# Elementary Programming
## Part2 : Style, Mathematical Operations, Console Input

Course: CPSC 1150

Instructor: Dr. Bita Shadgar

Lecture 4

# Learning Outcomes

- Write a program in Style
- Apply the basic mathematical operations in the program
- Identify augmented assignment operators, increment operators and decrement operators
- Define char and String
- Receive user input in the program
- Define a package
- Import a package into a program
- Apply above syntax and semantic to write a program

# Review of good coding style

- Indent all statements inside each block
- Opening curly brace goes directly after class or method definition
  - Can alternatively place it on the next line, by itself
  - Whichever style you choose, be consistent
- Closing curly brace is on the line after the last statement in the block, aligned with class or method definition

**Same-line brace**

```
public class MyClass {
   public static void main(String[] args) {
       // statements in here
   }   // end main
}   // end class
```

# Good style continued

## Next-line brace

```
public class MyClass
{
   public static void main(String[] args)
   {
         // statements in here
   } // end main
} // end class
```

– Follow naming conventions for ClassName, objectName, varName, and CONSTANT_NAME identifiers

# Basic mathematical operators

| Name | Symbol |
|---|---|
| **Multiplication** | * |
| **Division** | / |
| **Addition** | + |
| **Subtraction** | - |
| **Remainder** | % |

- We have already seen most of these
- Beware of the output type
- The output data type of any operation is always the most precise of the two input arguments

## Example

Multiplying an `int` by a `float` produces a `float`.
Dividing an `int` by an `int` produces an `int`.
Adding a `float` and a `double` produces a `double`.

Question: How could this kind of behavior cause unintended logic errors?

# Example – Output Data Type

```
/** This program converts a temperature from
 ** Fahrenheit to Celsius
 */
public class F2C {
   public static void main(String[] args) {
      double celsius, fahrenheit = 400;
      celsius = (5 / 9) * (fahrenheit - 32);
      System.out.println(fahrenheit
                         + " fahrenheit is "
                         + celsius + " celsius.");
   }
}
```

**$javac F2C.java**

**$java F2C**

400.0 fahrenheit is 0.0 celsius.

# Example of remainder %

```
public class DisplayTime {
  public static void main(String[] args) {
    int seconds = 1675;
    int minutes = seconds / 60;
    int remainingSeconds = seconds % 60;
    System.out.println(seconds + " seconds is "
                       + minutes + " minutes and "
                       + remainingSeconds + " seconds");
  }
}
```

**$javac DisplayTime.java**

**$java DisplayTime**

1675 seconds is 27 minutes and 55 seconds

# Augmented assignment operators

| Operator | Example | Equivalent |
|---|---|---|
| += | i += 8 | i = i + 8 |
| -= | f -= 8.0 | f = f - 8.0 |
| *= | i *= 8 | i = i * 8 |
| /= | i /= 8 | i = i / 8 |
| %= | i %= 8 | i = i % 8 |

– Practice the following:

```
short a = 25;
a %= 3;
System.out.println("a = " + a);
```

# Increment and decrement operators

| Operator | Example | Equivalent |
|---|---|---|
| var++ | i++ | i = i + 1 |
| ++var | ++i | i = i + 1 |
| --var | --i | i = i - 1 |
| var-- | i-- | i = i - 1 |

– Practice the following:

```
int a =5, x;
x = 3 + ++a + 4;
System.out.println("a = " + a + "\nx = " + x);
```

# Precedence of operators

1. ()
2. * / %
3. + -
4. =

Note: If there are several of operations with the same precedence, they are evaluated from left to right.

**Exercise**

Write the corresponding java expression for the following mathematical functions:

$$z = b^2 + 4ac \quad , \quad x = \frac{a+b}{c-d} \quad , \quad y = \frac{1}{x^2 + x + 3}$$

# A Possible Bug

– Never use combination of `a++, ++a, a--, --a,` and `a` in a single statement.

**Possible bug**

```
a = 5;
x = 5 + ++a + 2+ a + a++;
```

The result may be different in different compilers.

# Character Data Type

- The increment and decrement operators can be used on char variables to get the next or preceding Unicode character.

**Example**

The following statements display character *b*.

```
char ch = 'a';
System.out.println(++ch);
```

# ASCII Character Set

– ASCII Character Set is a subset of the Unicode from \u0000 to \u007f

| Left Digit(s) \ Right Digit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| | *ASCII* | | | | | | | | | |
| 0 | NUL | SOH | STX | ETX | EOT | ENQ | ACK | BEL | BS | HT |
| 1 | LF | VT | FF | CR | SO | SI | DLE | DC1 | DC2 | DC3 |
| 2 | DC4 | NAK | SYN | ETB | CAN | EM | SUB | ESC | FS | GS |
| 3 | RS | US | □ | ! | " | # | $ | % | & | ' |
| 4 | ( | ) | * | + | , | – | . | / | 0 | 1 |
| 5 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; |
| 6 | < | = | > | ? | @ | A | B | C | D | E |
| 7 | F | G | H | I | J | K | L | M | N | O |
| 8 | P | Q | R | S | T | U | V | W | X | Y |
| 9 | Z | [ | \ | ] | ^ | _ | ` | a | b | c |
| 10 | d | e | f | g | h | i | j | k | l | m |
| 11 | n | o | p | q | r | s | t | u | v | w |
| 12 | x | y | z | { | \| | } | ~ | DEL | | |

**FIGURE 3.5** The ASCII character set

# The String class

## Definition

- A string is a sequence of characters.
- Java has no primitive type to show strings. Class String is a predefined class in java.lang package that can be used to represent string.

- Use the same syntax to declare and initialize the String objects

## Example

```
String message = "Welcome to Java";
```

# String Concatenation

```
// Three strings are concatenated
String message = "Welcome " + "to " + "Java";

// String Chapter is concatenated with number 2
String s = "Chapter" + 2; // s becomes Chapter2

//String Part is concatenated with character B
String s1 = "Part" + 'B'; //s1 becomes PartB
```

# User input

- Until now, we have had to hard-code input directly into our programs
  - We want any user to be able to interact with java programs
    - Non-programmers shouldn't have to deal with code to change inputs
- Asking for inputs also helps us as programmers
  - Can write and compile code once, and then use it many times for different inputs

**Example**

Let's see what happens when we run `FahreheitToCelsius`, that accepts user input.

# Example : Hard-coded program

```
/** This program converts a temperature from
 ** Fahrenheit to Celsius
 */
public class F2C {
   public static void main(String[] args) {
      double celsius, fahrenheit = 400;
      celsius = (5.0 / 9) * (fahrenheit - 32);
      System.out.println(fahrenheit
                         + " fahrenheit is "
                         + celsius + " celsius.");
   }
}
```

**$javac F2C.java**

**$java F2Celsius**

400.0 fahrenheit is 204.44444444444446 celsius.

# Prompting for input

- The first task, when user input is desired, is to ask for input by printing to the console
- Be very specific and know your users
  - Describe the format their input must take
  - Describe the meaning the input should have
- It may help to remind the user to hit the “Enter” key

**Example**

Please type your height in centimeters, rounded to the nearest whole number, and then press enter.

# Catch the input using Scanner class

- The next step is to catch the input that the user types.
  - Need to introduce an object of `Scanner` type (class)
  - Located in `java.util` package in the API

**Definition**

- A `Scanner` is a class used to read from the console or a file. The `Scanner` class has **methods** that can be invoked on a `Scanner` object in order to ask the object to perform a task.
- A **package** is a namespace that organizes a set of related classes and interfaces. Conceptually you can think of packages as being similar to **different folders** on your computer.

# Importing a package

- To use the pre-defined classes, you must import the class.
- You need to know to which package the class belongs.
- This line tells the compiler to verify the existence of the class add, and loader loads its bytecode into JVM when your code is running.

**Example**

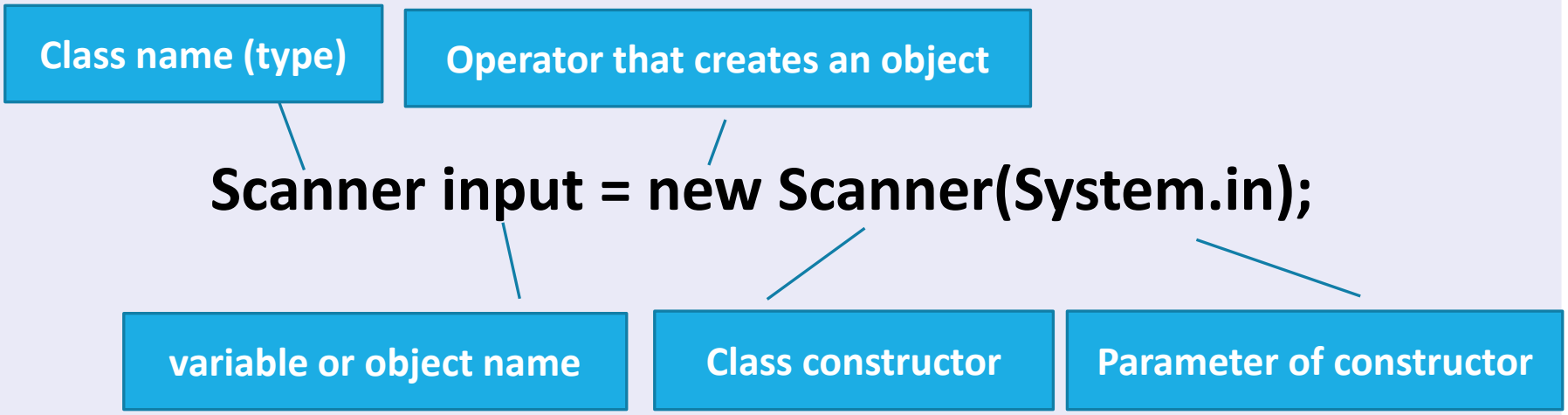
Before creating a `Scanner` object, you must include the following statement at the beginning of your file

**`import java.util.Scanner;`** or **`import java.util.*;`**

Which one is a better practice?

# Creating an object of Scanner class

## Declaration and Initialization




- The Scanner class is very powerful – we will see later on that it can also be used to scan files and Strings!
- Don't use a new Scanner each time you get console input
  - One is enough, reuse it

# Some methods in the Scanner class

**Example**

The Scanner method nextInt() returns the next int typed by the user, and stops when there is whitespace entered.

**Example**

The Scanner method next() returns the next String of characters typed by the user, stopping when whitespace is entered.

**Definition**

A **token** is an input sequence containing no whitespace.

There are methods to check if the input has a next **token** of various data types. They return true if the answer is yes, and false otherwise.

# Reading in an `int` or `float`

– Once you have created a `Scanner` named `input`, you can. . .

**Read an `int`**

```
int heightInCm;
heightInCm = input.nextInt();
```

– Note the use of the `objectName.method();` syntax – this is the general way to invoke a method belonging to an object

◦ Does the syntax remind you of something?

**Read a `float`**

```
float productCost;
productCost = input.nextFloat();
```

Note: Similar methods exist for all the types we have learned

# Closing a Scanner

- There is also a Scanner method which closes the input stream when the program is done with it
- Nothing too bad will happen if you forget to do this
  - Main reason to close a Scanner is to prevent unintended access to it
- Other objects, such as files, are very important to close, even if you are just coding a self-contained program

**Syntax**

```
input.close(); //here input is the name of the Scanner
```

# Questions to ask ourselves

Variables: What variables will we need? What types should they be?

Constants: Do we need any constants? If so, what type(s)?

Calculations: What kinds of calculations might we need to do?

Should we look up relevant formulas?

Input: Do we need user input? If so, what should we prompt for?

Output: What information do we need to return to the user?

Packages: Do we need to import any packages?

# More Practice

```
a = 5;
x = 3 - a++ +4;
x = ?
a = ?
```

```
a = 7;
x = 10 - --a +3;
x = ?
a = ?
```

```
a = 5;
x = 3 + ++a +4;
x = ?
a = ?
```

```
a = 7;
X = 7 + a-- -3;
x = ?
a = ?
```

# More Practice

```
y=3;
y*=5+2;
y = ?
```

```
y=6, x=5;
y-=x++ +3;
y = ?
```

```
y=6, x=5;
y%=--x +3;
y = ?
```

# More Practice: Calculating tips on meals

**Problem statement**

Write a program that asks the user for the subtotal (at a restaurant) and the rate at which they'd like to tip the server. The program calculates and prints the amount for the gratuity (tip), and the total cost of the meal.

Note: This is programming exercise 2.5 in the textbook.