جامعة بيرزيت
BIRZEIT UNIVERSITY

Selections

Comparison Operators

Java Operator	Mathematics Symbol	Name	Example (radius is 5)	Result
`<`	$<$	less than	`radius < 0`	`false`
`<=`	$\leq$	less than or equal to	`radius <= 0`	`false`
`>`	$>$	greater than	`radius > 0`	`true`
`>=`	$\geq$	greater than or equal to	`radius >= 0`	`true`
`==`	$=$	equal to	`radius == 0`	`false`
`!=`	$\neq$	not equal to	`radius != 0`	`true`

if-else

```
if (radius >= 0) {
    area = radius * radius * 3.14159;
    System.out.println("The area for the " +
    "circle of radius " + radius +  " is " + area);
}
else {
    System.out.println("Negative input");
}
```

Common Errors

❖ Adding a **semicolon** at the end of an **if** clause is a common mistake.

```
if (radius >= 0) ;        <-------- Wrong
{
  area = radius*radius*PI;
  System.out.println( "The area for the circle is " + area);
}
```

❖ This mistake is hard to find, because it is not a compilation error or a runtime error, it is a **logic** error.

❖ This error often occurs when you use the next-line block style.

Logical Operators

Operator	Name
`!`	not
`&&`	and
`||`	or
`^`	exclusive or

switch Statements

```
switch (status) {
      case 0:  compute taxes for single filers;
               break;
      case 1:  compute taxes for married file jointly;
               break;
      case 2:  compute taxes for married file separately;
               break;
      case 3:  compute taxes for head of household;
               break;
      default: System.out.println("Errors: invalid status");
               System.exit(1);
}
```

Conditional Operator

```
if (x > 0)
    y = 1;
else
    y = -1;
```

- is equivalent to:

```
y = (x > 0)  ?  1  :  -1;
```

(boolean-expression) **?** expression1 **:** expression2

Conditional Operator

```
if (num % 2 == 0)
    System.out.println(num + “is even”);
else
    System.out.println(num + “is odd”);
```

```
System.out.println(   (num % 2 == 0)  ?  num + “is even” :   num + “is odd”);
```

Formatting Output

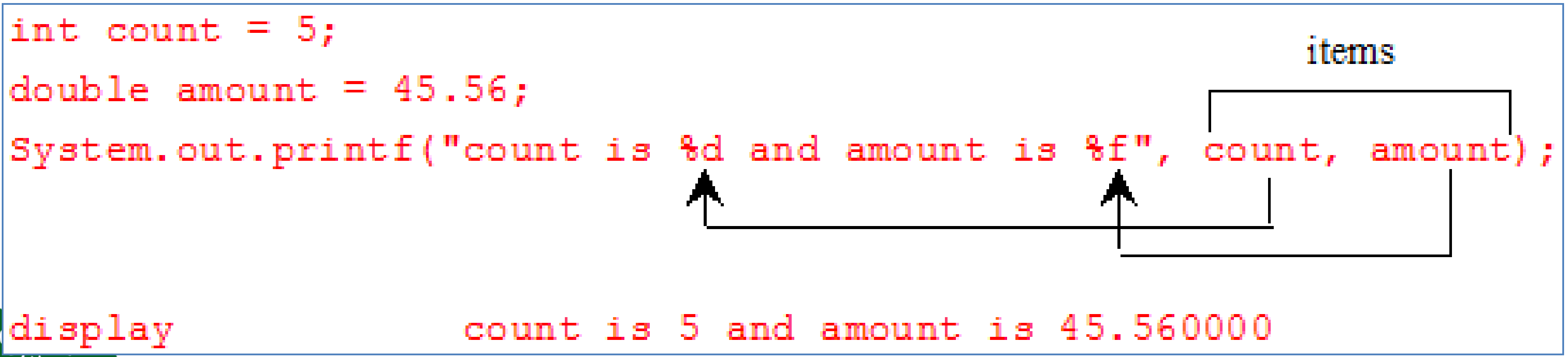
- Use the **printf** statement:

```
System.out.printf( format, items );
```

- Where format is a string that may consist of substrings and **format specifiers**.
- A format specifier specifies how an item should be displayed.
- An item may be a numeric value, character, boolean value, or a string.
- Each specifier begins with a **percent** sign.

Frequently-Used Specifiers

Specifier	Output	Example
%b	a boolean value	true or false
%c	a character	'a'
%d	a decimal integer	200
%f	a floating-point number	45.460000
%e	a number in standard scientific notation	4.556000e+01
%s	a string	"Java is cool"



```
int count = 5;
double amount = 45.56;
System.out.printf("count is %d and amount is %f", count, amount);



display          count is 5 and amount is 45.560000
```

Operator Precedence

- `var++, var--`
- `+, - (Unary plus and minus), ++var,--var`
- `(type) Casting`
- `! (Not)`
- `*, /, % (Multiplication, division, and remainder)`
- `+, - (Binary addition and subtraction)`
- `<, <=, >, >= (Comparison)`
- `==, !=; (Equality)`
- `^ (Exclusive OR)`
- `&& (Conditional AND) Short-circuit AND`
- `|| (Conditional OR) Short-circuit OR`
- `=, +=, -=, *=, /=, %= (Assignment operator)`

Operator Precedence and Associativity

- ❖ The expression in the parentheses is evaluated first. (Parentheses can be nested, in which case the expression in the inner parentheses is executed first.)

- ❖ When evaluating an expression without parentheses, the operators are applied according to the precedence rule and the **associativity rule**.

- ❖ If operators with the same precedence are next to each other, their associativity determines the order of evaluation. All binary operators except assignment operators are **left-associative**.

Operator Associativity

- When two operators with the same precedence are evaluated, the *associativity* of the operators determines the order of evaluation.
- All binary operators except assignment operators are ***left-associative***.

  a – b + c – d is equivalent to **((a – b) + c) – d**
- Assignment operators are ***right-associative***. Therefore, the expression

  a = b += c = 5 is equivalent to **a = (b += (c = 5))**