

Flow of Control

- The order in which statements in a program are executed is called the *flow of control*
- So far we have only seen sequential execution: statements execute one after the other in the order in which they appear in the program

Flow of Control cont.

- Consider the following tasks:
 - You want to compute the quotient of two variables but only if the divisor is not zero
 - You input some value (e.g., a date) and if it is in the correct format (mm/dd/yyyy) you continue the computation, otherwise you print an error
 - Given a grade between 0 and 100, you want to convert the numeric value to a letter grade (e.g., A for grade greater than 90, B for grade between 80 and 90, etc.)
- How can we check these conditions and execute the appropriate piece of code depending on the outcome of the check?

Selection Statements

```
Scanner in = new Scanner(System.in);
System.out.print("Enter the dividend: ");
int dividend = in.nextInt();
System.out.print("Enter the (non-zero) divisor: ");
int divisor = in.nextInt();
if (divisor != 0) {
    int quotient = dividend / divisor;
    System.out.println(
        dividend + " / " + divisor + " = " + quotient);
}
else {
    System.out.println("Cannot divide by 0");
}
```

The boolean Type

- A variable of the **boolean** data type stores one of two values: **true** or **false**
- **true** and **false** are the only two boolean constants
- boolean values/expressions are used to make decisions in programs
- For example:

```
if (x > 0) // boolean expression
{
    System.out.println("x is positive");
}
```

Boolean Expressions

- There are several kinds of boolean-valued expressions:

- a boolean variable or constant, e.g.,

```
boolean boolVar = in.nextBoolean();  
if (boolVar) { ... }
```

- an arithmetic expression followed by a relational operator followed by an arithmetic expression, e.g.,

```
int intVar = in.nextInt();  
if (intVar > 0) { ... }
```

Relational Operators

- | | |
|------------------|----------|
| ▪ == (equal) | $x == y$ |
| ▪ != (not equal) | $x != y$ |
| ▪ > | $x > y$ |
| ▪ < | $x < y$ |
| ▪ >= | $x >= y$ |
| ▪ <= | $x <= y$ |

Boolean Operators

- We can also build boolean expressions by combining two boolean expressions with a boolean operator:
 - `&&` (and) `(x > 0) && (x < 10)`
 - `||` (or) `(x <= 0) || (x >= 10)`
 - `!` (not) `!(x == 0)`

Boolean Operators cont.

- If A and B are boolean expressions, **A && B** is true if and only if both A and B are true (in other words, if either A or B or both are false, A && B is false)
- If A and B are boolean expressions, **A || B** is true if either A or B or both are true (in other words, A || B is false only if both A and B are false)
- If A is a boolean expression, **!A** is true if A is false, and **!A** is false if A is true

Some Boolean Expressions

- What's the value of each of the following expressions:

```
int x = 5, y = 12;
```

```
boolean a = true, b = false, c = true;
```

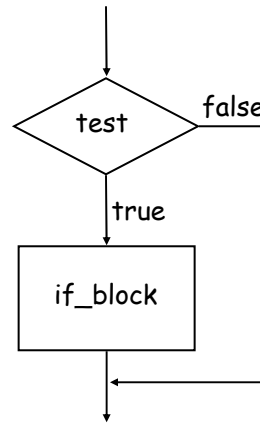
- $(x > 0) \ \&\& \ (x < 10)$
- $(x \leq 0) \ \|\| \ (x \geq 10)$
- $! (a \ \&\& \ b \ \&\& \ c)$
- $(a \ \|\| \ b \ \|\| \ c)$
- $((x - 1) == ((y / 5) + (y \% 5)))$
- $((x != y) \ \|\| \ ! (x == y))$

Your Turn

- Given three integer variable, i , j , and k , write a boolean expression for each of the following problems:
 - i is equal to 3 or 5
 - i is between 1 and 7 (but not 1 or 7)
 - i is even
 - i is odd
 - i is the smallest of i , j , and k

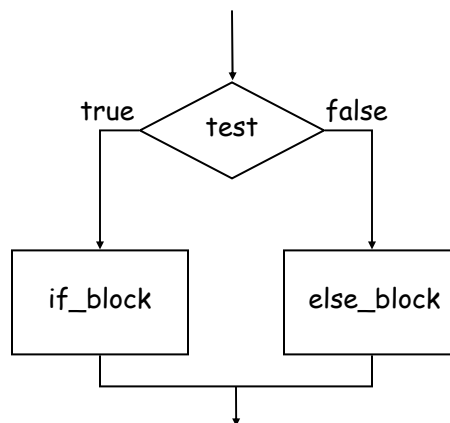
If: Syntax and Flow Chart

```
if ( test )  
{  
    if_block  
}
```



If-Else: Syntax and Flow Chart

```
if ( test )  
{  
    if_block  
}  
else  
{  
    else_block  
}
```



An Example

- Given an integer i , write a piece of code that outputs “even” or “odd” depending on whether the value of i is even or odd.

Another Example

- Given two integers i and j , write a piece of code that sets integer variable max to the value of the larger of the two.

Your Turn

- Given three integers i , j , and k , write a piece of code that sets integer variable max to the value of the largest of the three.

Max Of Three

Your Turn, Again

- Given an integer, *grade*, holding a grade between 0 and 100, write a piece of code that converts the numeric value to a letter grade according to the following table and prints the letter grade.

$grade \geq 90$	A
$80 \leq grade < 90$	B
$70 \leq grade < 80$	C
$60 \leq grade < 70$	D
$grade < 60$	E

Grade Conversion

Your Turn, One More Time

- Given a String, *s*, which is meant to hold a date in the **mm/dd/yyyy** format, check that it is in the correct format and print an error message if it is not.
- **boolean** `Character.isDigit(char ch)` allows you to check if a given character, *ch*, is a digit ('0', '1', '2', '3', '4', '5', '6', '7', '8', '9') or not.

Check Date Format