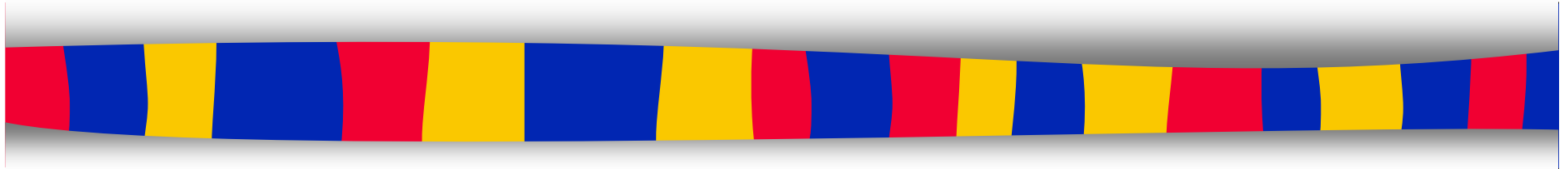


COMP-248

Object Oriented Programming I



Lecture 04:

Control Flow 1

Emad Shihab, PhD

Announcements

Assignment 1 posted on Moodle

If you have questions, first point of contact is your TA!

Procrastination Is Bad for Your Grades!

The results of a **5-year study of marketing students** at the Warwick Business School show an inverse correlation between procrastination and grades. The study of 777 students found that **students who turned in assignments just before a deadline performed worse on assignments than those who turned in their work more than 24 hours early**. There was **little statistical difference among students who submitted assignments more than 24 hours early**; however, after the 24-hour mark, average scores dropped at an increasing rate the closer the submission time was to the deadline. There was a **5% difference in scores** between students who submitted their work at the last minute and those who submitted it more than a day in advance, good for a full letter grade.

Flow of Control

1. Sequence:

Unless specified otherwise, the order of statement execution is linear/sequential

one statement after the other, in sequence

2. Conditional statements:

a statement may or may not be executed depending on some condition

3. Repetition statements (loops):

a statement is executed over and over, repetitively, until some condition becomes true or false

These decisions are based on a **boolean expression** (also called a *condition*) that evaluates to true or false

The order of statement execution is called the **flow of control**

In this chapter, we will see:

1. The `if` statement
2. The `if-else` statement
3. Relations Operators
4. Logical operators
5. Compound statements
6. Nested `if` statements
7. The `switch` statement
8. The conditional operator
9. The `while` loop
10. The `do-while` loop
11. The `for` loop
12. Nested loops
13. `break`, `continue` & `exit`

Conditional statements

let us choose which statement will be executed next

sometimes called *selection statements*

Java has 3 conditional statements:

- the *if* statement

- the *if-else* statement

- the *switch* statement

1- The if statement (p.96)

syntax:

`if` is a Java
reserved word

The *condition* is a boolean expression.
(evaluates to either true or false)

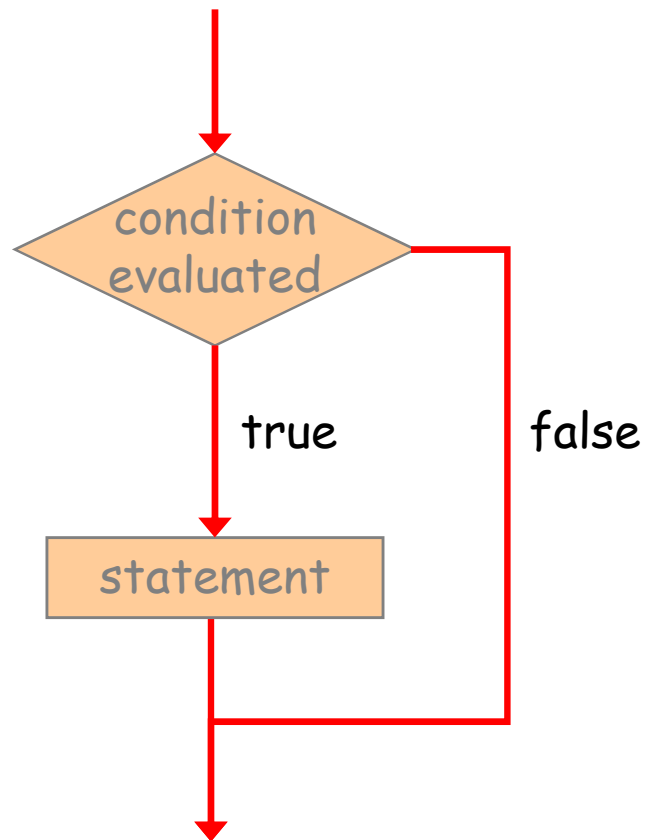


```
if ( condition )  
    statement;
```

The diagram illustrates the syntax of an if statement. A central box contains the code `if (condition)
 statement;`. Three red arrows point to different parts of this code: one from the text '`if` is a Java reserved word' to the `if` keyword, one from 'The *condition* is a boolean expression. (evaluates to either true or false)' to the *condition*, and one from 'If the *condition* is true, the *statement* is executed. If it is false, the *statement* is skipped.' to the *statement*.

If the *condition* is true, the *statement* is executed.
If it is false, the *statement* is skipped.

Logic of an if statement



Example



```
System.out.print("Enter the sum: ");  
int sum = myKeyboard.nextInt();  
int delta = 0;  
  
if (sum >= 100)  
    delta = 5;  
  
System.out.println("Delta is " + delta);
```

Output

```
Enter the sum: 5000  
Delta is 5
```

Example: Age.java



```
final int MINOR = 18;  
  
System.out.print("Enter your age: ");  
int age = myKeyboard.nextInt();  
  
if (age < MINOR)  
    System.out.println("wonderful");  
System.out.println("Oh well!");
```

Output

```
Enter your age: 16  
wonderful  
Oh well!
```

In this chapter, we will see:

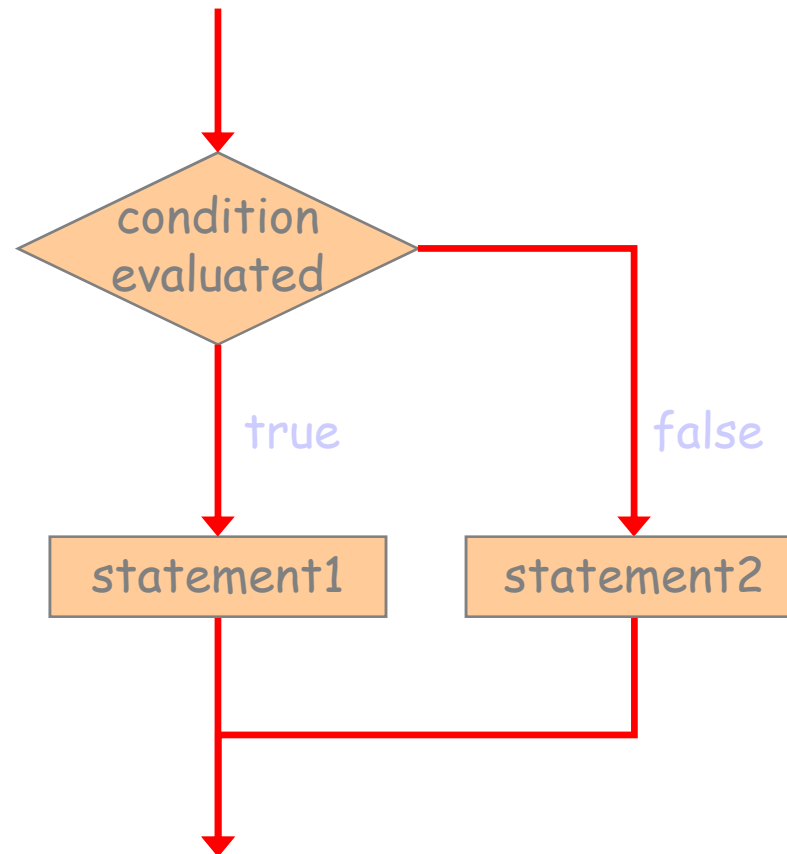
1. The `if` statement
2. **The `if-else` statement**
3. Relations Operators
4. Logical operators
5. Compound statements
6. Nested `if` statements
7. The `switch` statement
8. The conditional operator
9. The `while` loop
10. The `do-while` loop
11. The `for` loop
12. Nested loops
13. `break`, `continue` & `exit`

2- The if-else statement

An *else* clause can be added to an `if` statement to make an *if-else* statement

```
if ( condition )  
    statement1;  
else  
    statement2;
```

Logic of an if-else statement



Example: Wages.java

```
final double RATE = 10.0; // regular pay rate
final int STANDARD = 40; // standard hours
double pay = 0;

System.out.print("Number of hours worked: ");
int hours = myKeyboard.nextInt();

// Pay overtime at "time and a half"
if (hours > STANDARD)
    pay = STANDARD*RATE + (hours-STANDARD) * (RATE*1.5);
else
    pay = hours * RATE;
System.out.println("Pay: " + pay);
```

Number of hours worked: 50
Pay: 550.0

Output

What is the output?

```
int speed = 55;
if (speed > 50)
    System.out.println("Going too fast - School zone");
if (speed > 30)
    System.out.println("Going at the right speed");
else
    System.out.println("You can go a bit faster");
```

- | | | |
|---------------------------------|----|----|
| A. Going too fast - School zone | 55 | |
| B. Going at the right speed | 55 | 45 |
| C. You can go a bit faster | | 25 |
| D. Neither of the above choices | | |

What is the output?

```
int num = 4;  
if (num > 5)  
    System.out.println("line A");  
else  
    System.out.println("line B");  
if (num < 10)  
    System.out.println("line C");  
    System.out.println("line D");
```

See how much harder it is to read if not indented properly ...

- A. line A
line B
line C
line D
- B. line A
line C
line D
- C. line B
line C
line D
- D. line B
line C
- E. line B
line D

What is the output?

```
int someInt = 10;  
if (someInt > 30)  
    System.out.print("Moe ");  
    System.out.print("Larry ")  
System.out.print("Curly");
```

- A. Curly
- B. Moe Larry Curly
- C. Larry Curly
- D. no output; there is a compile-time error
- E. no output; there is a run-time error

In this chapter, we will see:

1. The `if` statement
2. The `if-else` statement
3. **Relations Operators**
4. Logical operators
5. Compound statements
6. Nested `if` statements
7. The `switch` statement
8. The conditional operator
9. The `while` loop
10. The `do-while` loop
11. The `for` loop
12. Nested loops
13. `break`, `continue` & `exit`

3- Relational operators

needed in control structures (ex. `if`)

to write conditions (boolean expressions)

return boolean results (evaluates to `true` or `false`)

<code>==</code>	equal to
<code>!=</code>	not equal to
<code><</code>	less than
<code>></code>	greater than
<code><=</code>	less than or equal to
<code>>=</code>	greater than or equal to

Note the difference between `==` and `=`

Example - IncomeTax.java

```
if (age == 18)
    System.out.println("you are 18");
else
    System.out.println("you are not 18");
```

Output

```
if (age = 18)
    System.out.println("you are 18");
else
    System.out.println("you are not 18");
```

Output

A note on comparing floats



be careful when comparing 2 floating point values
(float or double) for equality

do not use the equality operator (==)

because floats are approximated

you want to see if two floats are "close enough"

```
if (Math.abs(f1 - f2) < 0.00001)
    System.out.println ("Essentially equal.");
```

A note on comparing characters

We can use the relational operators to compare 2 characters

The results are based on the Unicode character set

```
if ('+' < 'J')  
    System.out.println("+ is less than J in Unicode");
```

```
char userAnswer = 'y';  
if (userAnswer == 'Y')  
    System.out.println("the user said yes");
```

Part of the Unicode Character Set (ASCII)

0	□	32	□	64	@	96	`	128	€	160	□	192	À	224	à
1	□	33	!	65	A	97	a	129	□	161	ı	193	Á	225	á
2	□	34	"	66	B	98	b	130	,	162	ç	194	Â	226	â
3	□	35	#	67	C	99	c	131	f	163	£	195	Ã	227	ã
4	□	36	\$	68	D	100	d	132	~	164	¥	196	Ä	228	ä
5	□	37	%	69	E	101	e	133	...	165	¥	197	Å	229	å
6	□	38	&	70	F	102	f	134	†	166	ı	198	Æ	230	æ
7	□	39	'	71	G	103	g	135	‡	167	§	199	Ç	231	ç
8	□	40	(72	H	104	h	136	^	168	¨	200	È	232	è
9	□	41)	73	I	105	i	137	‰	169	©	201	É	233	é
10	□	42	*	74	J	106	j	138	Š	170	ª	202	Ê	234	ê
11	□	43	+	75	K	107	k	139	<	171	«	203	Ë	235	ë
12	□	44	,	76	L	108	l	140	œ	172	¬	204	Ì	236	ì
13	□	45	-	77	M	109	m	141	□	173	-	205	Í	237	í
14	□	46	.	78	N	110	n	142	Ž	174	@	206	Î	238	î
15	□	47	/	79	O	111	o	143	□	175	—	207	Ï	239	ï
16	□	48	0	80	P	112	p	144	□	176	°	208	Ð	240	ð
17	□	49	1	81	Q	113	q	145	`	177	±	209	Ñ	241	ñ
18	□	50	2	82	R	114	r	146	/	178	²	210	Ò	242	ò
19	□	51	3	83	S	115	s	147	˘	179	³	211	Ó	243	ó
20	□	52	4	84	T	116	t	148	˘	180	´	212	Ô	244	ô
21	□	53	5	85	U	117	u	149	•	181	µ	213	Õ	245	õ
22	□	54	6	86	V	118	v	150	-	182	¶	214	Ö	246	ö
23	□	55	7	87	W	119	w	151	-	183	·	215	×	247	÷
24	□	56	8	88	X	120	x	152	˘	184	¸	216	Ø	248	ø
25	□	57	9	89	Y	121	y	153	˘	185	˙	217	Ù	249	ù
26	□	58	:	90	Z	122	z	154	š	186	°	218	Ú	250	ú
27	□	59	;	91	[123	{	155	>	187	»	219	Û	251	û
28	□	60	<	92	\	124		156	œ	188	¼	220	Ü	252	ü
29	□	61	=	93]	125	}	157	□	189	½	221	Ý	253	ý
30	□	62	>	94	^	126	~	158	ž	190	¾	222	Þ	254	þ
31	□	63	?	95	_	127	□	159	Ÿ	191	¿	223	ß	255	ÿ

A note on comparing strings

We cannot use the relational operators to compare strings (<, ==, ...)

use the `equals()` method
to determine if two strings have the same content

ex: `firstString.equals(secondString)`

returns a boolean:

true if `firstString` has the same content as `secondString`
false otherwise

A note on comparing strings

use the `compareTo()` method

to determine if one string comes before another (based on the Unicode character set)

ex: `firstString.compareTo(secondString)`

returns an `int`:

negative if `firstString` is lexicographically before
`secondString`

positive if `firstString` is lexicographically after
`secondString`

0 if the 2 strings have the same content

Example 1

```
String s1 = "Java isn't just for breakfast.";
String s2 = "JAVA isn't just for breakfast.";

if (s1.equals(s2))
    System.out.println("The two lines are equal.");
else
    System.out.println("The two lines are not equal.");

if (s2.equals(s1))
    System.out.println("The two lines are equal.");
else
    System.out.println("The two lines are not equal.");
```

Example 1 - equalsIgnoreCase

```
String s1 = "Java isn't just for breakfast.";
String s2 = "JAVA isn't just for breakfast.";

if (s1.equalsIgnoreCase(s2))
    System.out.println("But the lines are equal, ignoring case.");
else
    System.out.println("Lines are not equal,even ignoring case.");
```

Example 2

	Syntax Error?	Output?
<code>System.out.println("aBcD" < "abcd");</code>	CE	
<code>System.out.println('aBcD' < 'abcd');</code>	CE	
<code>System.out.println("a" < "b");</code>	CE	
<code>System.out.println('a' < 'b');</code>		true
<code>System.out.println("aBcD".equals("abcd"));</code>		false

Example 2 ...



	Syntax Error?	Output?
<code>System.out.println("aBcD".equalsIgnoreCase("aBcD"));</code>		true
<code>System.out.println("aBcD".compareTo("aBcD"));</code>		0
<code>System.out.println("aBcD".compareTo("aBcC"));</code>		+
<code>System.out.println("abc".compareTo("ab"));</code>		+
<code>System.out.println("abc".compareTo("abcd"));</code>		-

Next class, we will see:

1. The `if` statement
2. The `if-else` statement
3. Relations Operators
4. **Logical operators**
5. Compound statements
6. Nested `if` statements
7. The `switch` statement
8. The conditional operator
9. The `while` loop
10. The `do-while` loop
11. The `for` loop
12. Nested loops
13. `break`, `continue` & `exit`