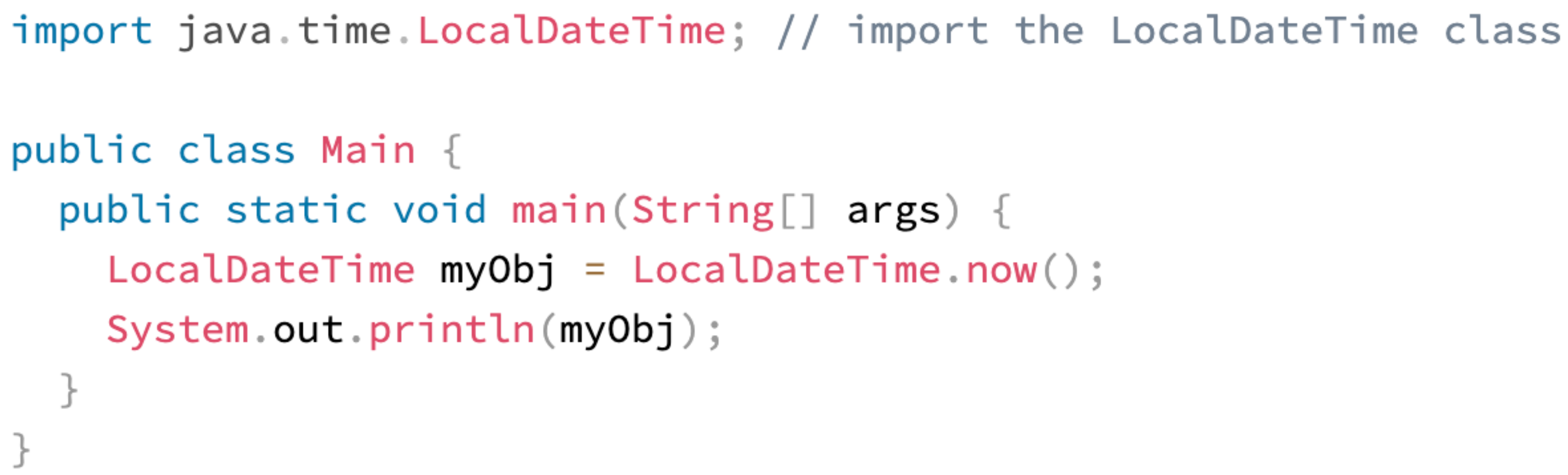
$$\left(3+4*x\right)/5-10*\left(y-5\right)*\left(a+b+c\right)/x+9*\left(4/x+\left(9+x\right)/y\right)$$

Problem: Converting Temperatures

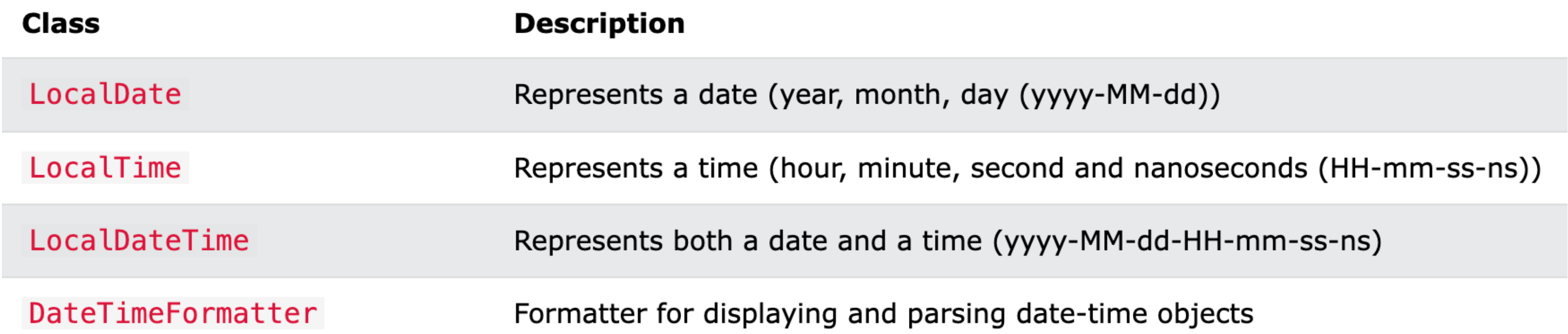


```java
import java.util.Scanner;

public class FahrenheitToCelsius {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

    System.out.print("Enter a degree in Fahrenheit: ");
    double fahrenheit = input.nextDouble();

    // Convert Fahrenheit to Celsius
    double celsius = (5.0 / 9) * (fahrenheit - 32);
    System.out.println("Fahrenheit " + fahrenheit + " is " +
      celsius + " in Celsius");
  }
}
```

Problem: Displaying Current Time

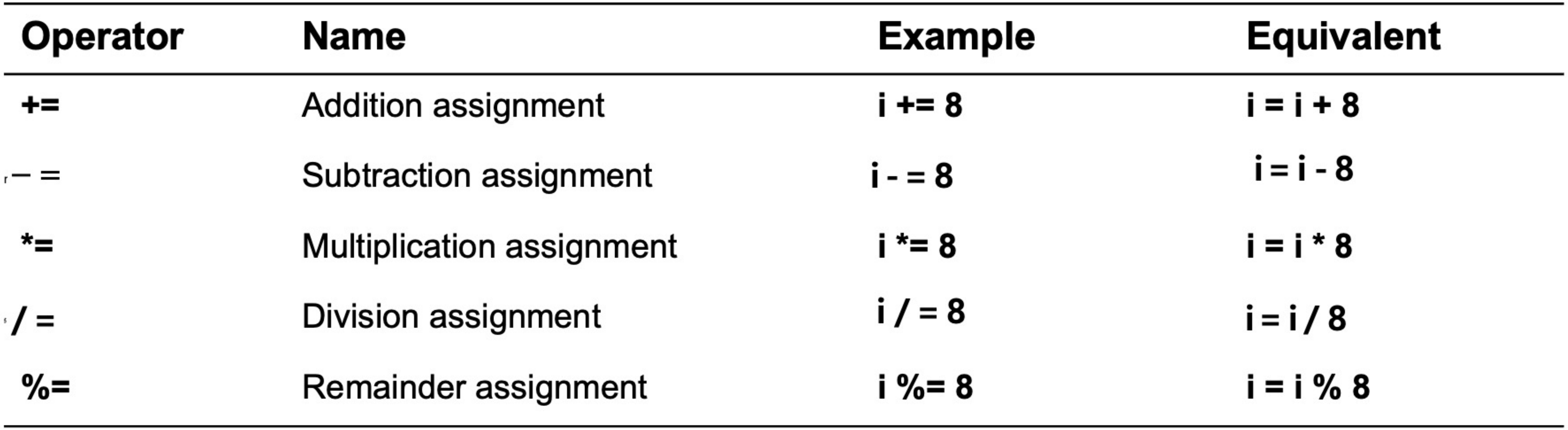


```java
import java.time.LocalDateTime; // import the LocalDateTime class

public class Main {
  public static void main(String[] args) {
    LocalDateTime myObj = LocalDateTime.now();
    System.out.println(myObj);
  }
}
```



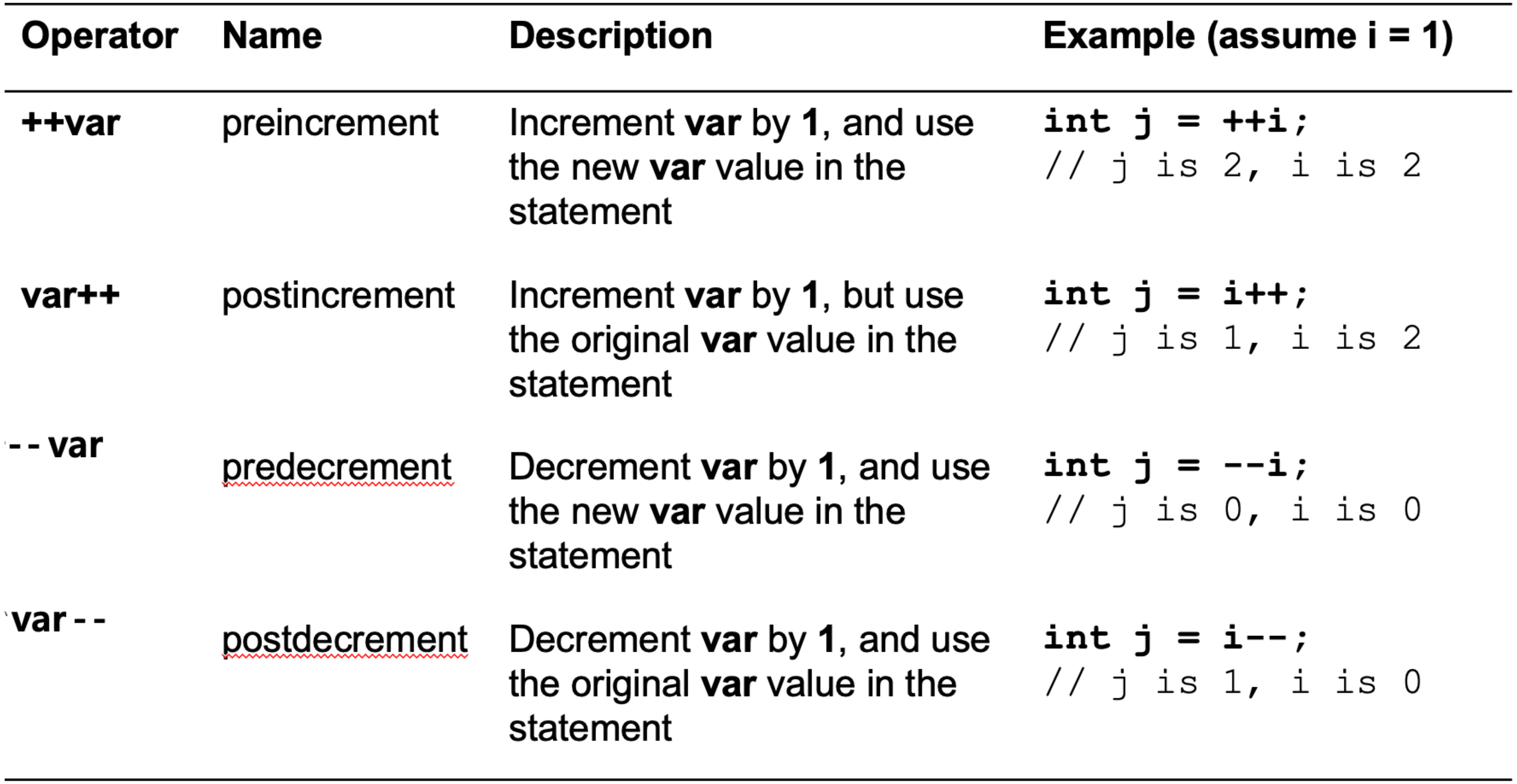
Class	Description
LocalDate	Represents a date (year, month, day (yyyy-MM-dd))
LocalTime	Represents a time (hour, minute, second and nanoseconds (HH-mm-ss-ns))
LocalDateTime	Represents both a date and a time (yyyy-MM-dd-HH-mm-ss-ns)
DateTimeFormatter	Formatter for displaying and parsing date-time objects

Augmented Assignment 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



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

- Using increment and decrement operators makes expressions short, but it also makes them complex and difficult to read.
- Avoid using these operators in expressions that modify multiple variables, or the same variable for multiple times such as this: int k = ++i + i.

Numeric Type Conversion

- When performing a binary operation involving two operands of different types, Java automatically converts the operand based on the following rules:

➢If one of the operands is double, the other is converted into double.

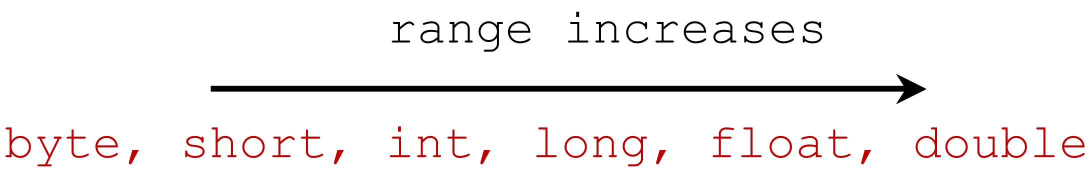
➢Otherwise, if one of the operands is float, the other is converted into float.

➢Otherwise, if one of the operands is long, the other is converted into long.

➢Otherwise, both operands are converted into int.

Type Casting

- double d = 3; (type widening)
- Explicit casting
- int i = (int)3.0; (type narrowing)
- int i = (int)3.9; (Fraction part is truncated)



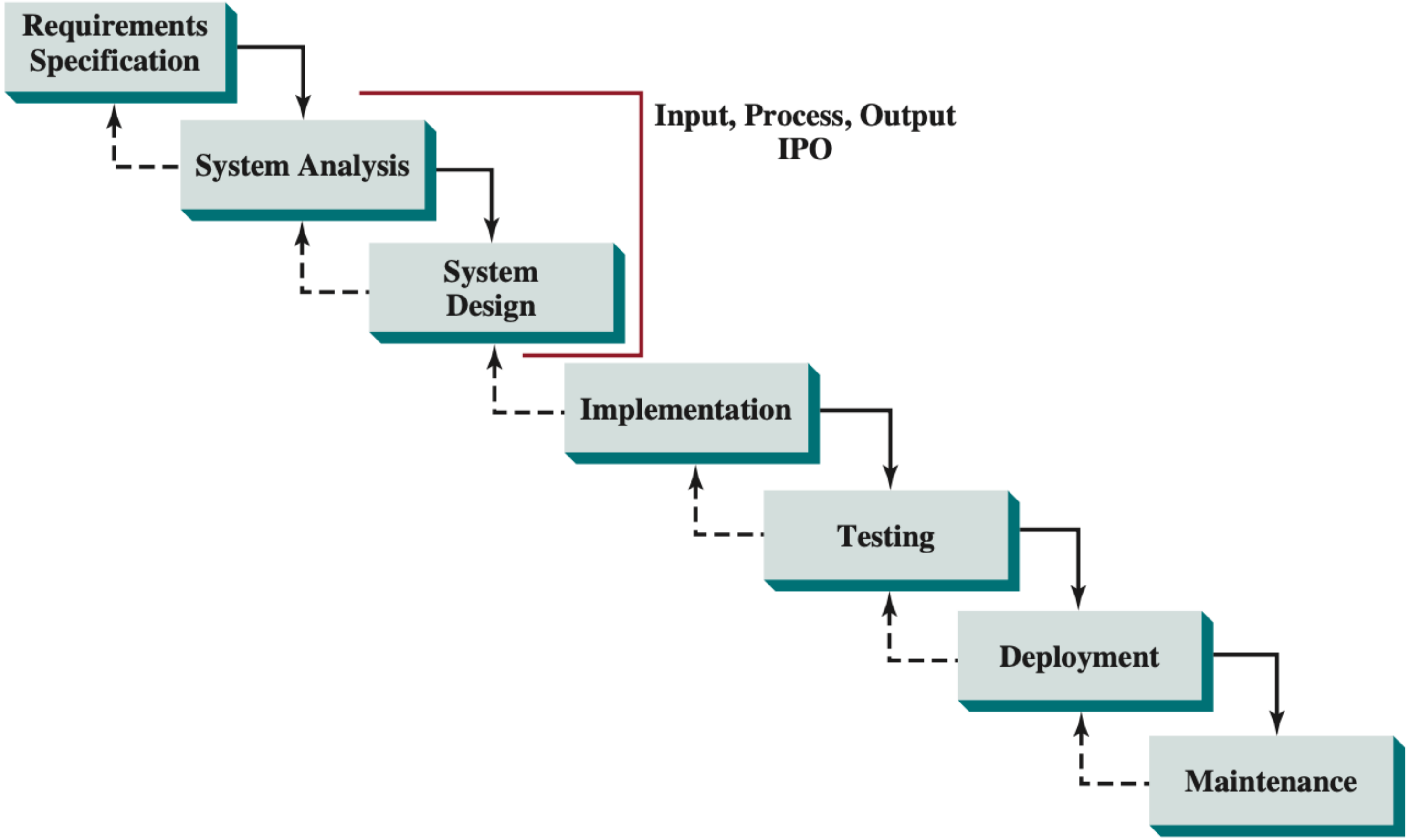
range increases

byte, short, int, long, float, double

Casting in an Augmented Expression

- In Java, an augmented expression of the form x1 op= x2 is implemented as x1 = (T)(x1 op x2), where T is the type for x1. Therefore, the following code is correct.
- int sum = 0;
- sum += 4.5
- sum += 4.5; // sum becomes 4 after this statement
- sum += 4.5 is equivalent to sum = (int)(sum + 4.5).

Software Development Process

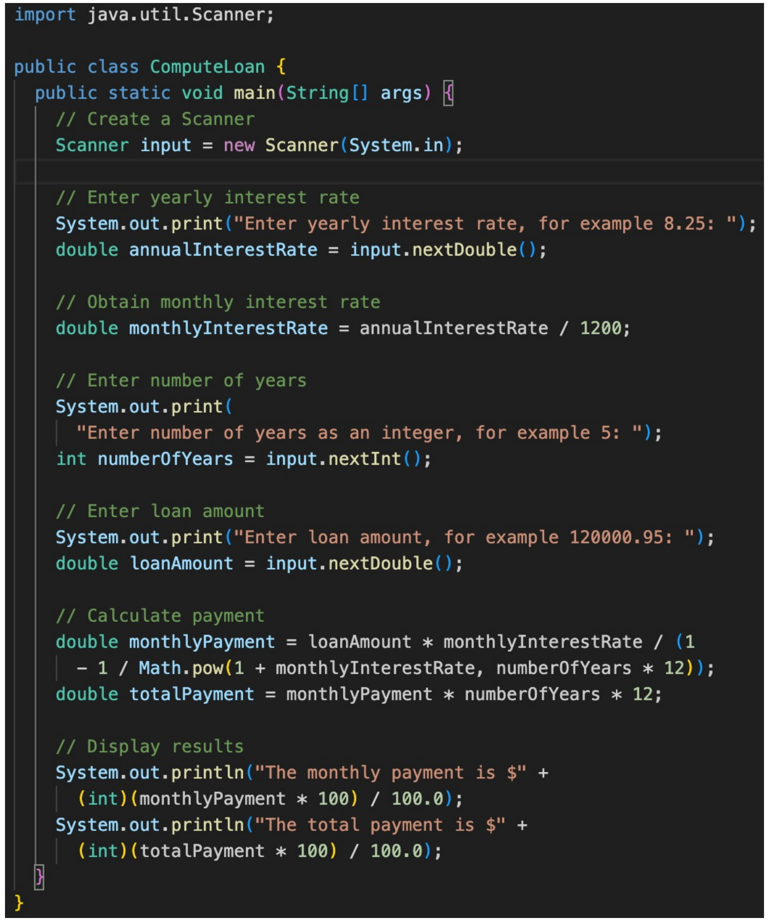


Problem: Computing Loan Payments

Formular:

$$monthlyPayment = \frac{loanAmount \times monthlyInterestRate}{1 - \frac{1}{(1 + monthlyInterestRate)^{numberOfYears \times 12}}}$$

Problem: Computing Loan Payments



```java
import java.util.Scanner;

public class ComputeLoan {
  public static void main(String[] args) {
    // Create a Scanner
    Scanner input = new Scanner(System.in);

    // Enter yearly interest rate
    System.out.print("Enter yearly interest rate, for example 8.25: ");
    double annualInterestRate = input.nextDouble();

    // Obtain monthly interest rate
    double monthlyInterestRate = annualInterestRate / 1200;

    // Enter number of years
    System.out.print(
      "Enter number of years as an integer, for example 5: ");
    int numberOfYears = input.nextInt();

    // Enter loan amount
    System.out.print("Enter loan amount, for example 120000.95: ");
    double loanAmount = input.nextDouble();

    // Calculate payment
    double monthlyPayment = loanAmount * monthlyInterestRate / (1
      - 1 / Math.pow(1 + monthlyInterestRate, numberOfYears * 12));
    double totalPayment = monthlyPayment * numberOfYears * 12;

    // Display results
    System.out.println("The monthly payment is $" +
      (int)(monthlyPayment * 100) / 100.0);
    System.out.println("The total payment is $" +
      (int)(totalPayment * 100) / 100.0);
  }
}
```