

Unit 2: Selection and Iteration

Boolean Expressions and if statements

Adapted from:

- 1) Building Java Programs: A Back to Basics Approach
by Stuart Reges and Marty Stepp
- 2) Runestone CSAwesome Curriculum

Type boolean

- **boolean**: A logical type whose values are `true` and `false`.
 - It is legal to:
 - create a `boolean` variable
 - pass a `boolean` value as a parameter
 - return a `boolean` value from methods
 - call a method that returns a `boolean` and use it as a test

```
int age = 22;  
boolean minor = age < 21; // false  
boolean lovesAPCS = true;  
boolean is1049Prime = isPrime(1049);
```

Relational expressions

- Tests use *relational operators*:

Operator	Meaning	Example	Value
<code>==</code>	equals	<code>1 + 1 == 2</code>	true
<code>!=</code>	does not equal	<code>3.2 != 2.5</code>	true
<code><</code>	less than	<code>10 < 5</code>	false
<code>></code>	greater than	<code>10 > 5</code>	true
<code><=</code>	less than or equal to	<code>126 <= 100</code>	false
<code>>=</code>	greater than or equal to	<code>5.0 >= 5.0</code>	true

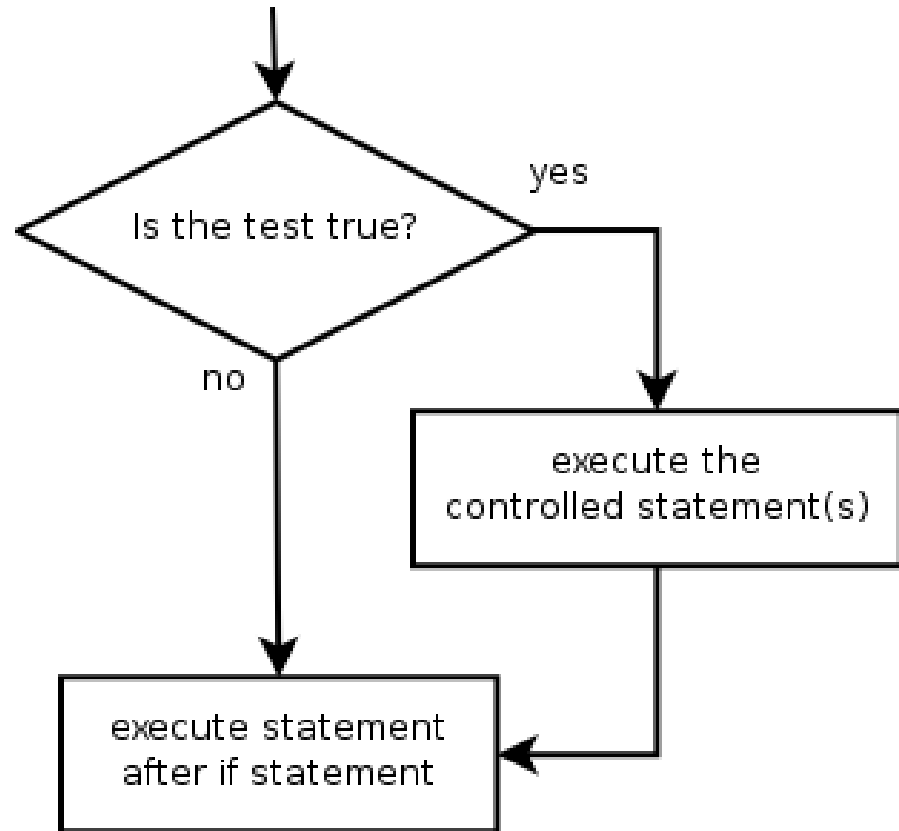
Relational Expressions

```
public class Boolean_Class{  
    public static void main(String[] args){  
        int x = 2, y = 3;  
        System.out.println(x == y); // false  
        System.out.println(x != y); // true  
        System.out.println(2 + 4 * 3 <= 15); // true  
        System.out.println(x > 5); // false  
        System.out.println(y >= 3); // true  
    }  
}
```

The if statement

Executes a block of statements only if a test is true

```
if (test) {  
    statement;  
    ...  
    statement;  
}  
statement;
```



The if statement

```
double gpa = 2.1;  
if (gpa >= 2.0) {  
    System.out.println("Application accepted.");  
}
```

Output:

Application accepted.

```
double gpa = 1.9;  
if (gpa >= 2.0) {  
    System.out.println("Application accepted.");  
}
```

Output: (No output)

The `if/if/if` statement

*executes any block whose condition is true
(independent tests; not exclusive)*

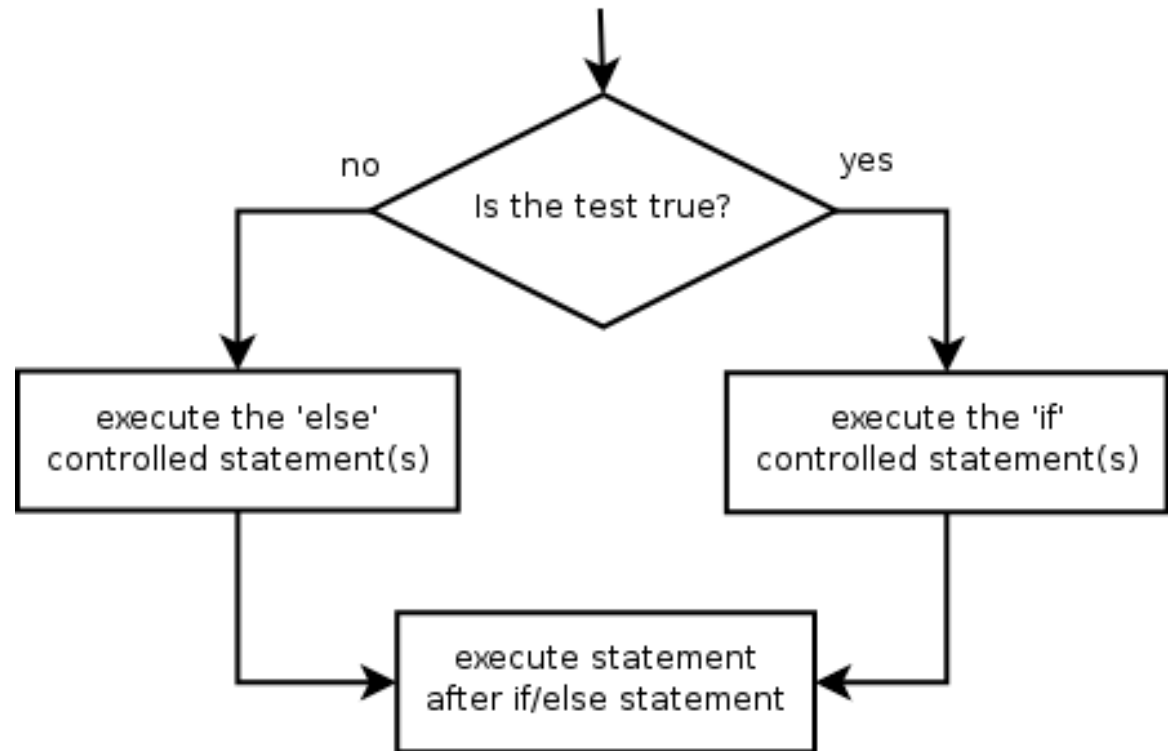
```
if (test) {  
    statement(s);  
}  
if (test) {  
    statement(s);  
}  
if (test) {  
    statement(s);  
}
```

Any, all or none of the above can execute.

The if/else statement

Executes one block if a test is true, another if false

```
if (test) {  
    statement(s);  
} else {  
    statement(s);  
}
```



The if/else statement

```
double gpa = 3.0;  
if (gpa >= 2.0){  
    System.out.println("Welcome to Mars University!");  
} else{  
    System.out.println("Application denied.");  
}
```

Output:

Welcome to Mars University.

The if/else statement

```
double gpa = 1.0;  
if (gpa >= 2.0){  
    System.out.println("Welcome to Mars University!");  
} else{  
    System.out.println("Application denied.");  
}
```

Output:

Application denied.

Misuse of `if`

- What's wrong with the following code?

```
int percent = <Code to ask user to enter a percentage>
```

```
if (percent >= 90) {  
    System.out.println("You got an A!");  
}  
if (percent >= 80) {  
    System.out.println("You got a B!");  
}  
if (percent >= 70) {  
    System.out.println("You got a C!");  
}  
if (percent >= 60) {  
    System.out.println("You got a D!");  
}  
if (percent < 60) {  
    System.out.println("You got an F!");  
}  
...
```

Misuse of if

- What's wrong with the following code?

```
int percent = 90;

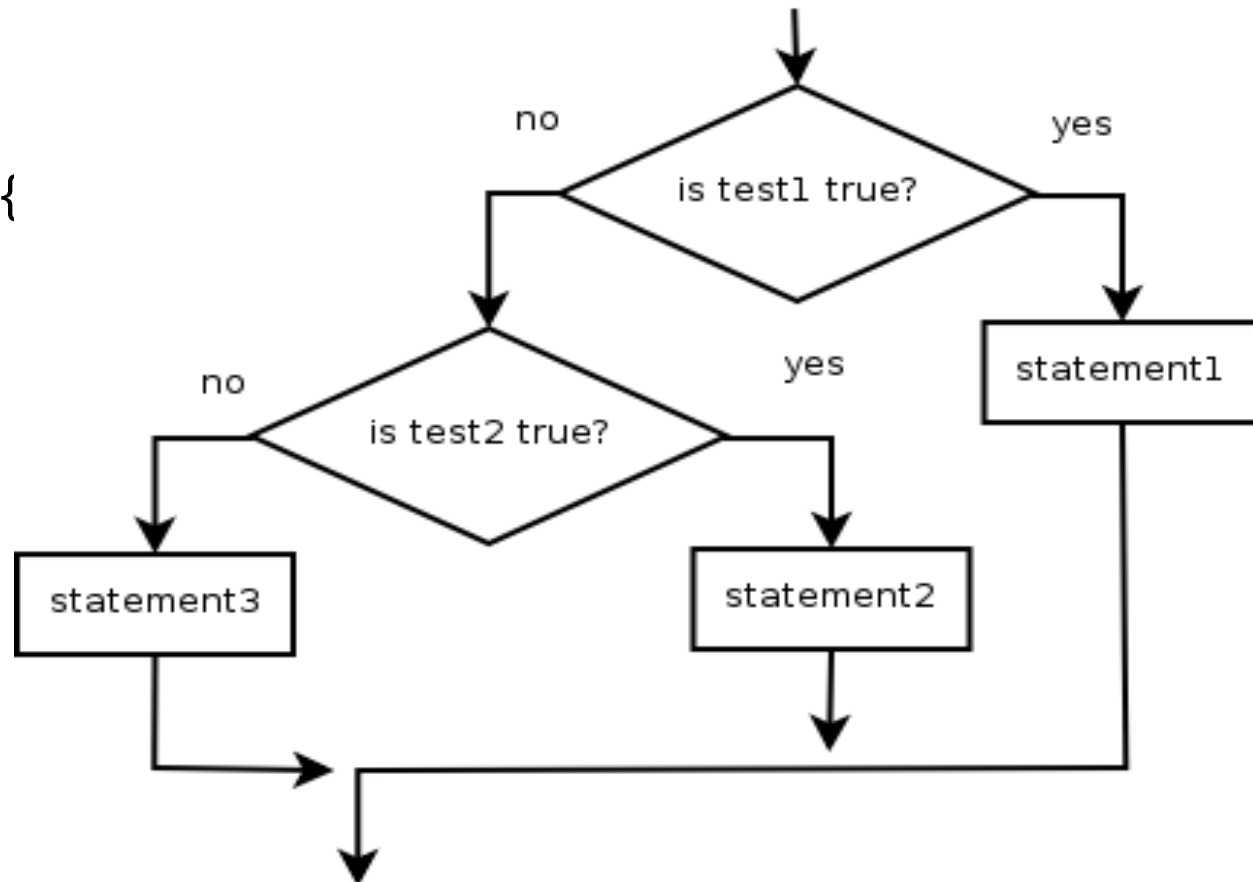
if (percent >= 90) {
    System.out.println("You got an A!");
}
if (percent >= 80) {
    System.out.println("You got a B!");
}
if (percent >= 70) {
    System.out.println("You got a C!");
}
if (percent >= 60) {
    System.out.println("You got a D!");
}
if (percent < 60) {
    System.out.println("You got an F!");
}
...
```

Output:
You got an A!
You got a B!
You got a C!
You got a D!

Nested if/else

Chooses between outcomes using many tests

```
if (test) {  
    statement(s);  
}  
else if (test){ {  
    statement(s);  
}  
else {  
    statement(s);  
}
```



Exactly one of the above can execute.

Nested if/else

```
    int x = 10;  
    if (x > 0) {  
        System.out.println("Positive");  
    }  
    else if (x < 0) {  
        System.out.println("Negative");  
    }  
    else {  
        System.out.println("Zero");  
    }
```

Output:
Positive

Nested if/else

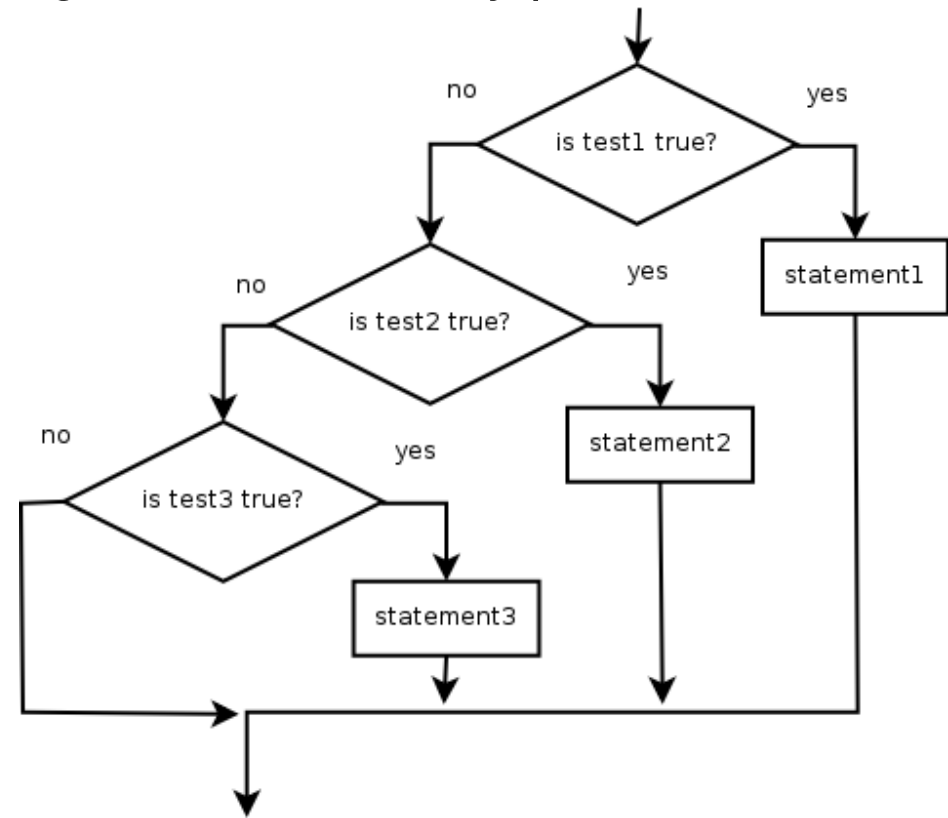
```
    int x = 0;  
    if (x > 0) {  
        System.out.println("Positive");  
    }  
    else if (x < 0) {  
        System.out.println("Negative");  
    }  
    else {  
        System.out.println("Zero");  
    }
```

Output:
Zero

Nested if/else/if

- If it ends with `else`, exactly one path must be taken.
- If it ends with `else if`, the code might not execute any path.

```
if (test) {  
    statement(s);  
}  
else if (test) {  
    statement(s);  
}  
else if (test) {  
    statement(s);  
}
```



One or none of the above can execute.

Nested if/else/if

```
int place = 2;
```

```
if (place == 1) {  
    System.out.println("Gold medal!");  
}  
else if (place == 2) {  
    System.out.println("Silver medal!");  
}  
else if (place == 3) {  
    System.out.println("Bronze medal.");  
}
```

Output:
Silver medal!

Nested if/else/if

```
int place = 6;
```

```
if (place == 1) {  
    System.out.println("Gold medal!");  
}  
else if (place == 2) {  
    System.out.println("Silver medal!");  
}  
else if (place == 3) {  
    System.out.println("Bronze medal.");  
}
```

Output:
No output.

Nested if structures

- exactly 1 path (*mutually exclusive*)

```
if (test) {  
    statement(s);  
}  
else if (test) {  
    statement(s);  
}  
else {  
    statement(s);  
}
```

Most common
use case



- 0 or 1 path (*mutually exclusive*)

```
if (test) {  
    statement(s);  
}  
else if (test) {  
    statement(s);  
}  
else if (test) {  
    statement(s);  
}  
}
```

- 0, 1, or many paths (*independent tests; not exclusive*)

```
if (test) {  
    statement(s);  
}  
if (test) {  
    statement(s);  
}  
if (test) {  
    statement(s);  
}
```

Which nested if/else?

- **(1) if/if/if (2) nested if/else (3) nested if/else if**
 - Whether a user is lower, middle, or upper-class based on income.
 - **(2)** nested if / else if / else
 - Whether you made the dean's list ($\text{GPA} \geq 3.8$) or honor roll (3.5-3.8).
 - **(3)** nested if / else if
 - Whether a number is divisible by 2, 3, and/or 5.
 - **(1)** sequential if / if / if
 - Computing a grade of A, B, C, D, or F based on a percentage.
 - **(2)** nested if / else if / else if / else if /
else

"Boolean Zen", part 1

- Students new to boolean often test if a result is true:

Suppose `isPrime(n)` returns whether `n` is prime (a boolean).

```
if (isPrime(57) == true) {      // bad
    ...
}
```

- But this is unnecessary and redundant. Preferred:

```
if (isPrime(57)) {            // good
    ...
}
```

"Boolean Zen", part 1

- A similar pattern can be used for a false test:

```
if (isPrime(57) == false) {    // bad
```

```
...
```

```
}
```

```
if (!isPrime(57)) {          // good
```

```
...
```

```
}
```

Note: ! is the "not" operator, which flips the boolean value from true to false and false to true.

"Boolean Zen", part 2

- Methods that return boolean often have an if/else that returns true or false:

```
public static boolean odd(int n) {  
    if (n % 2 != 0) {  
        return true;  
    } else {  
        return false;  
    }  
}
```

- But the code above is unnecessarily verbose.

"Boolean Zen", part 3

- We could store the result of the logical test.

```
public static boolean both(int n) {  
    boolean test = n % 2 != 0;  
    if (test) {    // test == true  
        return true;  
    } else {      // test == false  
        return false;  
    }  
}
```

- Notice: Whatever test is, we want to return that.
 - If test is true , we want to return true.
 - If test is false, we want to return false.

Final "Boolean Zen"

- Observation: The `if/else` is unnecessary.
 - The variable `test` stores a boolean value; its value is exactly what you want to return. So return that!

```
public static boolean odd(int n) {  
    boolean test = n % 2 != 0;  
    return test;  
}
```

- An even shorter version:
 - We don't even need the variable `test`.
We can just perform the test and return its result in one step.

```
public static boolean odd(int n) {  
    return n % 2 != 0;  
}
```

"Boolean Zen" template

- Replace

```
public static boolean name(parameters) {  
    if (test) {  
        return true;  
    } else {  
        return false;  
    }  
}
```

- with

```
public static boolean name(parameters) {  
    return test;  
}
```

Lab 1: Day Of the Week

Create a new repl on repl.it. Write a program that outputs the day of the week for a given date! Your program has just the main method and the dayOfWeek method below.

Given the month, m , day, d and year y , the day of the week (Sunday = 0, Monday = 1, ..., Saturday = 6) D is given by:

$$\begin{aligned}y_0 &= y - (14 - m)/12 \\x_0 &= y_0 + y_0/4 - y_0/100 + y_0/400 \\m_0 &= m + 12 \times ((14 - m)/12) - 2 \\D &= (d + x_0 + 31 \times m_0/12) \bmod 7\end{aligned}$$

Note: the division above is integer division. mod is the % operator.

Your program needs one method:

```
public static String dayOfWeek(int m, int d, int y){  
    // fill in code  
}
```

Lab 1: Day Of the Week

Write the main method so that the output is similar to the following: (Use scanner)

Output:

Enter month: 10

Enter day: 15

Enter year: 2019

Day of the week: Tuesday

Use conditionals! And try entering your birthday and test your parents!

References

- 1) Building Java Programs: A Back to Basics Approach by Stuart Reges and Marty Stepp
- 2) Runestone CSAwesome Curriculum:
<https://runestone.academy/runestone/books/published/csawesome/index.html>

For more tutorials/lecture notes in Java, Python, game programming, artificial intelligence with neural networks:

<https://longbaonguyen.github.io>