Building Java Programs

Chapter 4: Conditional Execution

Lecture outline

- conditional execution
  - the `if` statement and the `if/else` statement
  - relational expressions
  - nested `if/else` statements
- subtleties of conditional execution
  - factoring `if/else` code
  - methods with conditional execution: revisiting return values

if/else statements

reading: 4.2

The if statement

- **if statement**: Executes a block of statements only if a certain condition is true.
  - Otherwise, the block of statements is skipped.
  - General syntax:

```
if (<condition>) {
    <statement> ;
    <statement> ;
    ...
    <statement> ;
}
```

- Example:

```
double gpa = console.nextDouble();
if (gpa >= 2.0) {
    System.out.println("Your application is accepted.");
}
```

The if/else statement

- **if/else statement**: Executes one block of statements if a certain condition is true, and another if it is false.
  - General syntax:

```
if (<condition>) {
    <statement(s)> ;
} else {
    <statement(s)> ;
}
```

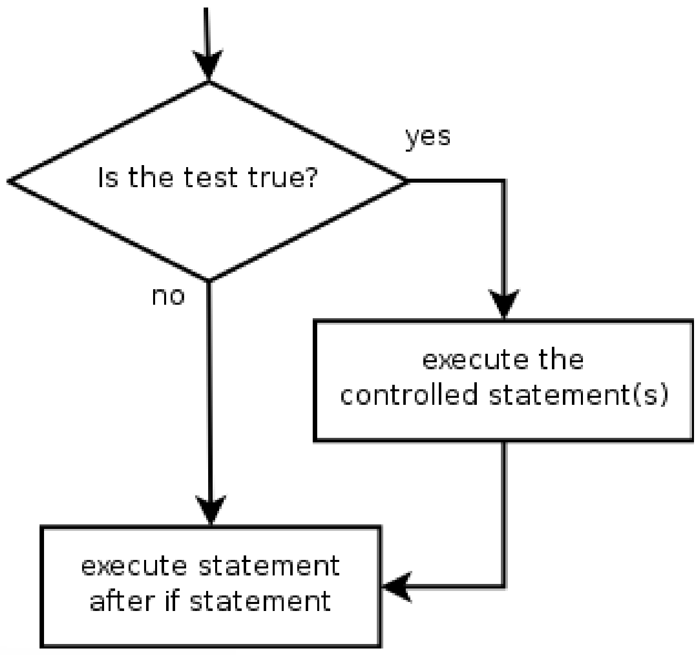
- Example:

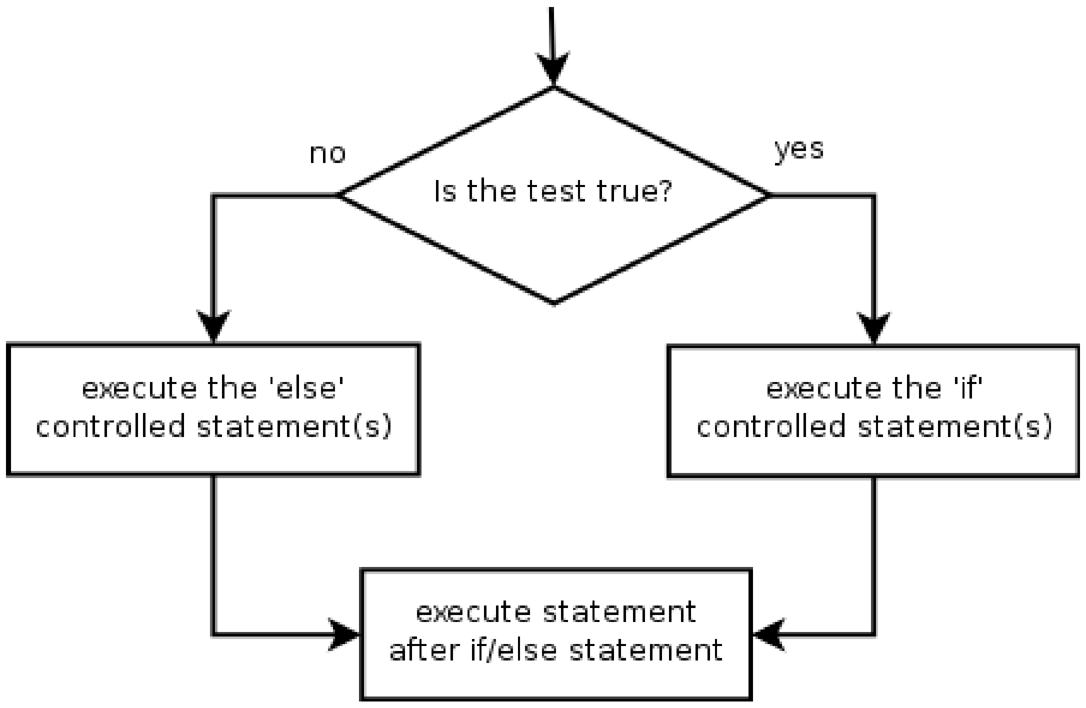
```
double gpa = console.nextDouble();
if (gpa >= 2.0) {
    System.out.println("Welcome to Mars University!");
} else {
    System.out.println("Your application is denied.");
}
```

if statement flow diagrams

```
if (<condition>) {
    <statement>;
    <statement>;
    ...
    <statement>;
}
```

```
if (<condition>) {
    <statement(s)>;
} else {
    <statement(s)>;
}
```


Is the test true?
yes
no
execute the controlled statement(s)
execute statement after if statement


no
Is the test true?
yes
execute the 'else' controlled statement(s)
execute the 'if' controlled statement(s)
execute statement after if/else statement

Relational expressions

- The ***<condition>*** in an `if` or `if/else` statement is the same kind as in a `for` loop.

```
for (int i = 1; i <= 10; i++) {

if (i <= 10) {
```

  - The conditions are actually of type `boolean`, seen in Ch. 5.

- These conditions are called *relational expressions* and use the following *relational operators*:

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

Logical operators && || !

- Conditions can be combined using *logical operators*:

Operator	**Description**	**Example**	**Result**
`&&`	and	`(9 != 6) && (2 < 3)`	`true`
`||`	or	`(2 == 3) || (-1 < 5)`	`true`
`!`	not	`!(7 > 0)`	`false`

- "Truth tables" for each operator,
  when used with logical values *p* and *q*:

p	**q**	**p && q**	**p || q**
`true`	`true`	`true`	`true`
`true`	`false`	`false`	`true`
`false`	`true`	`false`	`true`
`false`	`false`	`false`	`false`

p	**!p**
`true`	`false`
`false`	`true`

Evaluating logic expressions

- Relational operators have lower precedence than math operators.

```
5 * 7 >= 3 + 5 * (7 - 1)
5 * 7 >= 3 + 5 * 6
35    >= 3 + 30
35    >= 33
true
```

- Relational operators cannot be "chained" as they can in algebra.

```
2 <= x <= 10                         (assume that x is 15)
true   <= 10
   error!
```

- Instead, combine multiple tests with && or ||

```
2 <= x && x <= 10                    (assume that x is 15)
true   && false
   false
```

Logical questions

- What is the result of each of the following expressions?

```
int x = 42;
int y = 17;
int z = 25;
```

  - `y < x && y <= z`
  - `x % 2 == y % 2 || x % 2 == z % 2`
  - `x <= y + z && x >= y + z`
  - `!(x < y && x < z)`
  - `(x + y) % 2 == 0 || !((z - y) % 2 == 0)`

- Answers: `true`, `false`, `true`, `true`, `false`

Loops with if/else

- if/else statements can be used with loops or methods:

```
Scanner console = new Scanner(System.in);
System.out.print("Type 10 numbers: ");

int nonNegative = 0;
int negative = 0;

for (int i = 1; i <= 10; i++) {
    int next = console.nextInt();
    if (next >= 0) {
        nonNegative++;
    } else {
        negative++;
    }
}

System.out.println(nonNegative + " non-negative");
System.out.println(negative + " negative");
```

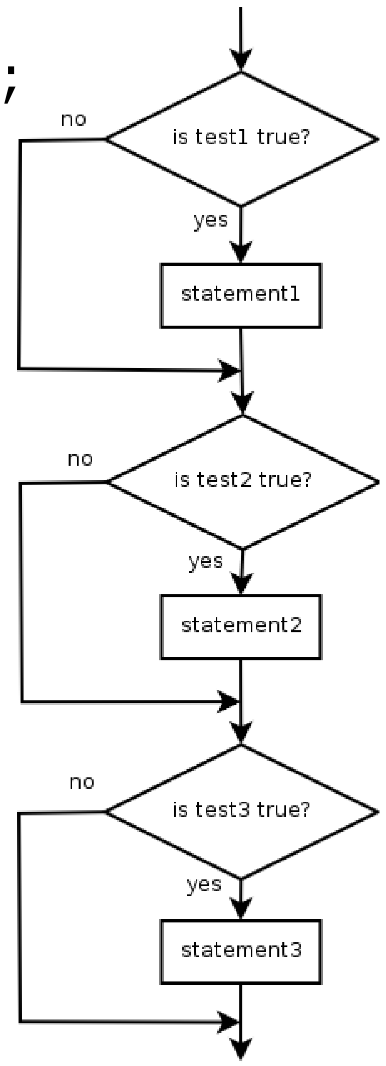
Nested if/else statements

reading: 4.2

"Sequential if" bug

- Many students new to `if/else` write code like this:

```
Scanner console = new Scanner(System.in);
System.out.print("What percentage did you earn? ");
int percent = console.nextInt();
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!");
}
else {
    System.out.println("You got an F!");
}
...
```

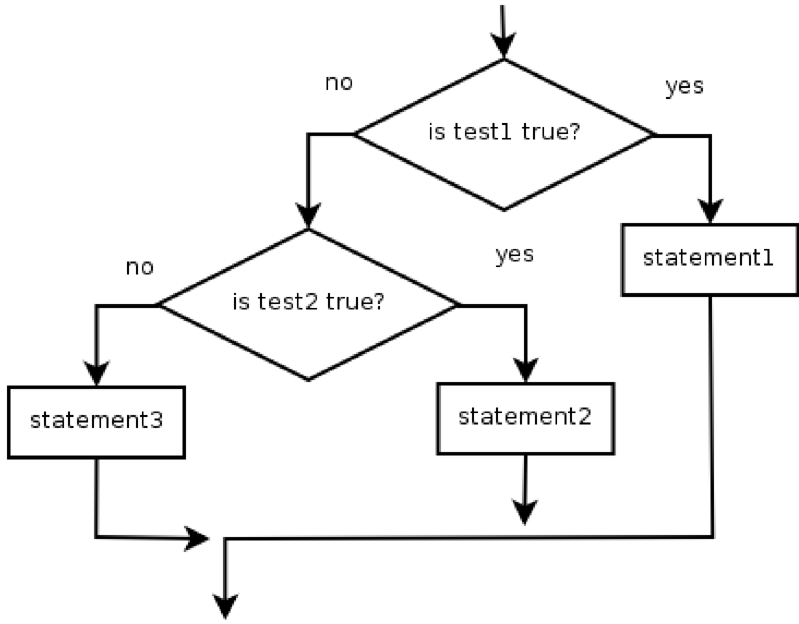


- What's the bug?

Nested if/else

- **nested if/else statement**: A chain of `if/else` that chooses between outcomes using many conditions.
  - General syntax:

```
if (<condition>) {
    <statement(s)> ;
} else if (<condition>) {
    <statement(s)> ;
} else {
    <statement(s)> ;
}
```

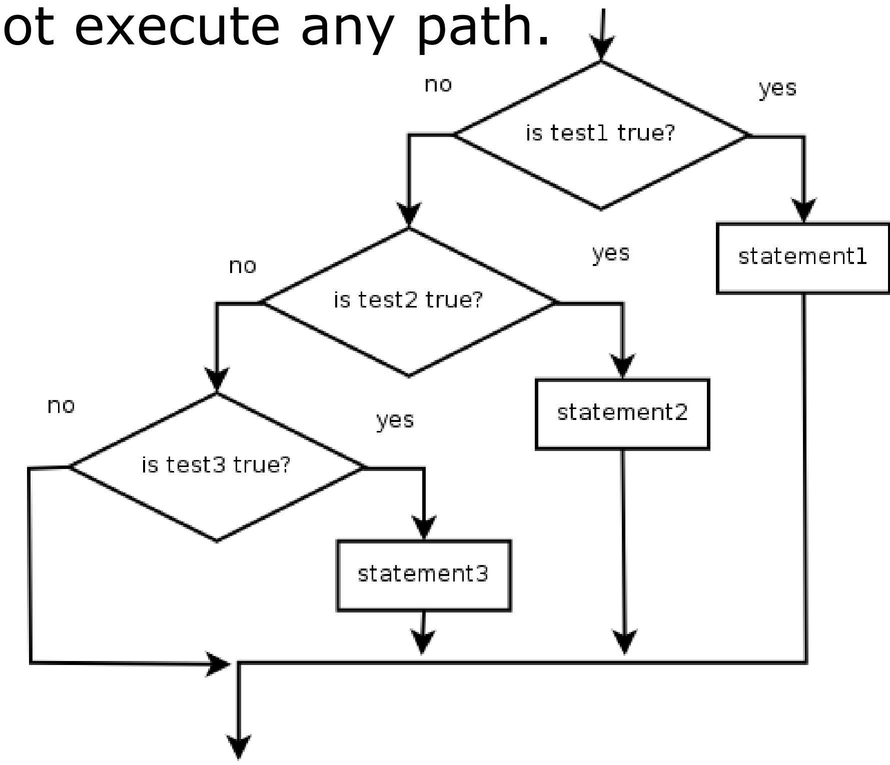


- Example:

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

Nested if/else/if

- A nested `if/else` can end with an `if`.
  - If it ends with `else`, one code path must be taken.
  - If it ends with `if`, the program might not execute any path.



  - General syntax:

```
if (<condition>) {
    <statement(s)> ;
} else if (<condition>) {
    <statement(s)> ;
} else if (<condition>) {
    <statement(s)> ;
}
```

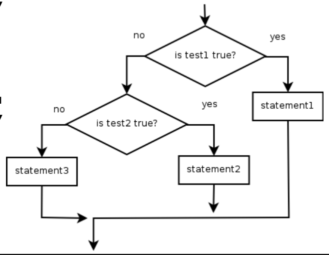
- Example ending with `if`:

```
if (place == 1) {
    System.out.println("You win the gold medal!");
} else if (place == 2) {
    System.out.println("You win a silver medal!");
} else if (place == 3) {
    System.out.println("You earned a bronze medal.");
}
```

Structures of if/else code

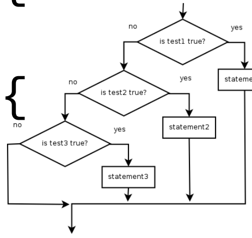
- Choose 1 of many paths:
  (conditions are mutually exclusive)

```
if (<condition>) {
    <statement(s)>;
} else if (<condition>) {
    <statement(s)>;
} else {
    <statement(s)>;
}
```



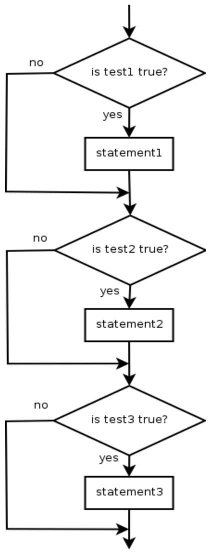
- Choose 0 or 1 of many paths:
  (conditions are mutually exclusive and any action is optional)

```
if (<condition>) {
    <statement(s)>;
} else if (<condition>) {
    <statement(s)>;
} else if (<condition>) {
    <statement(s)>;
}
```



- Choose 0, 1, or many of many paths:
  (conditions/actions are independent of each other)

```
if (<condition>) {
    <statement(s)>;
}
if (<condition>) {
    <statement(s)>;
}
if (<condition>) {
    <statement(s)>;
}
```



Which nested if/else to use?

- Which `if/else` construct is most appropriate?
  - Reading the user's GPA and printing whether the student is on the dean's list (3.8 to 4.0) or honor roll (3.5 to 3.8).
    - **nested `if / else if`**
  - Printing whether a number is even or odd.
    - **simple `if / else`**
  - Printing whether a user is lower-class, middle-class, or upper-class based on their income.
    - **nested `if / else if / else`**
  - Reading a number from the user and printing whether it is divisible by 2, 3, and/or 5.
    - **sequential `if / if / if`**
  - Printing a user's grade of A, B, C, D, or F based on their percentage in the course.
    - **nested `if / else if / else if / else if / else`**

Nested if/else problem

- Modify our BMI program from a previous lecture so that it prints information about each person's BMI according to the table at right.

BMI	Status
below 18.5	underweight
18.5 - 24.9	normal
25 - 29.9	overweight
30 and up	obese

Produce the following output:

```
This program reads in data for two people
and computes their body mass index (BMI)
and weight status.

Enter next person's information:
height (in inches)? 62.5
weight (in pounds)? 130.5
normal

Enter next person's information:
height (in inches)? 58.5
weight (in pounds)? 90
underweight

Person #1 body mass index = 23.485824
Person #2 body mass index = 18.487836949375414
Difference = 4.997987050624587
```

Methods with if/else and return

reading: 4.5

if/else with return

- Methods can be written to return different values under different conditions using `if/else` statements:

```
// Returns the largest of the three given integers.
public static int max3(int a, int b, int c) {
    if (a >= b && a >= c) {
        return a;
    } else if (b >= c && b >= a) {
        return b;
    } else {
        return c;
    }
}
```

  - Whichever path the code enters, it will return the appropriate value. Returning a value causes a method to immediately exit.
  - All paths through the code must reach a `return` statement, or the code will not compile.

All code paths must return

- Not returning a value in every path is an error:

```
public static int max3(int a, int b, int c) {
    if (a >= b && a >= c) {
        return a;
    } else if (b >= c && b >= a) {
        return b;
    }
    // Error; not all paths return a value. What if c is max?
}
```

- Surprisingly, the following code also does not compile:

```
public static int max3(int a, int b, int c) {
    if (a >= b && a >= c) {
        return a;
    } else if (b >= c && b >= a) {
        return b;
    } else if (c >= a && c >= b) {
        return c;
    }
}
```

  - To our eyes, it seems that all paths do return a value.
  - The compiler thinks `if/else/if` code might skip all the paths.

if/else return question

- Write a method `countFactors` that returns the number of factors of an integer.
  - For example, `countFactors(60)` returns 12 because 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, and 60 are factors of 60.

- Write a program that prompts the user for a maximum integer and prints all prime numbers up to that max.

```
Maximum number? 52
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
15 primes (28.846 %)
```

Method return answer 1

```
// Prompts for a maximum number and prints each prime up to that maximum.
import java.util.*;

public class Primes {
    public static void main(String[] args) {
        // read max from user
        Scanner console = new Scanner(System.in);
        System.out.print("Maximum number? ");
        int max = console.nextInt();
        printPrimes(max);
    }

    // Prints all prime numbers up to the given maximum.
    public static void printPrimes(int max) {
        int primes = 0;
        for (int i = 2; i <= max; i++) {
            if (countFactors(i) == 2) {       // i is prime
                System.out.print(i + " ");
                primes++;
            }
        }
        System.out.println();

        double percent = 100.0 * primes / max;
        System.out.printf("%d primes (%.3f %%)\n", primes, percent);
    }
```

Method return answer 2

```
    ...

    // Returns how many factors the given number has.
    public static int countFactors(int number) {
        int count = 0;
        for (int i = 1; i <= number; i++) {
            if (number % i == 0) {
                count++;  // i is a factor of number
            }
        }
        return count;
    }
}
```

Factoring if/else code

reading: 4.3

Factoring if/else code

- **factoring**: extracting common/redundant code
  - Factoring `if/else` code reduces the size of the `if` and `else` statements and can sometimes eliminate the need for `if/else` altogether.

- Example:

```
if (a == 1) {
    x = 3;
} else if (a == 2) {
    x = 6;
    y++;
} else {  // a == 3
    x = 9;
}
```

$\longrightarrow$

```
x = 3 * a;
if (a == 2) {
    y++;
}
```

Code in need of factoring

- The following example has a lot of redundant code:

```
if (money < 500) {
    System.out.println("You have, $" + money + " left.");
    System.out.print("Caution!  Bet carefully.");
    System.out.print("How much do you want to bet? ");
    bet = console.nextInt();
} else if (money < 1000) {
    System.out.println("You have, $" + money + " left.");
    System.out.print("Consider betting moderately.");
    System.out.print("How much do you want to bet? ");
    bet = console.nextInt();
} else {
    System.out.println("You have, $" + money + " left.");
    System.out.print("You may bet liberally.");
    System.out.print("How much do you want to bet? ");
    bet = console.nextInt();
}
```

Code after factoring

- Here is an improved ("factored") version of the same code:

```
System.out.println("You have, $" + money + " left.");

if (money < 500) {
    System.out.print("Caution!  Bet carefully.");
} else if (money < 1000) {
    System.out.print("Consider betting moderately.");
} else {
    System.out.print("You may bet liberally.");
}

System.out.print("How much do you want to bet? ");
bet = console.nextInt();
```

- Factoring tips:
  - If the start of each branch is the same, move it *before* the `if/else`.
  - If the end of each branch is the same, move it *after* the `if/else`.
  - If similar but not identical code exists in each branch, look for patterns.