CSE 1223: Introduction to Computer Programming in Java
Chapter 3 – Branching
Flow of Control

- The order in which statements in a program are executed is called the *flow of control*.
- So far we have only seen sequential execution: statements execute one after the other in the order in which they appear in the program.
Consider the following tasks:

- You want to compute the quotient of two variables but only if the divisor is not zero
- You input some value (e.g., a date) and if it is in the correct format (mm/dd/yyyy) you continue the computation, otherwise you print an error
- Given a grade between 0 and 100, you want to convert the numeric value to a letter grade (e.g., A for grade greater than 90, B for grade between 80 and 90, etc.)

How can we check these conditions and execute the appropriate piece of code depending on the outcome of the check?
Scanner in = new Scanner(System.in);
System.out.print("Enter the dividend: ");
int dividend = in.nextInt();
System.out.print("Enter the (non-zero) divisor: ");
int divisor = Integer.parseInt(in.nextLine());
if (divisor != 0) {
    int quotient = dividend / divisor;
    System.out.println(dividend + " / " + divisor + " = " + quotient);
}
else {
    System.out.println("Cannot divide by 0");
}
The **boolean** Type

- A variable of the **boolean** data type stores one of two values: **true** or **false**
- **true** and **false** are the only two boolean constants
- boolean values/expressions are used to make decisions in programs
- For example:
  ```java
  if (x > 0) // boolean expression
  {
    System.out.println("x is positive");
  }
  ```
Boolean Expressions

There are several kinds of boolean-valued expressions:

- a boolean variable or constant, e.g.,
  ```java
  boolean boolVar = in.nextBoolean();
  if (boolVar) { ... }
  ```

- an arithmetic expression followed by a relational operator followed by an arithmetic expression, e.g.,
  ```java
  int intVar = in.nextInt();
  if (intVar > 0) { ... }
  ```
Relational Operators

- `==` (equal) \( x == y \)
- `!=` (not equal) \( x != y \)
- `>` \( x > y \)
- `<` \( x < y \)
- `>=` \( x >= y \)
- `<=` \( x <= y \)
Boolean Operators

We can also build boolean expressions by combining two boolean expressions with a boolean operator:

- `&&` (and) \((x > 0) \&\& (x < 10)\)
- `||` (or) \((x <= 0) \|\| (x >= 10)\)
- `!` (not) \(! (x == 0)\)
Boolean Operators cont.

- If A and B are boolean expressions, 
  A && B is true if and only if both A and B are true (in other words, if either A or B or both are false, A && B is false)

- If A and B are boolean expressions, 
  A || B is true if either A or B or both are true (in other words, A || B is false only if both A and B are false)

- If A is a boolean expression, !A is true if A is false, and !A is false if A is true
Some Boolean Expressions

What’s the value of each of the following expressions:

```plaintext
int x = 5, y = 12;
boolean a = true, b = false, c = true;

- (x > 0) && (x < 10)
- (x <= 0) || (x >= 10)
- ! (a && b && c)
- (a || b || c)
- ((x - 1) == ((y / 5) + (y % 5)))
- ((x != y) || ! (x == y))
```
Your Turn

Given three integer variable, $i$, $j$, and $k$, write a boolean expression for each of the following problems:

- $i$ is equal to 3 or 5
- $i$ is between 1 and 7 (but not 1 or 7)
- $i$ is even
- $i$ is odd
- $i$ is the smallest of $i$, $j$, and $k$
If: Syntax and Flow Chart

boolean expression

if ( test )
{
 if_block
}

statement sequence

test

true

false

if_block
If-Else: Syntax and Flow Chart

```
if ( test )
{
  if_block
}
else
{
  else_block
}
```

boolean expression

```
statement sequences
```

```
if_block
else_block
```

```
test
true
false
```

```
if_block
else_block
```

13
An Example

Given two integers $i$ and $j$, write a piece of code that sets integer variable $max$ to the value of the larger of the two.

```c
if (i > j){
    max = i;
}
else { // i <= j
    max = j;
}
```
Tracing an if-else statement

- To trace an if-else statement, trace only the portions of the statement that get executed
  - Base this on an evaluation of the boolean expression
**A trace with an initial state**

<table>
<thead>
<tr>
<th>Program Line</th>
<th>Program state</th>
</tr>
</thead>
<tbody>
<tr>
<td>i = 10</td>
<td>i = 10</td>
</tr>
<tr>
<td>j = 5</td>
<td>j = 5</td>
</tr>
<tr>
<td>max = 0</td>
<td>max = 0</td>
</tr>
</tbody>
</table>

```java
if (i > j) {
    max = i;
} else {
    max = j;
}
```
## A trace with an initial state

<table>
<thead>
<tr>
<th>Program Line</th>
<th>Program state</th>
</tr>
</thead>
<tbody>
<tr>
<td>i = 10</td>
<td>i = 10</td>
</tr>
<tr>
<td>j = 5</td>
<td>j = 5</td>
</tr>
<tr>
<td>max = 0</td>
<td>max = 10</td>
</tr>
</tbody>
</table>

### if (i > j) {
  max = i;
}

### else {
  max = j;
}

### }
A trace with an initial state

<table>
<thead>
<tr>
<th>Program Line</th>
<th>Program state</th>
</tr>
</thead>
<tbody>
<tr>
<td>i = 10</td>
<td>i = 10</td>
</tr>
<tr>
<td>j = 5</td>
<td>j = 5</td>
</tr>
<tr>
<td>max = 0</td>
<td>max = 0</td>
</tr>
<tr>
<td>if (i &gt; j) {</td>
<td></td>
</tr>
<tr>
<td>max = i;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>else {</td>
<td></td>
</tr>
<tr>
<td>max = j;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>

```java
int i = 10;
int j = 5;
int max = 0;
if (i > j) {
    max = i;
} else {
    max = j;
}
```

```java
i = 10
j = 5
max = 10
```
## A trace with an initial state

<table>
<thead>
<tr>
<th>Program Line</th>
<th>Program state</th>
</tr>
</thead>
<tbody>
<tr>
<td>i = 10</td>
<td>i = 10</td>
</tr>
<tr>
<td>j = 5</td>
<td>j = 5</td>
</tr>
<tr>
<td>max = 0</td>
<td>max = 0</td>
</tr>
</tbody>
</table>

```java
if (i > j) {
    max = i;
}
else {
    max = j;
}
```

Note that this portion gets skipped over with these inputs!
A trace with another initial state

<table>
<thead>
<tr>
<th>Program Line</th>
<th>Program state</th>
</tr>
</thead>
<tbody>
<tr>
<td>i = 10</td>
<td>i = 10</td>
</tr>
<tr>
<td>j = 500</td>
<td>j = 500</td>
</tr>
<tr>
<td>max = 0</td>
<td>max = 0</td>
</tr>
<tr>
<td>if (i &gt; j) {</td>
<td></td>
</tr>
<tr>
<td>max = i;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>else {</td>
<td></td>
</tr>
<tr>
<td>max = j;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>

```java
if (i > j) {
    max = i;
}
else {
    max = j;
}
```
A trace with another initial state

<table>
<thead>
<tr>
<th>Program Line</th>
<th>Program state</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>i = 10</td>
</tr>
<tr>
<td></td>
<td>j = 500</td>
</tr>
<tr>
<td></td>
<td>max = 0</td>
</tr>
<tr>
<td>if (i &gt; j) {</td>
<td></td>
</tr>
<tr>
<td>max = i;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>else {</td>
<td></td>
</tr>
<tr>
<td>max = j;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>i = 10</td>
<td></td>
</tr>
<tr>
<td>j = 500</td>
<td></td>
</tr>
<tr>
<td>max = 500</td>
<td></td>
</tr>
</tbody>
</table>
### A trace with another initial state

<table>
<thead>
<tr>
<th>Program Line</th>
<th>Program state</th>
</tr>
</thead>
<tbody>
<tr>
<td>i = 10</td>
<td>i = 10</td>
</tr>
<tr>
<td>j = 500</td>
<td>j = 500</td>
</tr>
<tr>
<td>max = 0</td>
<td>max = 500</td>
</tr>
</tbody>
</table>

```java
if (i > j) {
    max = i;
} else {
    max = j;
}
```

```java
i = 10
j = 500
max = 500
```
A trace with another initial state

<table>
<thead>
<tr>
<th>Program Line</th>
<th>Program state</th>
</tr>
</thead>
<tbody>
<tr>
<td>i = 10</td>
<td>i = 10</td>
</tr>
<tr>
<td>j = 500</td>
<td>j = 500</td>
</tr>
<tr>
<td>max = 0</td>
<td>max = 500</td>
</tr>
<tr>
<td>if (i &gt; j) {</td>
<td></td>
</tr>
<tr>
<td>max = i;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>else {</td>
<td></td>
</tr>
<tr>
<td>max = j;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>

And this time it’s THIS portion that gets skipped with these inputs!
An alternative solution

- Given two integers $i$ and $j$, write a piece of code that sets integer variable $max$ to the value of the larger of the two.

```c
max = i;
if (j > max)
{
    max = j;
}
```

- Trace through the code above?
  - Does it do the same thing as the previous code?
  - Sometimes a good choice of initial value can make for simpler code.
Problem solving with if-else

- Suppose we have an integer $i$ and we need to know whether it is even or not
  - Recall the % operation
    - $i$ is even if $i \% 2 == 0$
  - This suggests a choice:
    - If $i \% 2 == 0$, then $i$ is even
    - Otherwise, $i$ is odd

- Problems that can be solved with if-else often have this form:
  - “If $x$ is true then perform action $y$ otherwise perform action $z$”
Problem-solving with if-else

- Given an integer \( i \), write some code that will set a string to “even” if \( i \) is even, and “odd” if \( i \) is odd.
Problem-solving with if-else

- Given an integer $i$, write some code that will set a string to “even” if $i$ is even, and “odd” if $i$ is odd.

  ```java
  String result = "";
  ```
Problem-solving with if-else

Given an integer \( i \), write some code that will set a string to “even” if \( i \) is even, and “odd” if \( i \) is odd.

```java
String result = "";
if ( ) {
    // \( i \) is even
    // set result to “even”
}
else{
    // \( i \) is odd
    // set result to “odd”
}
```
Problem-solving with if-else

Given an integer \( i \), write some code that will set a string to “even” if \( i \) is even, and “odd” if \( i \) is odd.

```java
String result = "";
if ( ) {
    // i is even
    // set result to “even”
}
else {
    // i is odd
    // set result to “odd”
}
```

What goes here?
Problem-solving with if-else

Given an integer $i$, write some code that will set a string to “even” if $i$ is even, and “odd” if $i$ is odd.

```java
String result = "";
if ( i % 2 == 0 ) {
    // i is even
    // set result to “even”
}
else {
    // i is odd
    // set result to “odd”
}
```

Our boolean expression test!
Problem-solving with if-else

Given an integer $i$, write some code that will set a string to “even” if $i$ is even, and “odd” if $i$ is odd.

```java
String result = "";
if ( i % 2 == 0 ) {
    // $i$ is even
    // set result to “even”
}
else{
    // $i$ is odd
    // set result to “odd”
}
```
Problem-solving with if-else

- Given an integer $i$, write some code that will set a string to “even” if $i$ is even, and “odd” if $i$ is odd.

```java
String result = "";
if ( i % 2 == 0 ) {
    // i is even
    result = "even";
}
else{
    // i is odd
    result = "odd";
}
```

The code that performs this action!
Another Example (Alternative)

- Given an integer \( i \), write a piece of code that sets a string to "even" or "odd" depending on whether the value of \( i \) is even or odd.

  - Sometimes a good choice of initial value can make your code simpler

```java
String result = "odd";
if ((i % 2) == 0) // i is even
{
    result = "even";
}
```

- Trace the above code
  - Is it the same as what was on the previous slide?
Your Turn

- Given a string \( str \) and a boolean variable \( \text{startsWithA} \), write a segment of code that sets \( \text{startsWithA} \) to \text{true} if the first character of \( str \) is a capital \( ‘A’ \), and \text{false} otherwise.
Nested ifs

- Given three integers $i, j, \text{ and } k$, write a piece of code that sets integer variable $max$ to the value of the largest of the three.
  - What does this logic look like?
Nested ifs

Given three integers $i$, $j$, and $k$, write a piece of code that sets integer variable $max$ to the value of the largest of the three.

What does this logic look like?

- If $i$ is bigger than $j$, then set $max$ to the larger of $i$ and $k$
- Otherwise set $max$ to the larger of $j$ and $k$
Nested ifs

```c
if (i > j) {
    // set max to the larger of i and k
}
else {
    // i <= j
    // set max to the larger of j and k
}
```
Nested ifs

```java
if (i > j){
    // set max to the larger of i and k
}
else {
    // i <= j
    // set max to the larger of j and k
}
```

How do we do this?
Nested if statements:

```java
if (i > j) {
    // set max to the larger of i and k
}
else {
    // i <= j
    // set max to the larger of j and k
}
```

How do we do this?
We’ve already done it!
Nested ifs

```c
if (i > j) {
    if (i > k) { // i is max
        max = i;
    }
    else {      // k is max
        max = k;
    }
} else {
    // i <= j
    // set max to the larger of j and k
}
```

How do we do this?

We’ve already done it!

Just use the logic for finding the max of two values here!
Nested ifs

```c
if (i > j){
    if (i > k) { // i is max
        max = i;
    }
    else {  // k is max
        max = k;
    }
}
else {  // i <= j
    // set max to the larger of j and k
}
```

How do we do this?
Nested ifs

```java
if (i > j){
    if (i > k) { // i is max
        max = i;
    }
    else { // k is max
        max = k;
    }
}
else { // i <= j
    // set max to the larger of j and k
}
```

How do we do this?

We’ve already done it!
Nested ifs

```java
if (i > j){
   if (i > k) { // i is max
      max = i;
   }
   else { // k is max
      max = k;
   }
}
else { // i <= j
   if (j > k) // j is max
   { // j is max
      max = j;
   }
   else // k is max
   { // k is max
      max = k;
   }
}
```

How do we do this?
We’ve already done it!
Just use the logic for finding the max of two values here!
Max Of Three

```c
if (i > j)
{
    if (i > k) // i is max
    {
        max = i;
    }
    else // k is max
    {
        max = k;
    }
}
else // i <= j
{
    if (j > k) // j is max
    {
        max = j;
    }
    else // k is max
    {
        max = k;
    }
}
```
Max Of Three

```c
if (i > j)
{
    if (i > k) // i is max
    {
        max = i;
    }
    else       // k is max
    {
        max = k;
    }
}
else       // i <= j
{
    if (j > k) // j is max
    {
        max = j;
    }
    else       // k is max
    {
        max = k;
    }
}
```
Given an integer, \textit{grade}, holding a grade between 0 and 100, write a piece of code that converts the numeric value to a letter grade according to the following table and prints the letter grade.

<table>
<thead>
<tr>
<th>grade ≥ 90</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>80 ≤ grade &lt; 90</td>
<td>B</td>
</tr>
<tr>
<td>70 ≤ grade &lt; 80</td>
<td>C</td>
</tr>
<tr>
<td>60 ≤ grade &lt; 70</td>
<td>D</td>
</tr>
<tr>
<td>grade &lt; 60</td>
<td>E</td>
</tr>
</tbody>
</table>
Given a String, s, which is meant to hold a date in the \texttt{mm/dd/yyyy} format, check that it is in the correct format and print an error message if it is not.