Augmented Assignment and Data Type Casting

Hui Chen

Department of Computer & Information Science

Brooklyn College

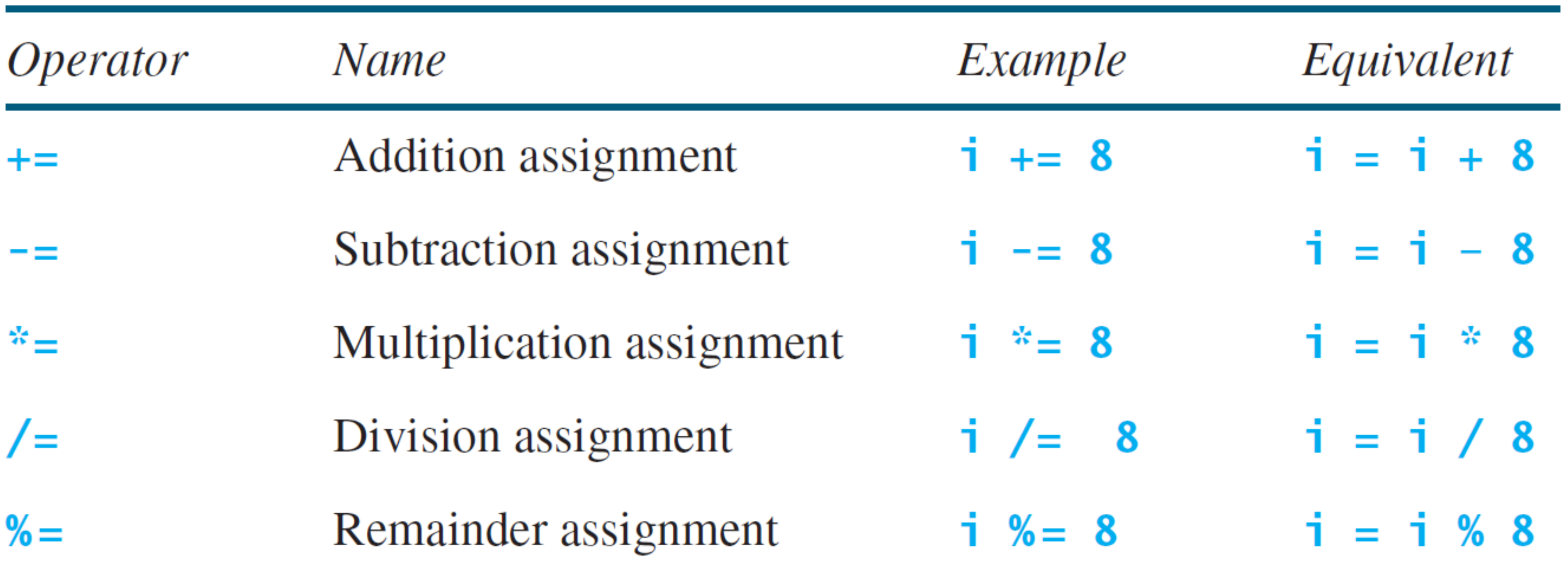
Objectives

- To program with assignment statements and assignment expressions (§2.6).
- To use augmented assignment operators (§2.13).
- To distinguish between postincrement and preincrement and between postdecrement and predecrement (§2.14).
- To cast the value of one type to another type (§2.15).

Outline

- Discussed
  - Problem → Algorithm → Implementation
  - Design a program with input and output
  - Naming convention (best practice)
  - Numeric data types
  - Numeric operators (operating on numeric data types)
- This lesson covers
  - Augmented assignment statements
  - Increment and decrement operators
  - Type casting

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`

Post- vs. Pre- Increment/Decrement

```
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;
```

Best Practice

- 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
- Example. Don’t write this – although is grammatically correct, it is difficult to read, and to understand for some
  - int k = ++i + i;

Augmented Assignment Expressions and Assignment Statements

- Only the following types of expressions can be statements

```
variable op= expression; // Where op is +, -, *, /, or %

++variable;

variable++;

--variable;

variable--;
```

Questions?

Numeric Type Conversion

- Consider the following statements

```
byte i = 100;
long k = i * 3 + 4;
double d = i * 3.1 + k / 2;
```

Conversion Rules

- When performing a binary operation involving two operands of different types, Java automatically converts the operand based on the following rules:
    1. If one of the operands is double, the other is converted into double.
    2. Otherwise, if one of the operands is float, the other is converted into float.
    3. Otherwise, if one of the operands is long, the other is converted into long.
    4. Otherwise, both operands are converted into int.

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;

Range of Values

`range increases` $\longrightarrow$

`byte, short, int, long, float, double`

Questions?

Let’s use these to solve a problem

- Convert a Fahrenheit degree to Celsius and keep two digits after the decimal point.

Problem. Keeping two digits after decimal point

- We want to display GPA in a nice format, i.e., only display two digits after the decimal point.
- We are computing sales tax, but the smallest denomination is a cent. So, …

Computing and Displaying Sales Tax

```
import java.util.Scanner;

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

    System.out.print("Enter purchase amount in cents: ");
    long purchaseAmount = input.nextLong();

    double tax = purchaseAmount * 0.06;
                  long taxInCents = Math.round(tax)
                  long dollars = taxInCents / 100;
                  int cents = taxIncents % 100;
                  // ...
  }
}
```

How about this (solution in the textbook)?

```
import java.util.Scanner;

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

    System.out.print("Enter purchase amount: ");
    double purchaseAmount = input.nextDouble();

    double tax = purchaseAmount * 0.06;
    System.out.println("Sales tax is " + (int)(tax * 100) / 100.0);
  }
}
```

Questions?

- Is there a bug in
  - System.out.println("Sales tax is " + (int)(tax * 100) / 100.0);

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 after this statement
```

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

Questions?

Lab Exercise

- Write a program that convert a Fahrenheit degree to Celsius and keep two digits after the decimal point ***without rounding***.