Decisions (If Statements) 
And 
Boolean Expressions

CSE 1310 – Introduction to Computers and Programming 
Alexandra Stefan 
University of Texas at Arlington

Last updated: 1/30/19
Syntax

```c
if (boolean_expr) {
    if-line 1;
    if-line 2;
    ...
    if-line m;
} else {
    else-line 1;
    else-line 2;
    ...
    else-line n;
}
```

It is a “compound” statement: a statement that includes multiple statements.

Flow-chart

- Last instruction before if
- True: Cond? → True
  - if-line 1
  - if-line 2
  - ...
  - if-line m
- False: Cond? → False
  - else-line 1
  - else-line 2
  - ...
  - else-line n

- 1st instruction after if

Summary

Multiple choices (e.g. for menus):
(There is also a dedicated instruction for this: `switch`).

```
if (condition 1) {
    ...
}
else if (condition 2) {
    ...
}
else if (condition 3) {
    ...
}
... else if (condition n) {
    ...
}
else {  //with or without this else
}
```
import java.util.Scanner;

public class example1 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.printf("How old are you? ");
        int age = in.nextInt();

        if (age < 21) {
            System.out.printf("How about some milk?\n");
        } else {
            System.out.printf("How about some beer?\n");
        }
        System.out.printf("Bye\n");
    }
}

Example Output 1:
How old are you? 18

Example Output 2:
How old are you? 21

Example Output 3:
How old are you? 24
import java.util.Scanner;

public class example1 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.printf("How old are you? ");
        int age = in.nextInt();

        if (age < 21)
        {
            System.out.printf("How about some milk?\n");
        }
        else
        {
            System.out.printf("How about some beer?\n");
        }
        System.out.printf("Bye\n");
    }
}

Example Output 1:
How old are you? 18
How about some milk?
Bye

Example Output 2:
How old are you? 21
How about some beer?
Bye

Example Output 3:
How old are you? 24
How about some beer?
Bye
Example 2: *no else*

```java
import java.util.Scanner;

public class example1 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.printf("How old are you? ");
        int age = in.nextInt();

        if (age <= 18) {
            System.out.printf("You are a teenager.\n");
            System.out.printf("You are young.\n");
        }
        System.out.printf("Bye.\n");
    }
}
```

Example Output 1:
```
How old are you? 29
```

Example Output 2:
```
How old are you? 15
```

Note: the *else* part of an if statement IS OPTIONAL.
No *else* in this example.
Example 2: *no else*

```java
import java.util.Scanner;

public class example1 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.printf("How old are you? ");
        int age = in.nextInt();

        if (age <= 18) {
            System.out.printf("You are a teenager.\n");
            System.out.printf("You are young.\n");
        }
        System.out.printf("Bye\n");
    }
}
```

Example Output 1:
How old are you? 29

Example Output 2:
How old are you? 15

Note: the *else* part of an if statement IS OPTIONAL. No *else* in this example.
import java.util.Scanner;

public class example1 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.printf("How old are you? ");
        int age = in.nextInt();

        if (age <= 18) {
            System.out.printf("You are a teenager.\n");
            System.out.printf("You are young.\n");
        }
        System.out.printf("Bye\n");
    }
}

Example Output 1:
How old are you? 29
Bye

Example Output 2:
How old are you? 15
You are a teenager.
You are young.
Bye

Note: the else part of an if statement IS OPTIONAL. No else in this example.
First way:
{ placed under if

Second way:
{ placed at the end of the if line.
This is also fine, if you want to do it that way.
public class example1 {
    public static void main(String[] args) {
        int a = 3;
        if (a > 5);
            System.out.printf("a = %d.\n", a);
        System.out.printf("a > 5.\n");
    }
}

Run it. What will it print?
Common Bug: semicolon

It will print:

```java
public class example1 {
    public static void main(String[] args) {
        int a = 3;
        if (a > 5);
        {
            System.out.printf("a = %d.\n", a);
            System.out.printf("a > 5.\n");
        }
    }
}
```

What is the problem?

What did you expect it to print?

– Nothing since (a>5) is false.
Common Bug: semicolon

It will print:

a = 3.
a > 5.

What is the problem?

Semicolon on the if line.

fixed.
Nested if statements

• See example
Combining if statements: 4 ways

• Sequential:
  – one after the other

• Nested
  – One inside another

• Multiple choice
  – More than 2 options.
  – E.g. given age, print: baby, toddler, kid, teenager,...

• A combination of the above
Sequential if-statements

```java
if (condition 1) {
    ... // block A
} else {
    ... // block B
}
if (condition 2) {
    ... // block C
} else {
    ... // block D
}
if (condition 3) {
    ... // block E
} else {
    ... // block F
}
```
Multiple Cases / Multiple Choice

if (condition 1) {
    ... // block A
}
else if (condition 2) {
    ... // block B
}
else if (condition 3) {
    ... // block C
}
else {
    // with or without this else
    ... // block D
}

E.g.: print category by age:
    age < 18: not an adult
    18 ≤ age < 40: young adult
    40 ≤ age < 60: middle-aged
    60 ≤ age: senior citizen

Draw the flow chart!
if (condition 1) {
  ...
  // block A
}
else if (condition 2) {
  ...
  // block B
}
else if (condition 3) {
  ...
  // block C
}
else {
  // with or without this else
  ...
  // block D
}

Important:
- If C1 is True, C2 will NOT even be evaluated!
- The else “binds” them together. DO not skip it!

How many different paths of execution are possible here?

A path of execution is the sequence of instructions that a program goes through. It results in different program behavior.
Multiple Cases

Error: missing else

Draw the flow chart!

```java
if (condition 1) {
    ... // block A
}
if (condition 2) {// no else
    ... // block B
}
if (condition 3) {// no else
    ... // block C
}
else {//with or without this else
    ... // block D
}
```
Multiple Cases  Error: missing else
Becomes Sequential if statements

if (condition 1) {
    ... // block A
}
if (condition 2) { // no else
    ... // block B
}
if (condition 3) { // no else
    ... // block C
}
else { // with or without this else
    ... // block D
}
public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    System.out.printf("How old are you? ");
    int age = in.nextInt();

    if (age < 18) {
        System.out.printf("You are not an adult.\n");
    }
    else if (age < 40) {
        System.out.printf("You are a young adult.\n");
    }
    else if (age < 60) {
        System.out.printf("You are middle aged.\n");
    }
    else {
        System.out.printf("You are a senior citizen.\n");
    }
}
public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    System.out.printf("How old are you? ");
    int age = in.nextInt();

    if (age < 18) {
        System.out.printf("You are not an adult.\n");
    } else if (age < 40) {
        System.out.printf("You are a young adult.\n");
    } else if (age < 60) {
        System.out.printf("You are middle aged.\n");
    } else {
        System.out.printf("You are a senior citizen.\n");
    }
}
import java.util.Scanner;

public class example1 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.printf("How old are you? ");
        int age = in.nextInt();

        if (age < 40) {
            System.out.printf("You are young.\n");
        } else if (age < 60) {
            System.out.printf("You are middle aged.\n");
        } else {
            System.out.printf("You are old.\n");
        }
    }
}
import java.util.Scanner;

public class example1 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.printf("How old are you? ");
        int age = in.nextInt();

        if (age < 40)
        {
            System.out.printf("You are young.\n");
        }
        else if (age < 60)
        {
            System.out.printf("You are middle aged.\n");
        }
        else
        {
            System.out.printf("You are old.\n");
        }
    }
}
Auto-indent your code in jGRASP

- Most IDEs will have an option for code “auto indent”
- jGRASP does not, but you can trick it to do it, by generating a CSD (Control Structure Diagram). Do:
  View -> Generate CSD
after this you can keep the CSD or remove it (View -> Remove CSD)
These two examples do not use braces under if.

This is legal, but it can lead to bugs when you add more lines.

***NOT*** RECOMMENDED
public class example1 {
    public static void main(String[] args) {
        int a = 3;
        if (a > 5)
            System.out.printf("a = %d.\n", a);
        System.out.printf("a > 5.\n");
    }
}

Not using braces under if: it is legal, but it can lead to bugs when you add more lines.

What will this example print?

How many if-lines are there?
Not Using Braces - Example of Bug

```java
public class example1 {
    public static void main(String[] args) {
        int a = 3;
        if (a > 5)
            System.out.printf("a = %d.\n", a);
        System.out.printf("a > 5.\n");
    }
}
```

Not using braces under if: it is legal, but it can lead to bugs when you add more lines.

What will this example print?
`a > 5`

How many if-lines are there?
Just one (if you do not use braces under if, there can only be one if-line).

The top example does the same thing as the bottom example.
Not Using Braces - Example of Bug

```java
public class example1 {
    public static void main(String[] args) {
        int a = 3;
        if (a > 5)
            System.out.printf("a = %d.\n", a);
        System.out.printf("a > 5.\n");
    }
}
```

```
public class example1 {
    public static void main(String[] args) {
        int a = 3;
        if (a > 5)
            {
                System.out.printf("a = %d.\n", a);
                System.out.printf("a > 5.\n");
            }
    }
}
```

If you wanted two if-lines, you should have used braces, as shown on the bottom example on this slide.
The Importance of Indentation

import java.util.Scanner;

public class example1 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.printf("How old are you? ");
        int age = in.nextInt();

        if (age < 40) {
            System.out.printf("You are young.\n");
        } else if (age < 60) {
            System.out.printf("You are middle aged.\n");
        } else {
            System.out.printf("You are old.\n");
        }
    }
}

This program is indented appropriately.

Every time we open a brace, we increase indentation.

Every time we close a brace, we decrease indentation.

Netbeans does this for you automatically, but may get confused every now and then, and then you need to fix the indentations manually.
The Importance of Indentation

This program is indented inappropriately.

Indentation does not change program behavior, but makes program harder to read, and mistakes harder to find.
Indentation on NetBeans

- NetBeans can automatically indent your code.
- Select Source->Format.
- This will work only if your code is valid Java code. If your code cannot run because of syntax errors, NetBeans may get confused about the correct indentation.
• Write a program that:
  – Asks the user to enter the name of the month.
  – Prints "M has X days" where M is the month and X is the correct number of days.
  – If the user did not enter a valid month name, the program prints "M is not a valid month"
import java.util.Scanner;

class example1 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.printf("Enter the name of a month: ");
        String m = in.next();

        if (m.equals("January") || m.equals("March") || m.equals("May") || m.equals("July") || m.equals("August") || m.equals("October") || m.equals("December")) {
            System.out.printf("%s has 31 days.\n", m);
        }
        else if (m.equals("April") || m.equals("June") || m.equals("September") || m.equals("November")) {
            System.out.printf("%s has 30 days.\n", m);
        }
        else if (m.equals("February")) {
            System.out.printf("%s has 28 or 29 days.\n", m);
        }
        else {
            System.out.printf("%s is not a valid month.\n", m);
        }
    }
}
The String `indexOf` Method

- Suppose that variables `str1` and `str2` are strings.
- Suppose you want to see if `str1` contains `str2`.
- You can call `str1.indexOf(str2)`.
- If `str1` contains `str2`, `indexOf` returns the FIRST position where `str2` appears in `str1`.
- If `str1` does NOT contain `str2`, `indexOf` returns -1.
indexOf Example

• Write a program that:
  – Asks the user to enter a single letter.
  – If the user enters a string with more than one letter, exit the program.
  – If the letter is a vowel, print that it is a vowel
  – Else, print that the letter is not a vowel.
import java.util.Scanner;

public class example1 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.printf("Enter a single letter: ");

        String c = in.next();
        if (c.length() != 1)
            {
                System.out.printf("invalid input.\n");
                System.exit(0);
            }

        String vowels = "aeiouAEIOU";
        int result = vowels.indexOf(c);
        if (result != -1)
            {
                System.out.printf("%s is a vowel.\n", c);
            }
        else
            {
                System.out.printf("%s is not a vowel.\n", c);
            }
    }
}
import java.util.Scanner;

public class example1 {  
    public static void main(String[] args) {  
        Scanner in = new Scanner(System.in);
        System.out.printf("Enter a single letter: ");

        String c = in.next();
        if (c.length() != 1)  
            {  
                System.out.printf("invalid input.\n");
                System.exit(0);
            }

        String vowels = "aeiouAEIOU";
        int result = vowels.indexOf(c);
        if (result != -1)  
            {  
                System.out.printf("%s is a vowel.\n", c);
            }
        else  
            {  
                System.out.printf("%s is not a vowel.\n", c);
            }
    }
}
import java.util.Scanner;

public class example1 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.printf("Enter a single letter: ");

        String c = in.next();
        if (c.length() != 1)
        {
            System.out.printf("invalid input.\n");
            System.exit(0);
        }

        if (c.equals("a") || c.equals("e") || c.equals("i") || c.equals("o") || c.equals("u") || c.equals("A") || c.equals("E") || c.equals("I") || c.equals("O") || c.equals("U"))
        {
            System.out.printf("%s is a vowel.\n", c);
        }
        else
        {
            System.out.printf("%s is not a vowel.\n", c);
        }
    }
}
Boolean Variables and Expressions

Comparing Strings
The boolean Type

• Answers to “questions” are data of type boolean.
• Data of type boolean can only have two values: true, or false.
  – true and false are reserved keywords in Java.

```java
class example1 {
    public static void main(String[] args) {
        double a = 3.2;
        double b = 4.0;
        boolean v1 = (a < 4.3);
        System.out.printf("v1 = %b\n", v1);

        boolean v2 = (a == b);
        System.out.printf("v2 = %b\n", v2);

        boolean v3 = (a != b);
        System.out.printf("v3 = %b\n", v3);
    }
}
```

Output:

```
v1 = true
v2 = false
v3 = true
```
Comparisons of Numbers

- The following operators compare numerical values (of type `double` or `int`), and generate `boolean` results:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
<th>Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>==</code></td>
<td>equals</td>
<td><code>!=</code></td>
<td>not equal to</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>greater than</td>
<td><code>&gt;=</code></td>
<td>greater than or equal to</td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>less than</td>
<td><code>&lt;=</code></td>
<td>less than or equal to</td>
</tr>
</tbody>
</table>

```java
public class example1 {
    public static void main(String[] args) {
        double a = 3.2;
        double b = 4.0;
        System.out.printf("a = %.1f, b = %.1f\n", a, b);
        System.out.printf("a == b: %b\n", a == b);
        System.out.printf("a != b: %b\n", a != b);
        System.out.printf("a > b: %b\n", a > b);
        System.out.printf("a >= b: %b\n", a >= b);
        System.out.printf("a < b: %b\n", a < b);
        System.out.printf("a <= b: %b\n", a <= b);
    }
}
```

Output:

```
a = 3.2, b = 4.0
a == b: false
a != b: true
a > b: false
a >= b: false
a < b: true
a <= b: true
```
Using Parentheses

• When you assign a boolean variable, use parentheses to make it easy to read your code.
• Even if your code runs correctly without parentheses, parentheses are still recommended to make sure you don't get confused.
• Example: setting c equal to the value of "a equals b".

```java
public class example1 {
    public static void main(String[] args) {
        double a = 3.2;
        double b = 4.0;

        boolean c = a == b;  // Correct, but confusing (not recommended!)
        boolean d = (a == b);  // Preferred style (parenthesize)

        System.out.println(c);  // false
        System.out.println(d);  // false
    }
}
```

What is the value of c in this example?

What is the value of d in this example?

They are both equal to false

3.2 is NOT equal to 4.0.
Comparing Numbers: Examples

• Four ways of doing the same comparison (3.2 < 4.0)
• And combining operators: $v5 = (a < 4.3 - 2.6)$;

```java
public class example1 {
    public static void main(String[] args) {
        double a = 3.2;
        double b = 4.0;

        boolean v1 = (a < b);  //first way
        System.out.printf("v1 = %b\n", v1);
        boolean v2 = (a < 4.0);  //second way
        boolean v3 = (3.2 < 4.0);  // 3-rd way
        // 4-th way. Note that v4 does not exist.
        System.out.printf("v4 = %b\n", 3.2 < 4.0);
        boolean v5 = (a < 4.3 - 2.6);
        System.out.printf("v5 = %b\n", v5);
    }
}
```

Output:

$v1 = true$
$v4 = true$
$v5 = false$
Comparing Numbers

common error = / ==

public class example1 {
    public static void main(String[] args) {
        double a = 3.2;
        double b = 4.0;

        //boolean v1 = (a = 3.2);
        boolean v1 = (a == 3.2);
        System.out.printf("v1 = %b\n", v1);
    }
}

Output:

Error (does not run, we need == sign instead of = sign.

Very common error!!!
public class example1 {
    public static void main(String[] args) {
        double x = 2;
        double y = Math.sqrt(2);
        if ((y*y) == 2) {
            System.out.println("equal");
        } else {
            System.out.println("not equal");
        }
    }
}

Output:
not equal

Reason:
y*y is not exactly 2 !!!

Solution:
Replace:  (y*y) == 2
with:
Math.abs((y*y)-2)<0.0000001
Truth Tables for: | |, &&, !

The following logical operators can be used to ask more complex questions. They produce boolean results:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
</tr>
<tr>
<td>a &amp;&amp; b</td>
<td>a AND b</td>
</tr>
<tr>
<td>!a</td>
<td>NOT a</td>
</tr>
</tbody>
</table>

**Syntax**

| a | b | a || b | a && b |
|---|---|-------|-------|
| true | true | true  | true  |
| true | false| false | false |
| false| true | true  | true  |
| false| false| false | false |

**NOT**

<table>
<thead>
<tr>
<th>a</th>
<th>!a</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>
Truth Tables for: | |, &&, !

The following logical operators can be used to ask more complex questions. They produce boolean results:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
</tr>
<tr>
<td>a &amp;&amp; b</td>
<td>a AND b</td>
</tr>
<tr>
<td>!a</td>
<td>NOT a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>OR</th>
<th>AND</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>NOT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>!a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Boolean Expressions Worksheet

<table>
<thead>
<tr>
<th></th>
<th>OR</th>
<th>AND</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>NOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>!a</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>

// What will the following instructions print?

```
int x = 3;
int y = 5;
System.out.println("B1: " + ((x == 3) && (y < 10)) );
System.out.println("B2: " + ((x == 3) && (y > 10)) );
System.out.println("B3: " + ((x == 3) || (y < 10)) );
System.out.println("B4: " + ((x == 3) || (y > 10)) );
System.out.println("B5: " + !(x == 3) );
System.out.println("B6: " + !(x == 4) );
System.out.println("B7: " + ((x == y) && (x + y == 8)) );
System.out.println("B8: " + ((x == y) || (x + y == 8)) );
```
Complicated Use of Operators

• What does this code print?

```java
System.out.println((3 == 5) && (2 < 3) || (3 >= 0));
```

– Use parentheses to make the meaning clear.

```java
((3 == 5) && (2 < 3)) || (3 >= 0)
(3 == 5) && ((2 < 3) || (3 >= 0))
```
Complicated Use of Operators

• What does this code print?

```java
System.out.println((3 == 5) && (2 < 3) || (3 >= 0));
```

– Use parentheses to make the meaning clear.

```
((3 == 5) && (2 < 3)) || (3 >= 0)  \rightarrow \text{true}
(3 == 5) && ((2 < 3) || (3 >= 0))  \rightarrow \text{false}
```
And just when you thought you got it...

```java
// What will these lines print?

// DO not compare strings with == . Gives the wrong result.
String s2 = "Robert";
String s3 = s2.substring(0, 3);
System.out.println("Ex.A1: ", s3 == "Rob" );

// Exceptional case where the '==' works. DO NOT DO THIS.
String s1 = "Rob";
System.out.println("Ex.B1: ", s1 == "Rob" );

// Correct string comparison: use equals
System.out.println("Ex.A2: ", s1.equals("Rob") );
System.out.println("Ex.B2 ", s3.equals("Rob") );
```
## Comparisons of Strings

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1.equals(s2)</td>
<td>s1 equals s2</td>
</tr>
<tr>
<td>s1.compareTo(s2) &lt; 0</td>
<td>s1 comes before s2 in Java's version of alphabetical order</td>
</tr>
<tr>
<td>s1.compareTo(s2) == 0</td>
<td>s1 equals s2</td>
</tr>
<tr>
<td>s1.compareTo(s2) &gt; 0</td>
<td>s1 comes after s2 in Java's version of alphabetical order</td>
</tr>
</tbody>
</table>

**Hint to remember** `compareTo` behavior: think of it as computing `s1-s2`

To compare strings you do the same process as for a word look-up in a dictionary, but for the order of characters look at an ASCII table.
Character ordering

• $0 < 1 < \ldots < 9 < A < B < \ldots < Z < a < b \ldots < z$

• Each character has a ‘code’. This code is used to:
  – Store/represent the character (in binary)
  – Compare characters (based on the value of the code)

• Encoding
  – The mapping between characters and codes
  – There are several encodings: ASCII (old) UTF-8 (popular)
// What will these lines print?
// Discuss WHAT EACH PIECE OF CODE DOES: "6a: " + ("h".compareTo("H") < 0)
System.out.println("1 : " + "hello".equals("hello") );
System.out.println("2 : " + "hello".equals("Hello") );
System.out.println("3a: " + "hello".compareTo("hello") );
System.out.println("3c: " + "world".compareTo("hello") );
System.out.println("3b: " + "hello".compareTo("world") );
System.out.println("4a: " + "h".compareTo("W") );
System.out.println("4b: " + "h".compareTo("World") );
System.out.println("4c: " + "hello".compareTo("World") );
// Add code below so that it prints true for all (e.g. “5a: true”)
System.out.println("5a: " + "hello".compareTo("hello") );
System.out.println("5b: " + "world".compareTo("hello") );
// What will these lines print?
System.out.println("6a: " + ("h".compareTo("H") < 0) );
System.out.println("6b: " + ("h".compareTo("W") < 0) );
System.out.println("6c: " + ("hello".compareTo("World") < 0 ) );
System.out.println("7 : " + ("act".compareTo("actor") < 0 ) );
System.out.println("8 : " + ("97".compareTo("145") < 0 ) );
Modulo operator: %

% computes the **remainder of division**. Returns an int.

- \( a \% b \) gives what remains from \( a \) after removing as many of \( b \) as possible.
- \( a \% b = a \), whenever \( a < b \) (I cannot fit \( b \) in \( a \) at all, therefore all of \( a \) is the remainder.)
  - Example: \( 5 \% 7 \) is 5

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 % 5</td>
<td>2</td>
</tr>
<tr>
<td>5 % 7</td>
<td>5</td>
</tr>
<tr>
<td>39 % 12</td>
<td>3</td>
</tr>
<tr>
<td>100 % 25</td>
<td>0</td>
</tr>
<tr>
<td>125 % 25</td>
<td>0</td>
</tr>
</tbody>
</table>

**Typical Usage**

<table>
<thead>
<tr>
<th>Example applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify odd/even</td>
</tr>
<tr>
<td>No TV on even days.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Identify white squares on a chess board.</td>
</tr>
<tr>
<td>Identify every b-th event</td>
</tr>
<tr>
<td>Every b days I run my Sprinklers. Use ( \text{day} % b == 0 )</td>
</tr>
<tr>
<td>Produce patterns (using loops)</td>
</tr>
<tr>
<td>Produce: <strong>-</strong>-<strong>-</strong> (user chooses length)</td>
</tr>
<tr>
<td>Produce a chess board of size ( n )</td>
</tr>
</tbody>
</table>
More Examples of Conditionals

• Determining if integer K is a divisor of integer N.
• Determining if a day is a weekend.
• Determining if a day is a weekday or a weekend.
• Determining if a month is a summer month.
• Determining the season of a month.
• Determining if a year is a leap year.
• Calculating tax.
• Translating English to Spanish.
  – More accurately: translating a few English words to Spanish.
• Determining the weekday for a date in February 2015.
Example 2: no else program flowchart

```java
import java.util.Scanner;

public class example1 {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.printf("How old are you? ");
        int age = in.nextInt();

        if (age <= 18) {
            System.out.printf("You are a teenager.\n");
            System.out.printf("You are young.\n");
        }
        System.out.printf("Bye\n");
    }
}
```

Note: the else part of an if statement IS OPTIONAL. No else in this example.