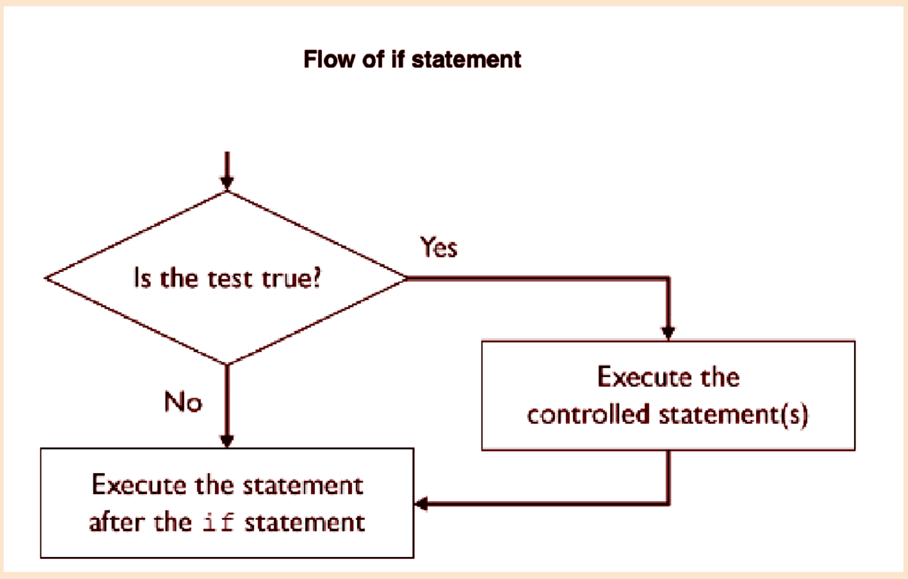
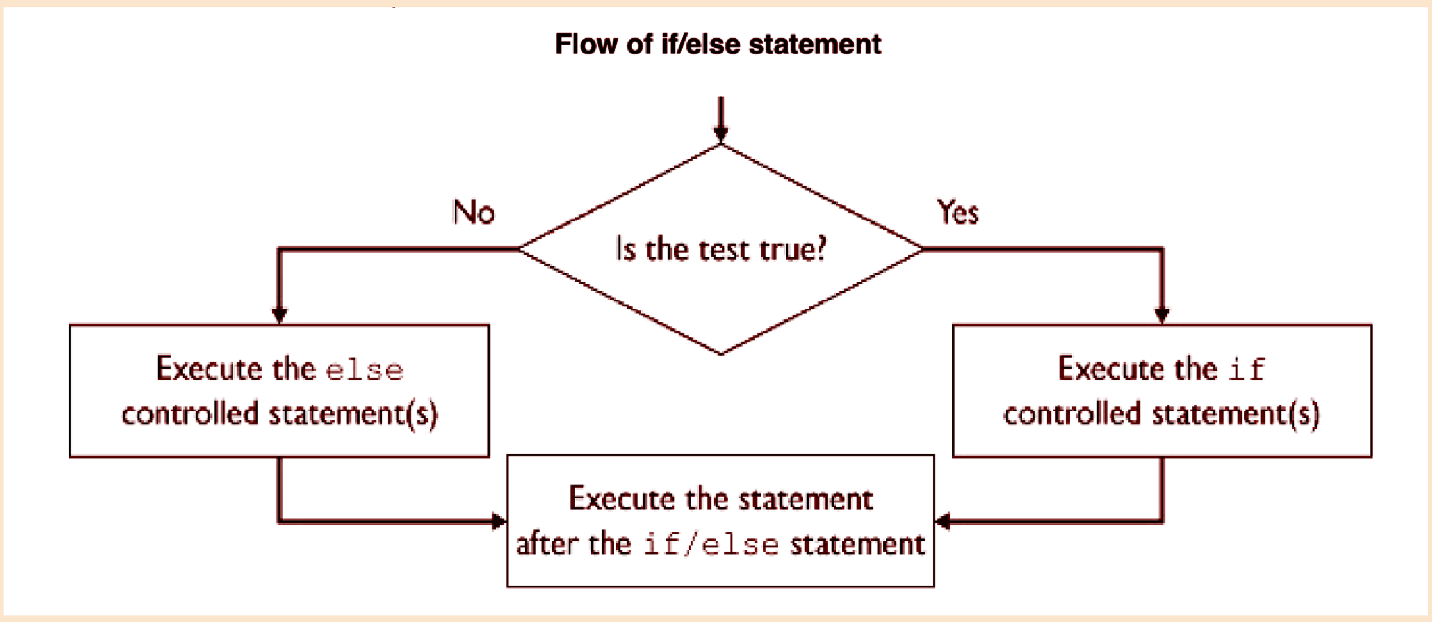
Chapter 4:

The computer performs the test, and if it evaluates to true, the computer executes the controlled statements

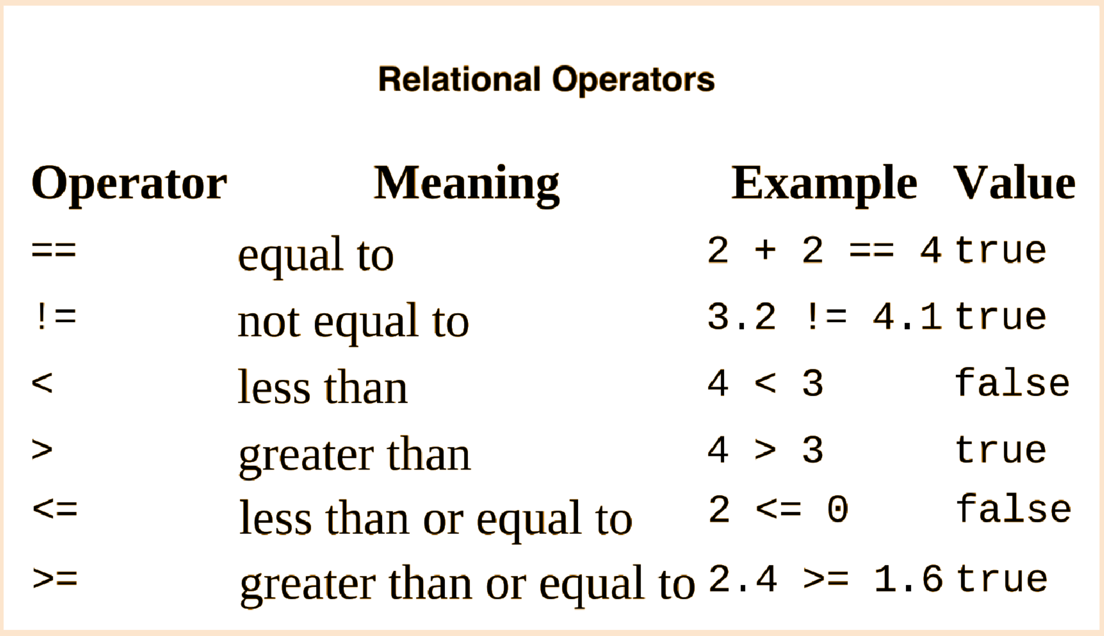


Flow of if statement
Is the test true?
Yes
No
Execute the controlled statement(s)
Execute the statement after the if statement

The computer performs the test and, depending upon whether the code evaluates to true or false, executes one or the other group of statements.



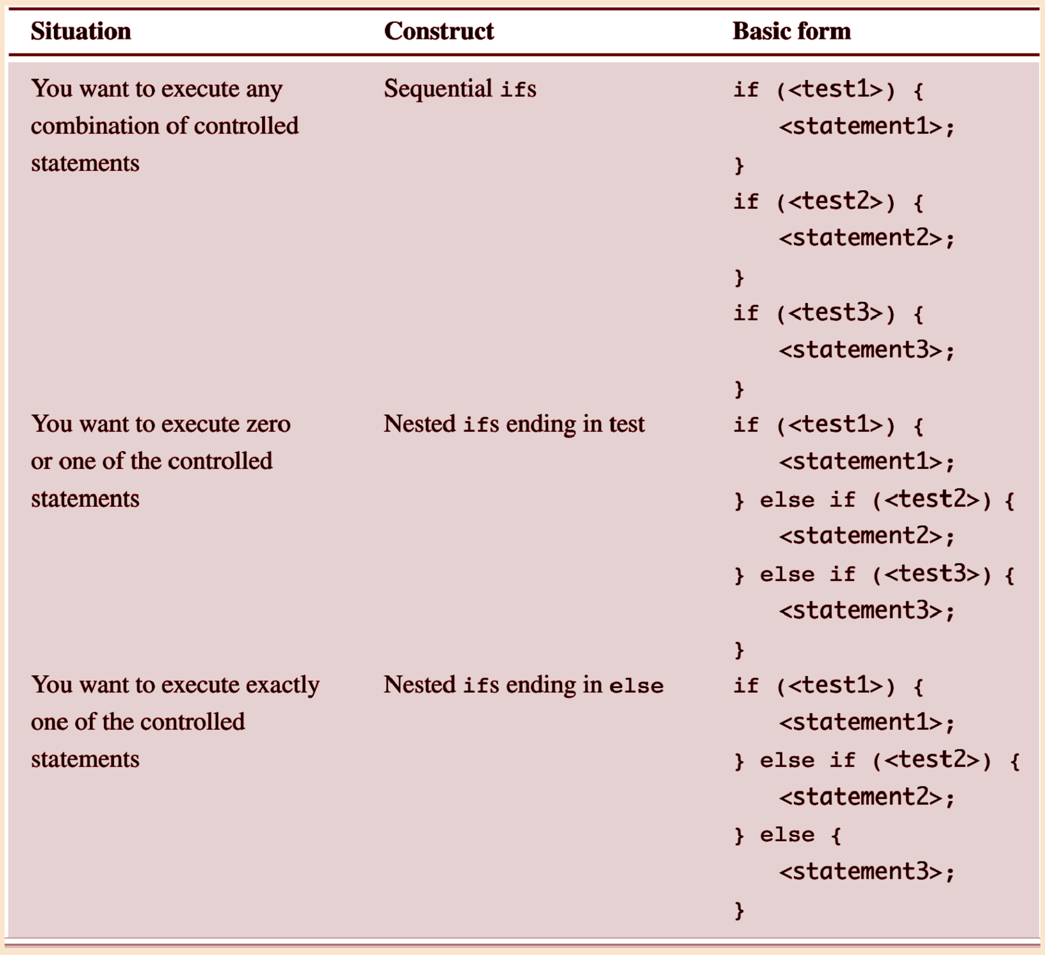
Flow of if/else statement
No
Is the test true?
Yes
Execute the else controlled statement(s)
Execute the if controlled statement(s)
Execute the statement after the if/else statement



Relational Operators

Operator	Meaning	Example	Value
`==`	equal to	`2 + 2 == 4`	`true`
`!=`	not equal to	`3.2 != 4.1`	`true`
`<`	less than	`4 < 3`	`false`
`>`	greater than	`4 > 3`	`true`
`<=`	less than or equal to	`2 <= 0`	`false`
`>=`	greater than or equal to	`2.4 >= 1.6`	`true`

if/else Options



Situation	Construct	Basic form
You want to execute any combination of controlled statements	Sequential `if`s	`if (<test1>) {` `<statement1>;` `}` `if (<test2>) {` `<statement2>;` `}` `if (<test3>) {` `<statement3>;` `}`
You want to execute zero or one of the controlled statements	Nested `if`s ending in test	`if (<test1>) {` `<statement1>;` `} else if (<test2>) {` `<statement2>;` `} else if (<test3>) {` `<statement3>;` `}`
You want to execute exactly one of the controlled statements	Nested `if`s ending in `else`	`if (<test1>) {` `<statement1>;` `} else if (<test2>) {` `<statement2>;` `} else {` `<statement3>;` `}`

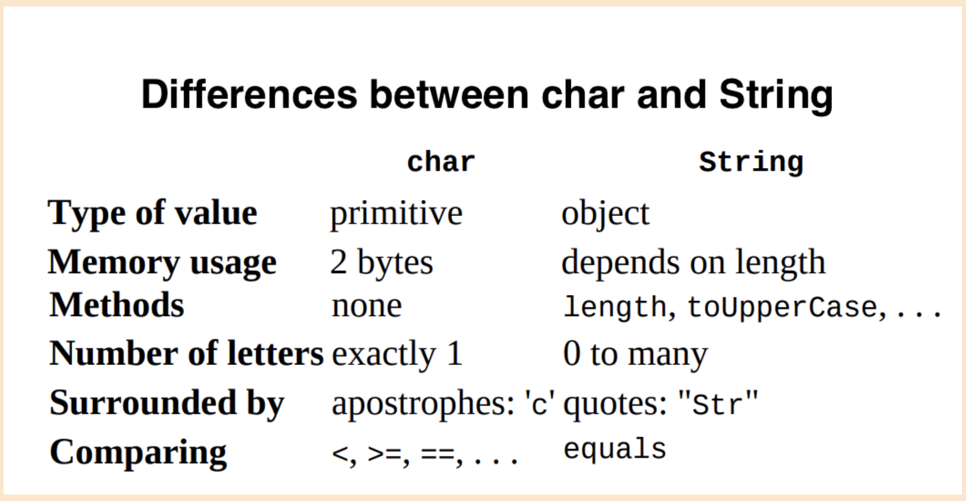
Object Equality: == and != operators do not work the way you might expect when you test for equality of objects like strings. Every Java object has a method called equals that takes another object as an argument. You can use this method to ask an object whether it equals another object

Factoring if/else Statements: Factoring involves moving redundant statements out of the control statements
- You can factor at both the top and the bottom of a construct
- If you notice that the top statement in each branch is the same, you factor it out of the branching part and put it before the branch.
- Similarly, if the bottom statement in each branch is the same, you factor it out of the branching part and put it after the loop

Testing Multiple Conditions: You can combine two tests by using an operator known as the logical AND operator, which is written as two ampersands

Roundoff Error: A numerical error that occurs because floating-point numbers are stored as approximations rather than exact values
- Be aware that when you store floating-point values (e.g., doubles), you are storing approximations and not exact values.
- Don't expect to be able to compare variables of type double for equality.
- We rarely use a test for exact equality when we work with doubles.
- We use the absolute value (abs) method from the Math class to find the magnitude of the difference and then test whether it is less than some small amount (0.001).

Text Processing: Editing and formatting strings of text

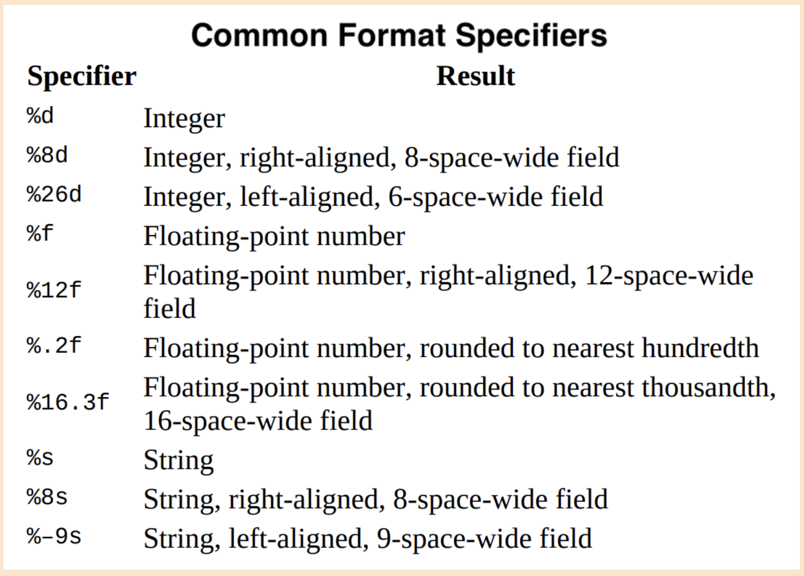


Differences between char and String

	`char`	`String`
Type of value	primitive	object
Memory usage	2 bytes	depends on length
Methods	none	`length`, `toUpperCase`, ...
Number of letters	exactly 1	0 to many
Surrounded by	apostrophes: 'c'	quotes: "`Str`"
Comparing	<, >=, ==, ...	`equals`

char vs int: Values of type char are stored internally as 16-bit integers
- A standard encoding scheme called Unicode determines which integer value represents each character.
  - so ASCII can be seen as a subset of Unicode
- Since chars are really integers, Java automatically converts a value of type char into an int whenever it is expecting an int
- Because values of type char are really integers, they can also be compared by using relational operators such as < or ==

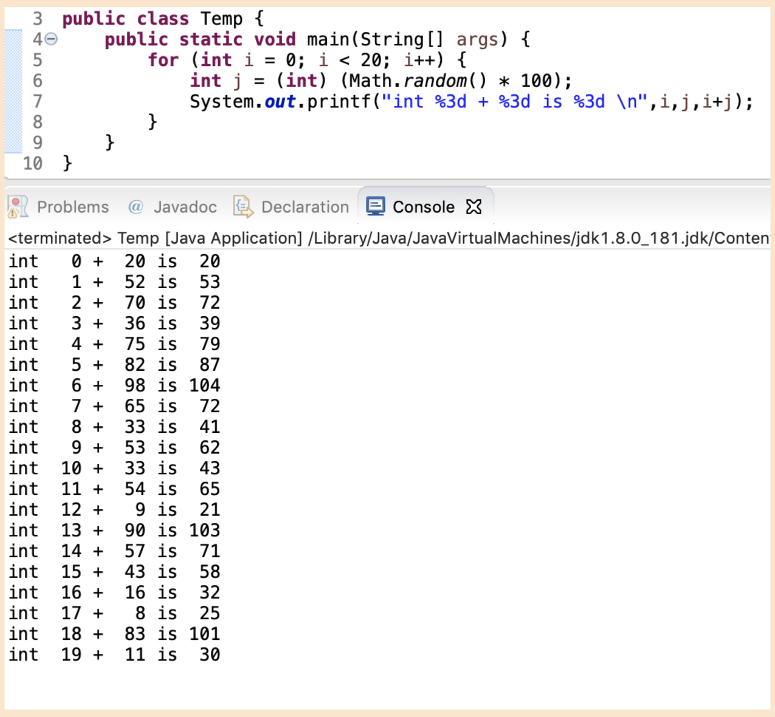
System.out.printf: A format string is like a normal String, except that it can contain placeholders called format specifiers that allow you to specify a location where a variable's value should be inserted, along with the format you'd like to give that value.



Common Format Specifiers

Specifier	Result
`%d`	Integer
`%8d`	Integer, right-aligned, 8-space-wide field
`%26d`	Integer, left-aligned, 6-space-wide field
`%f`	Floating-point number
`%12f`	Floating-point number, right-aligned, 12-space-wide field
`%.2f`	Floating-point number, rounded to nearest hundredth
`%16.3f`	Floating-point number, rounded to nearest thousandth, 16-space-wide field
`%s`	String
`%8s`	String, right-aligned, 8-space-wide field
`%-9s`	String, left-aligned, 9-space-wide field

Example of using printf



```java
public class Temp {
    public static void main(String[] args) {
        for (int i = 0; i < 20; i++) {
            int j = (int) (Math.random() * 100);
            System.out.printf("int %3d + %3d is %3d \n",i,j,i+j);
        }
    }
}
```

```
<terminated> Temp [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_181.jdk/Conten
int   0 +  20 is  20
int   1 +  52 is  53
int   2 +  70 is  72
int   3 +  36 is  39
int   4 +  75 is  79
int   5 +  82 is  87
int   6 +  98 is 104
int   7 +  65 is  72
int   8 +  33 is  41
int   9 +  53 is  62
int  10 +  33 is  43
int  11 +  54 is  65
int  12 +   9 is  21
int  13 +  90 is 103
int  14 +  57 is  71
int  15 +  43 is  58
int  16 +  16 is  32
int  17 +   8 is  25
int  18 +  83 is 101
int  19 +  11 is  30
```

Precondition: A condition that must be true before a method executes in order to guarantee that the method can perform its task.

Postcondition: A condition that the method guarantees will be true after it finishes executing, as long as the preconditions were true before the method was called.

Throwing Exceptions:

- Exceptions are objects. Before you can throw an exception, you have to construct an exception object using new
-
```
if (condition) {
        throw new IllegalArgumentException("message");
}
```