ÖZYEĞİN UNIVERSITY

CS105
Introduction to Object-Oriented Programming

Prof. Dr. Nizamettin AYDIN

naydin@itu.edu.tr

nizamettin.aydin@ozyegin.edu.tr

DATA TYPES

Outline

- Primitive Data Types
- Java's Primitive Data Types
- Expressions
- Arithmetic Operators
- Precedence
- Mixing Types
- String Concatenation
- Variables

PRIMITIVE DATA TYPES

- **type**:
  - –A category or set of data values
  - –Any type of information including, although not limited to,
    - numeric data, logical data, text and objects
  - –Constrains the operations that can be performed on data
  - –Many languages ask the programmer to specify types
    - Examples: integer, real number, string
- Internally, computers store everything as 1s and 0s

  104 ➔ 01101000

  "hi" ➔ 0110100110101
- Data is stored in the form of variables
- We can view the purpose of a program as a means of doing some useful work on data

JAVA'S PRIMITIVE TYPES

- primitive types:
  - –8 simple types for numbers, text, etc.
  - –Java also has object types, which we'll talk about later

Type	Description	Size	Example value
`boolean`	True or false	1 bit	`true, false`
`byte`	Integer	1 byte (8 bits)	
`char`	Unicode character	2 bytes	`'a', '\u0030'`
`short`	Integer	2 bytes	`-3, -2, -5`
`int`	Integer	4 bytes	`-3, -2, -5`
`long`	Integer	8 bytes	`-3L, 0L, 4L`
`float`	Floating point	4 bytes	`1.2f, -1.2e03f`
`double`	Floating point	8 bytes	`1.2, -1.2e03`

JAVA'S PRIMITIVE TYPES

- In Java, all numeric types are signed,
  - meaning that they can take on positive and negative values
    - there is no distinction between signed and unsigned types as there is in languages such as C
- Primitive data is stored in the form of variables.
  - To use a variable, we must declare it first.
    - This ensures that the compiler knows how much memory to set aside to store each variable.
- Java is a strongly typed language,
  - meaning that we must always state what kind of data something is before we can use it.
  - This declaration happens only once.
- The primitive data type keywords start with lower case letters to remind us that they do not have the status of a class.
- Primitive variables also have default values
  - 0 for the numeric ones and false for boolean ones.

JAVA'S PRIMITIVE TYPES

- some example of primitive variables being declared and then given some values:

```
int x;
boolean y;
double x1;
float x2;
char myLetter;

x = 3;
y = true;
x1 = 1.5;
x2 = 6.5f;
myLetter = 'x';
```

EXPRESSIONS

- can combine a declaration with setting an initial value:

```
int x = 3;
boolean y = false;
```

- **expression:**
  - A value or operation that computes a value.
  - Examples:

    1 + 4 * 5

    (7 + 2) * 6 / 3

    42
  - The simplest expression is a literal value.
  - A complex expression can use operators and parentheses.

EXPRESSIONS

- can manipulate variable values using an expression:

```
x = 5;
x = x + 2;      // Add 2 to the value of x
int z = 2;
x = z + 2;
x += 4;         // Add 4 to the value of x
x++;            // Increase x by 1
x--;            // Decrease x by 1
x = x * 6;      // Multiply x by 6
x = x / 2;      // Integer divide x by 2
y = false;
x2 = x2 / 5.2f;
```

  - For an expression, the right-hand side of the equals sign is evaluated and used to set the variable on the left-hand side

ARITHMETIC OPERATORS

- **operator**:
  - –Combines multiple values or expressions.

`+`	addition
`-`	subtraction (or negation)
`*`	multiplication
`/`	division
`%`	modulus (a.k.a. remainder)

- As a program runs, its expressions are evaluated.
  - `1+1` evaluates to `2`
  - `System.out.println(3*4);` prints `12`
    - How would we print the text `3*4` ?

ARITHMETIC OPERATORS

- When we divide integers, the quotient is also an integer.
  - 14 / 4 is 3, not 3.5

```
     3          4          52
  4 ) 14     10 ) 45    27 ) 1425
      12         40         135
       2          5          75
                             54
                             21
```

- More examples:
  - 32 / 5 is 6
  - 84 / 10 is 8
  - 156 / 100 is 1
  - Dividing by 0 causes an error when your program runs.

INTEGER REMAINDER WITH %

- The **%** operator computes the remainder from integer division.
  - 14 % 4 is 2
  - 218 % 5 is 3

```
        3
    ------
4 ) 14
    12
    --
     2
```

```
        43
    ------
5 ) 218
    20
    --
     18
     15
     --
      3
```

What is the result?

45 % 6

2 % 2

8 % 20

11 % 0

- Applications of % operator:
  - Obtain last digit of a number : `230857 % 10` is `7`
  - Obtain last 4 digits : `658236489 % 10000` is `6489`
  - See whether a number is odd : `7 % 2` is `1`, `42 % 2` is `0`

PRECEDENCE

- **precedence**:
  - Order in which operators are evaluated.
  - Generally, operators evaluate left-to-right.

    1 - 2 - 3 is (1 - 2) - 3 which is -4
  - But * / % have a higher level of precedence than + -

    1 + **3 * 4** is 13

    6 + **8 / 2** * 3

    6 + **4 * 3**

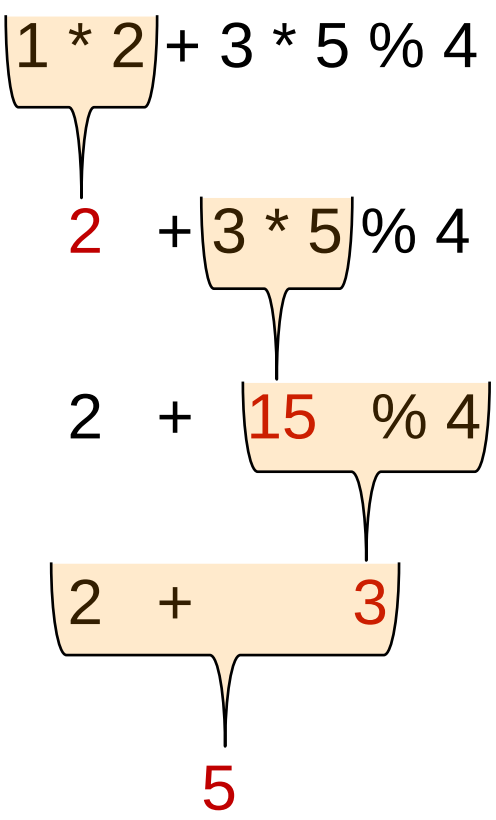
    6 + 12 is 18
  - Parentheses can force a certain order of evaluation:

    (**1 + 3**) * 4 is 16
  - Spacing does not affect order of evaluation

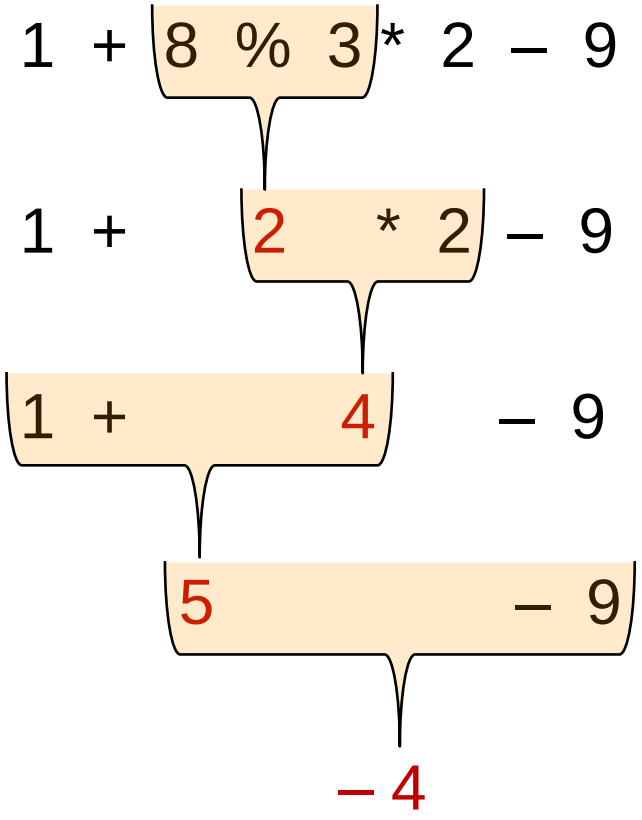
    1 + **3 * 4** - 2 is 11

PRESEDENCE EXAMPLE

- Example 1:



- Example 2:

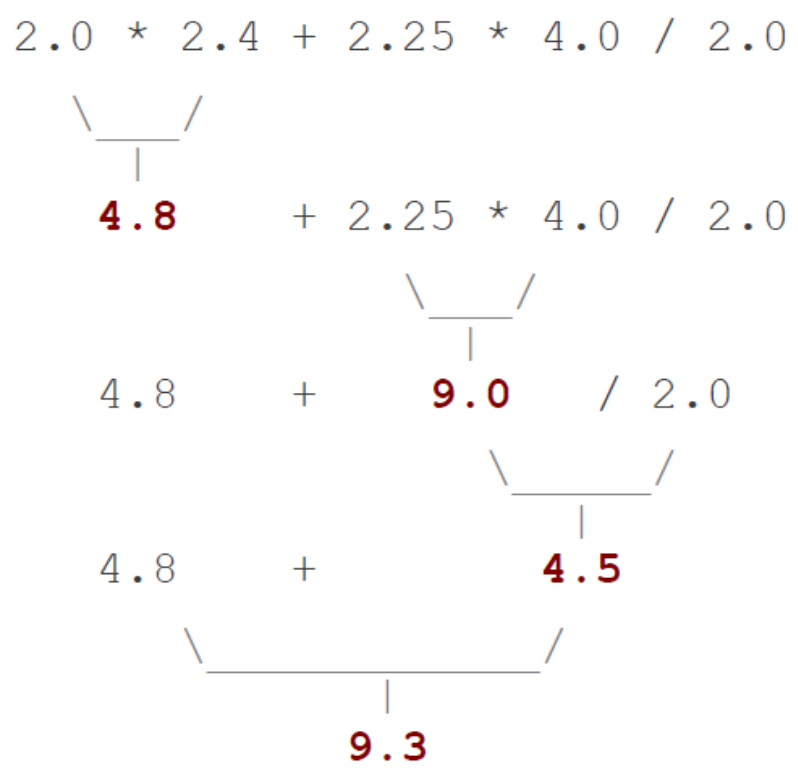


PRESEDENCE QUESTIONS

- What values result from the following expressions?
  - 9 / 5
  - 695 % 20
  - 7 + 6 * 5
  - 7 * 6 + 5
  - 248 % 100 / 5
  - 6 * 3 - 9 / 4
  - (5 - 7) * 4
  - 6 + (18 % (17 - 12))

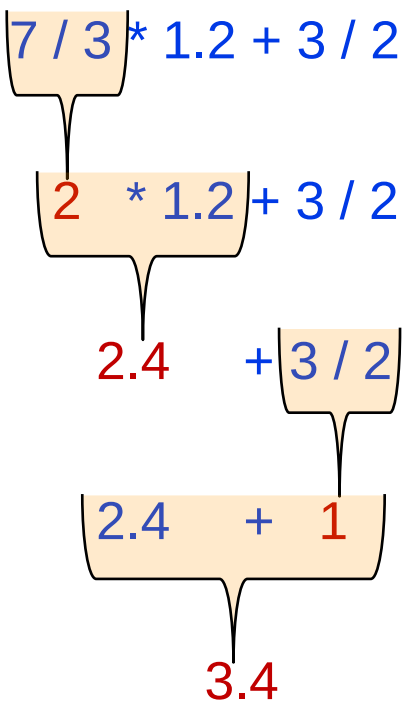
REAL NUMBERS (TYPE DOUBLE)

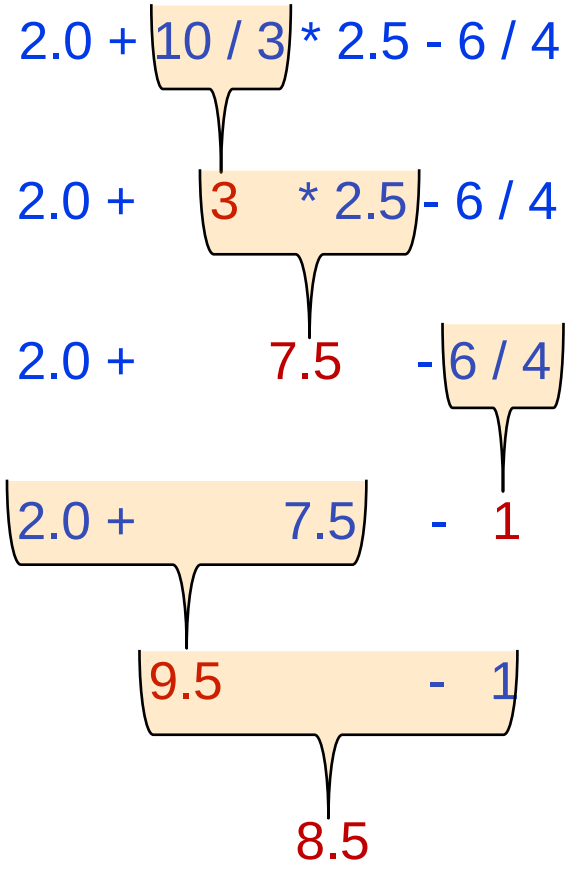
- Examples:
  - 6.022 , -42.0 , 2.143e17
    - Placing **.0** or **.** after an integer makes it a double.
- The operators **+ - * / % ()** all still work with double.
  - ***/*** produces an exact answer:
    - 15.0 / 2.0 is 7.5
- Precedence is the same:
  - () before * / % before + -
- Real number example:



MIXING TYPES

- When int and double are mixed, the result is a double.
  - 4.2 * 3 is 12.6
- The conversion is per-operator, affecting only its operands.





STRING CONCATENATION

- **string concatenation**:
  - –Using **+** between a string and another value to make a longer string.

"hello" + 42	is	"hello42"
1 + "abc" + 2	is	"1abc2"
"abc" + 1 + 2	is	"abc12"
1 + 2 + "abc"	is	"3abc"
"abc" + 9 * 3	is	"abc27"
"1" + 1	is	"11"
4 - 1 + "abc"	is	"3abc“

- Use **+** to print a string and an expression's value together.
  - – System.out.println("Grade: " + (95.1 + 71.9) / 2);
    - Output: Grade: 83.5

VARIABLES

RECEIPT EXAMPLE

- What's bad about the following code?

```
public class Receipt {
    public static void main(String[] args) {
       // Calculate total owed, assuming 8% tax / 15% tip
       System.out.print("Subtotal: ");
       System.out.println(38 + 40 + 30);
       System.out.print("Tax: ");
       System.out.println((38 + 40 + 30) * .08);
       System.out.print("Tip: ");
       System.out.println((38 + 40 + 30) * .15);
       System.out.print("Total: ");
       System.out.println(38 + 40 + 30 +
       (38 + 40 + 30) * .08 +
       (38 + 40 + 30) * .15);
    }
}
```

  - The subtotal expression `(38 + 40 + 30)` is repeated
  - So many `println` statements

VARIABLES

- **variable**:
  - –A piece of the computer's memory that is given a name and type,
    - can store a value.
    - Like preset stations on a car stereo, or cell phone speed dial:

  - –Steps for using a variable:
    - Declare it –
      - –state its name and type
    - Initialize it –
      - –store a value into it
    - Use it –
      - –print it or use it as part of an expression

DECLARATION

- variable declaration:
  - Sets aside memory for storing a value.
    - Variables must be declared before they can be used.

- Syntax:

  type name;

  - The name is an identifier.

  - int x;

  - double myGPA;

ASSIGNMENT

- **assignment**:
  - –Stores a value into a variable.
    - The value can be an expression;
      - – the variable stores its result.

- Syntax:

  name = expression;

  - int x;

    x = 3;

  - double myGPA;

    myGPA = 1.0 + 2.25;

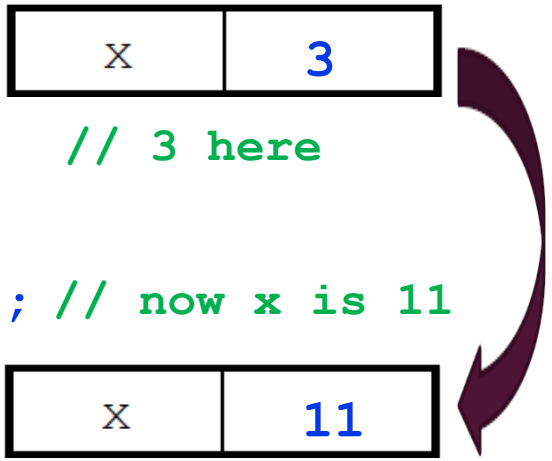
USING VARIABLES

- Once given a value, a variable can be used in expressions:

```
int x;
x = 3;
System.out.println("x is " + x);  // x is 3
System.out.println(5 * x - 1);    // 14
```

- You can assign a value more than once:

```
int x;
x = 3;
System.out.println(x + " here");      // 3 here
x = 4 + 7;
System.out.println("now x is " + x); // now x is 11
```



USING VARIABLES

- Once given a value, a variable can be used in expressions:

```
int x;
x = 3;
System.out.println("x is " + x); // x is 3
System.out.println(5 * x - 1);   // 14
```

- You can assign a value more than once:

```
int x;
x = 3;
System.out.println(x + " here");
x = 4 + 7;
System.out.println("now x is " + x);
```

DECLARATION/INITIALIZATION

- A variable can be declared/initialized in one statement.
- Syntax:

type name = value;

- `double myGPA = 3.95;`

myGPA	3.95

- `int x = (11 % 3) + 12;`

x	14

ASSIGNMENT AND ALGEBRA

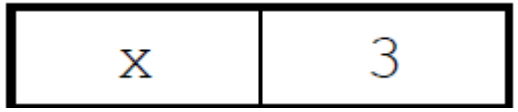
- Assignment uses **=** , but it is not an algebraic equation.

  = means, "store the value at right in variable at left"

  – The right-side expression is evaluated first, and then its result is stored in the variable at left.

- What happens here?

```
int x = 3;
x = x + 2; //Evaluate right side then put it into x
```



ASSIGNMENT AND TYPES

- A variable can only store a value of its own type.
  - `int x = 2.5;` **`// ERROR: incompatible types`**
- An int value can be stored in a double variable.
  - The value is converted into the equivalent real number.
  - `double myGPA = 4;`

  - `double avg = 11 / 2;`

    - Why does `avg` store `5.0` and not `5.5` ?

COMPILER ERRORS

- A variable can't be used until it is assigned a value.
  -
```
int x;
System.out.println(x);   // ERROR: x has no value
```

- You may not declare the same variable twice.
  -
```
int x;
 int x;                  // ERROR: x already exists
```
  -
```
int x = 3;
int x = 5;               // ERROR: x already exists
```

    - How can this code be fixed?

PRINTING A VARIABLE'S VALUE

- Use **+** to print a string and a variable's value on one line.

```
double grade = (95.1 + 71.9 + 82.6) / 3.0;
System.out.println("Your grade was " + grade);

int students = 11 + 17 + 4 + 19 + 14;
System.out.println("There are " + students +
                   " students in the course.");
```

- Output:

```
Your grade was 83.2
There are 65 students in the course.
```

PRECEDENCE QUESTIONS

- Improve the receipt program using variables.

```
public class Receipt {
    public static void main(String[] args) {
       // Calculate total owed, assuming 8% tax / 15% tip
       System.out.print("Subtotal: ");
       System.out.println(38 + 40 + 30);
       System.out.print("Tax: ");
       System.out.println((38 + 40 + 30) * .08);
       System.out.print("Tip: ");
       System.out.println((38 + 40 + 30) * .15);
       System.out.print("Total: ");
       System.out.println(38 + 40 + 30 +
       (38 + 40 + 30) * .08 +
       (38 + 40 + 30) * .15);
    }
}"
```

RECEIPT ANSWER

```
public class Receipt {
    public static void main(String[] args) {
        // Calculate total owed, assuming 8% tax / 15% tip
        int subtotal = 38 + 40 + 30;
        double tax = subtotal * .08;
        double tip = subtotal * .15;
        double total = subtotal + tax + tip;

        System.out.println("Subtotal: " + subtotal);
        System.out.println("Tax: " + tax);
        System.out.println("Tip: " + tip);
        System.out.println("Total: " + total);
    }
}
```

Any Questions?