COSC 111

Computer Programming I

Elementary Programming

Dr. Abdallah Mohamed

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

Clicker Question

What is wrong with this code?

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

A. We can’t use plus (+) between a string (text) and a number

B. We can’t write a Java statement over two lines.

C. There is a missing semicolon (;)

D. There are extra spaces around the (+) operator

E. The statement is misspelled (e.g. capital vs small letters)

Clicker Question

What is wrong with this code?

```
public class Q{
   public static void main(String[] args) {
    //greetings
    system.out.println("Welcome"+ "t" + "o Java");;
   }
}
```

A. We can’t concatenate 3 strings (i.e. can’t use two (+)’s)

B. we must put the ‘t’ and ‘o’ together (i.e. “to”, not “t” + “o…”)

C. There is an extra semicolon (;)

D. The statement is misspelled (e.g. capital vs small letters)

E. Something else

Clicker Question

A program is supposed to print the numbers from 1 to 10. It actually prints the numbers from 0 to 9. What type of error is it?

A. A syntax error

B. A compilation error

C. A fatal runtime error

D. A logic error

Outline

1) Variables, data types, and assignment

2) Reading input from the user

3) Named Constants

4) Numeric operations

5) Numeric Type Conversion

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

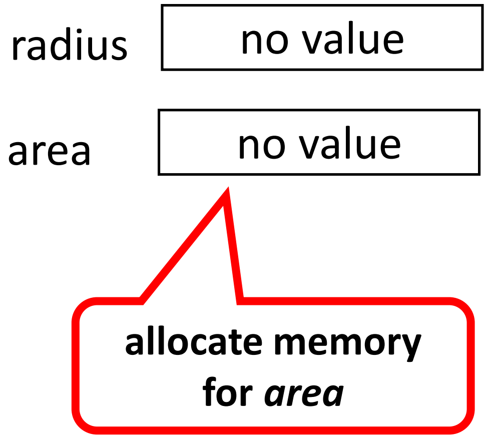
```
public class ComputeArea {
    /* Main method */
    public static void main(String[] args) {
        double radius;
        double area;

        // 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 variable
(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

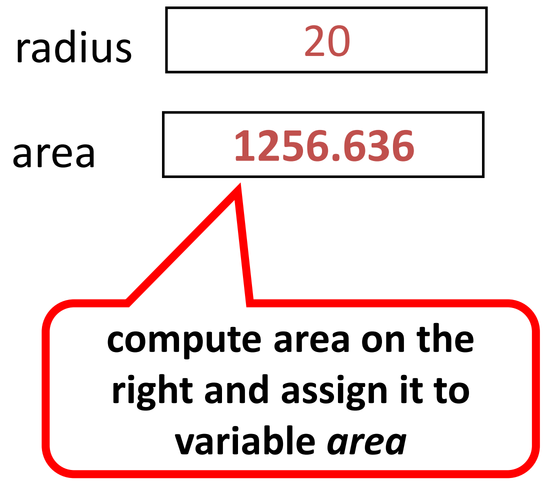
```
public class ComputeArea {
    /* Main method */
    public static void main(String[] args) {
        double radius;
        double area;

        // 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

radius	20
area	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);
    }
}
```



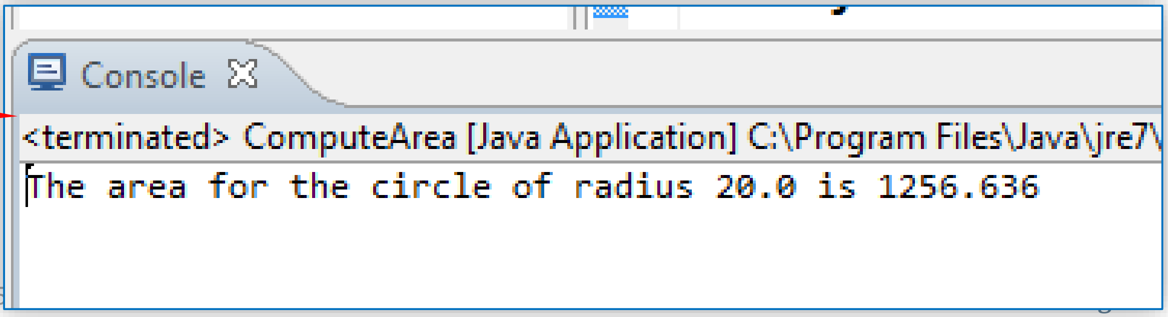
Trace a Program Execution

```java
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



Console

```
<terminated> ComputeArea [Java Application] C:\Program Files\Java\jre7\
The area for the circle of radius 20.0 is 1256.636
```

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
```

Advanced: Local-Variable Type Inference

<u>**Advanced**</u>

Local-Variable Type Inference (Java 10+)

- New feature in Java 10.
- Programmer can replace *Type Declarations* With `var` for ***local variable*** declarations ***with initializers***.
- The compiler infers the variable type from the right hand side of the declaration, i.e. the initializer.

  The type will be assigned to the variable will which can only store data of that type.

- Examples:

```
var x = 5;     // OK. x is of type int
var y = 5.0;   // OK. y is of type double
var s = "BC";  // OK. s is of type String

var z;         // ERROR. Compiler can’t infer the type
x = 1.2;       // ERROR. x was identified above as int
```

Where to use var? (Java 10+)

Note: this slide refers to topics not covered yet. Therefore, you can SKIP it for now and use it for reference in the future

var CAN only be used in the following cases:

- Local variable declarations with initializers

```
public void m(){var x = 5;}
```

- For loops

```
for(var i =1; i<10; i++){…}        for(var item: list){…}
```

- Try statements

```
try(var in=new FileInputStream(…)){…}
```

var CANNOT be used in other cases such as:

- global variables (instance variables).
- method signature

```
public var m(){…} // nope!        public void m(var a){…} // nope!
```

- Any case where the compiler cannot infer the type.

Advanced

Local-Variable Type Inference (Java 10+)

Note: this slide refers to topics not covered yet. Therefore, you can SKIP it for now and use it for reference in the future

More Examples:

```
// all this is OK
var input   = new Scanner(System.in);
var rob     = new Robot(2,4);
var list    = new ArrayList<Float>(3);
var address = new URL("https://ok.ubc.ca");

// all this is ERROR as compiler needs explicit target type
var nums    = {1,2,3};
var square  = a -> a * a;
```

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

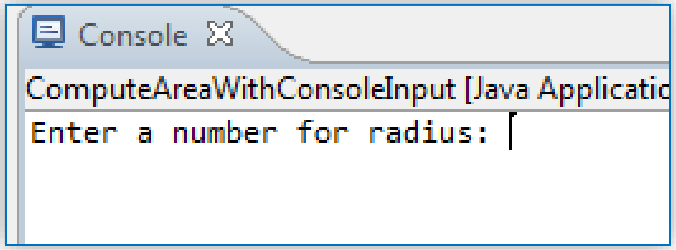
```
import java.util.Scanner; // Scanner is in the java.util package

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.



ComputeAreaWithConsoleInput [Java Applicatic

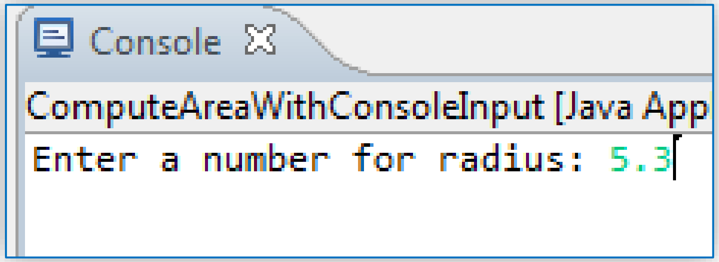
```
Enter a number for radius:
```

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



Console

```
ComputeAreaWithConsoleInput [Java App
Enter a number for radius: 5.3
```

Reading Input from the Keyboard

```java
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);
    }
}
```

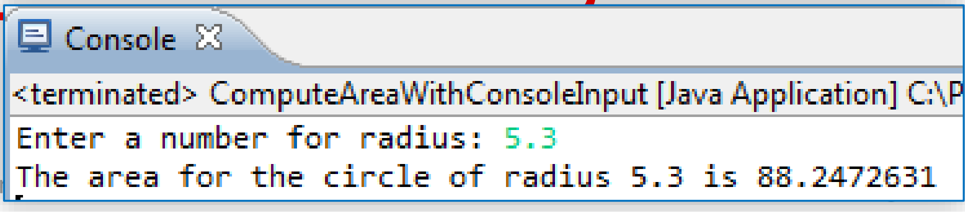
radius	5.3
area	88.2472631

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 $\boxed{5.3}$

area $\boxed{88.2472631}$



Console

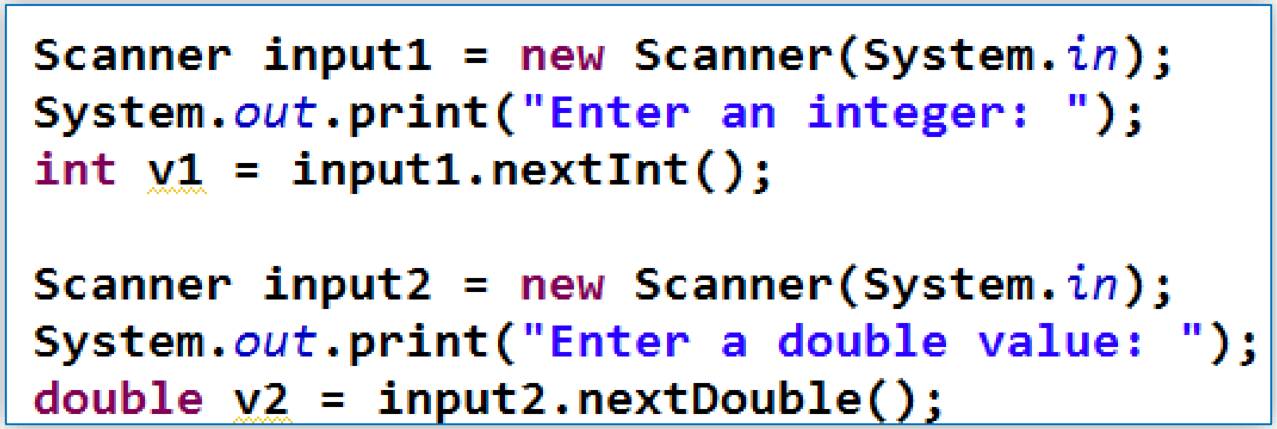
```
<terminated> ComputeAreaWithConsoleInput [Java Application] C:\P
Enter a number for radius: 5.3
The area for the circle of radius 5.3 is 88.2472631
```

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.

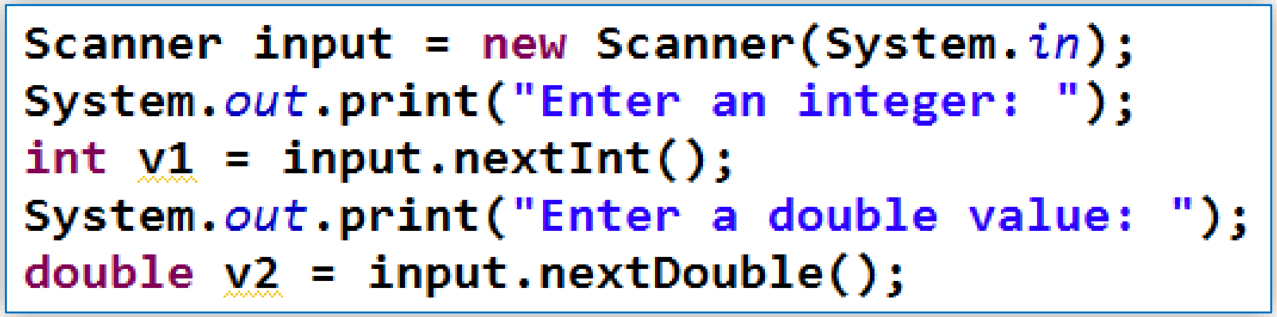
Redundant Input Objects

This code is not wrong, BUT inefficient!



```java
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



```java
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();
```

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

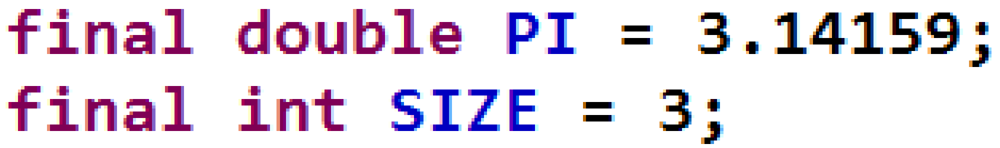
- Write a program that asks the user about his/her name and then display a simple greeting message.

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

Named Constants

Named Constants

A constant must be declared and initialized in the same statement.



```
final double PI = 3.14159;
final int SIZE = 3;
```

Named Constants - Example

```java
public class ComputeAreaWithConstant {
    public static void main(String[] args) {
        final double PI = 3.14159; // Declare a constant

        // 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 * PI;

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

Naming Conventions

Variables and method names:

- Use lowercase. If the name consists of several words, concatenate all in one and capitalize the first letter of all words after the first one.
  - Example: radius, area., computeArea().

Class names:

- Capitalize the first letter of each word in the name.
  - Example, ComputeArea.

Constants:

- Capitalize all letters in constants, use underscores for multiple words.
  - Example: PI and MAX_VALUE

Numeric operations

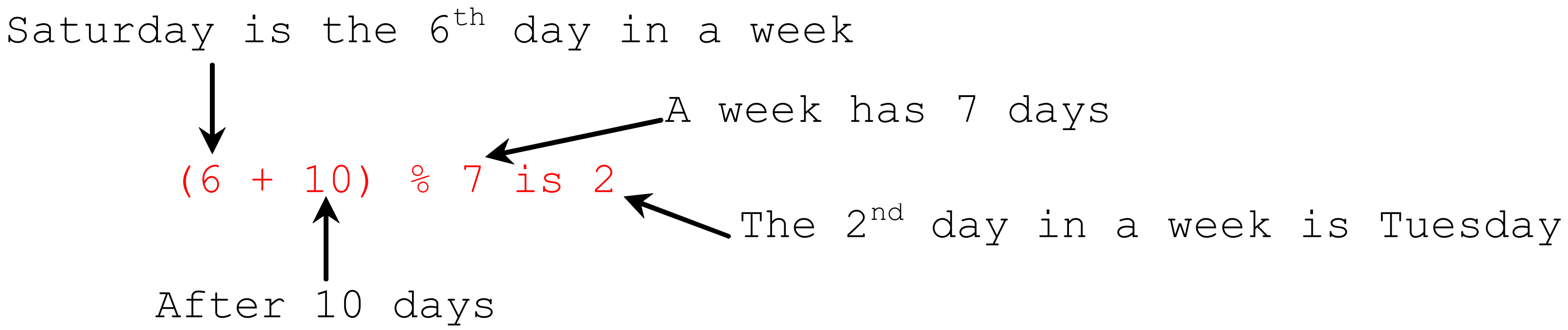
Numeric Operators

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


Saturday is the 6th day in a week
A week has 7 days
(6 + 10) % 7 is 2
The 2nd day in a week is Tuesday
After 10 days

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.

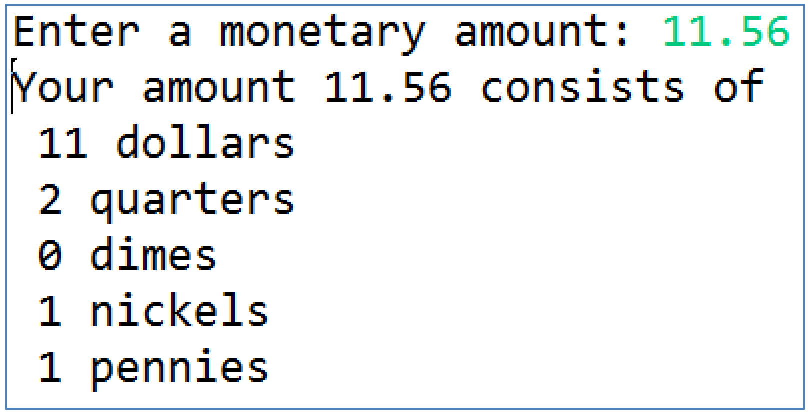
Problem: Monetary Units

solution

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 $\mathbf{a}^{\mathbf{b}}$

```
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

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

is translated to

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

Practice

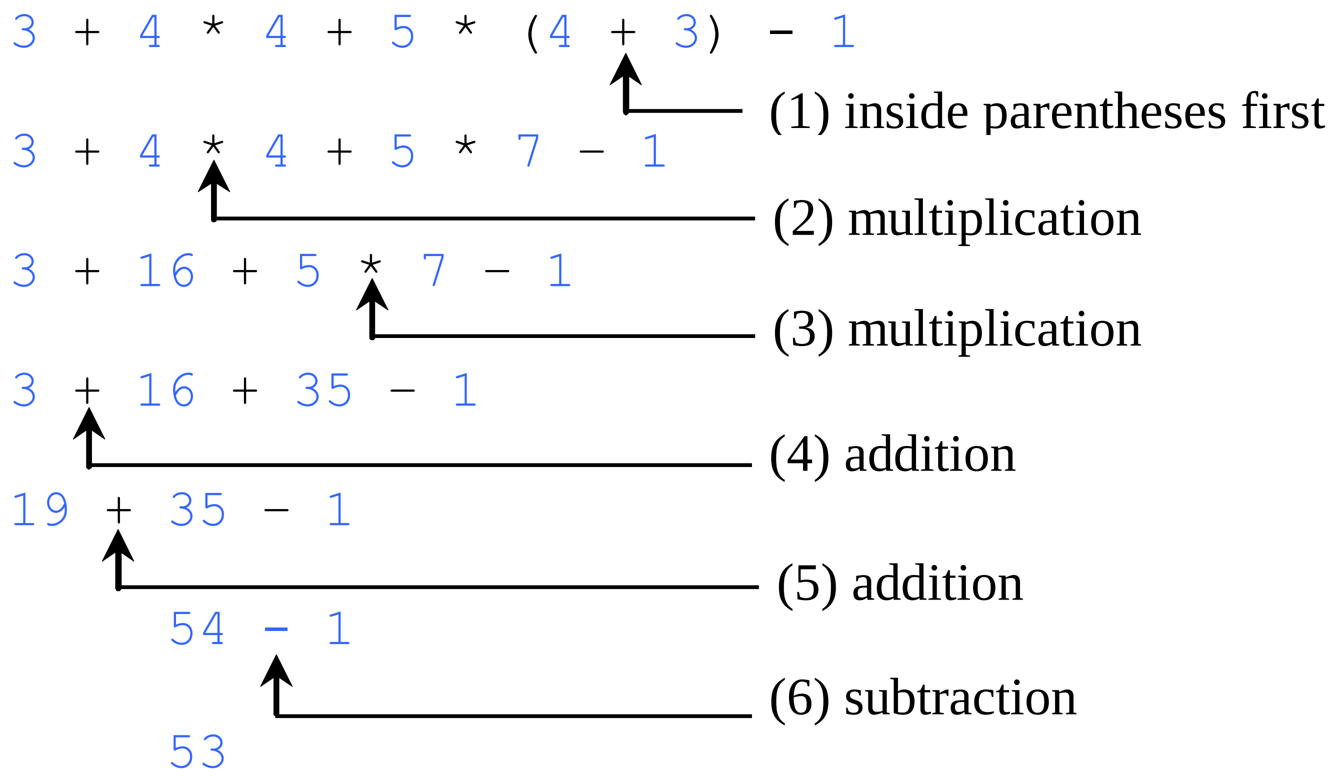
How would you write the following arithmetic expression in Java?

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

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

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:

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

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

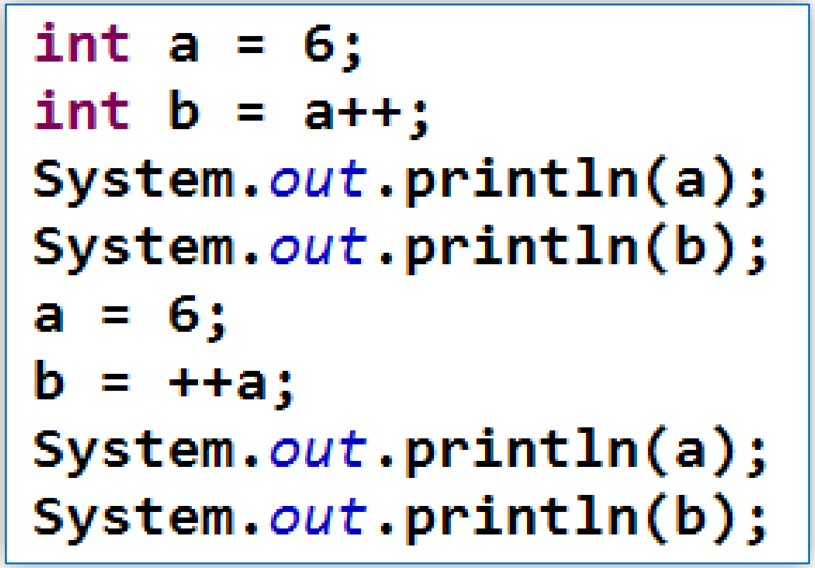
D. 7 7
   7 6

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:



```java
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);
```

Numeric Type Conversion

Numeric Type Conversion

Implicit Casting: You can always assign a value to a numeric variable whose type supports a larger range of values.

- `double x = 5; // no error (type widening)`

$$\begin{tikzcd}
\boxed{\text{byte}} \arrow[r] & \boxed{\text{short}} \arrow[r] & \boxed{\text{int}} \arrow[r] & \boxed{\text{long}} \arrow[r] & \boxed{\text{float}} \arrow[r] & \boxed{\text{double}} \\
 & & \boxed{\text{char}} \arrow[u] & & &
\end{tikzcd}$$

Explicit Casting: You cannot assign a value to a variable of a type with a smaller range unless you use ***type casting***.

- `int y = 3.5; // compilation error`
- `int z = (int) 3.5; // z = 3(type narrowing)`

-
  ```
  double d = 7.61;
  int n = (int) d; // n = 7. no change to d
  ```

Type Casting

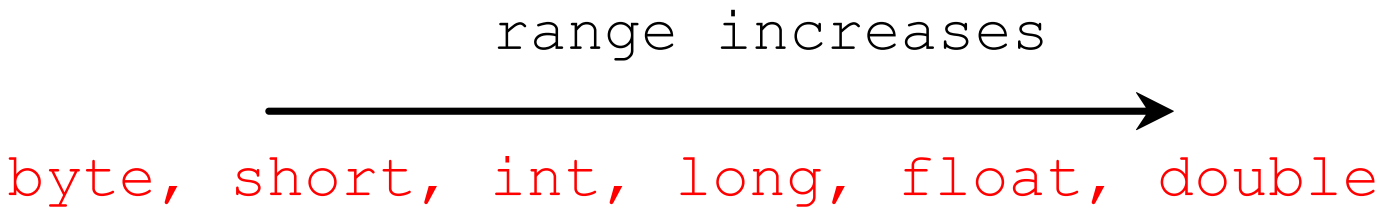
Implicit casting

- double d = 3; (type widening)

Explicit casting

- int i = (int)3.0; (type narrowing)
- int i = (int)3.9; (Fraction part is truncated)

What is wrong? int x = 5 / 2.0;



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 becomes 4
  ```

  //sum += 4.5 is equivalent to sum = (int)(sum + 4.5).

Conversion Rules

When performing a binary operation involving two operands of different numeric 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.

Question:

- What is wrong with this statement?

int x = 5 / 2.0;

Practice

A) What is the output of:

```
float f = 12.5F;
int i = (int)f;
System.out.println("f is " + f);
System.out.println("i is " + i);
```

B) What is the output of:

```
double x = 5.5;
System.out.println( (int)(x * 4) / 3 );
System.out.println( (int)(x * 4) / 3.0 );
System.out.println( (int)(x * 4 / 3) );
System.out.println( (int)(x * 4 / 3.0) );
System.out.println( (int) x * 4 / 3 );
System.out.println( (int) x * 4 / 3 );
```