CSE 142

Making Decisions

Overview

- Quick Review:
  - Classes, objects, methods, messages
- Today:
  - Making decisions – if statements
  - Decision trees
  - Comparison operators
  - Boolean operators
- Reading
  - Dugan notes ch. 10
  - Niño & Hosch ch. 6

Decisions – Conditional Execution

- So far, we only have the ability to execute statements (including method calls) one after the other
- Almost any real program needs to be able to make decisions during execution
  - Check whether there's enough money in the account for a withdrawal request
  - If it's dark outside, turn on the lights
  - If the temperature is less than 68, turn on the furnace
  - Make sure a House doesn't move off the edge of the screen
- Java, like any interesting programming language, allows statements to be executed *conditionally*, depending on the value of some *boolean (test) expression*

Buying Beer

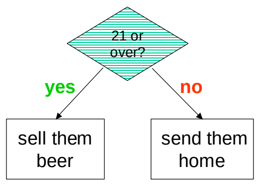
- Convenience store owners use this algorithm:

  Check the buyer's ID.
  If it says they are at least 21, sell them beer, otherwise send them home.

  - How can we say this in Java?

A Decision Tree

- It helps to visualize our algorithm:



Expressing a Decision Tree in Java

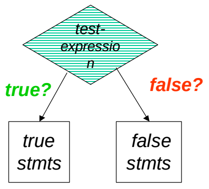
- Expressing this in Java

```
/** Sell beer to the customer if the age on his/her ID is >= 21
 * @param person  Object representing the customer */
public void checkID(Customer person) {
    // get person's age
    int age = person.ageOnID( );
    // sell beer if over 21
    if (age >= 21) {
        this.sellBeer( );
    } else {
        this.sendHome(person);
    }
}
```

The if Statement Pattern

• We use this pattern for our decisions:

```
if (<test-expression>) {
    <statements to do if test is true>
} else {
    <statements to do if test is false>
}
```



• *<test-expression>* is any expression of boolean type.

• Can have one, several, or even zero statements in each *branch* of the if statement.

Comparing Numbers

• Java provides operators for comparing numbers:

Math Symbol	Java Operator	Meaning	Example	Value if y is 11
>	>	greater than	y > 5	
<	<	less than	y < 5	
≥	>=	greater than or equal to	y >= 11	
≤	<=	less than or equal to	y <= 10	
≠	!=	not equal to	y != 5	
=	==	equal to	y == 5	

• Comparison expressions compute answers of type boolean.

• Comparison operators have lower precedence than + etc.

An Enhanced Algorithm

• This is closer to the real algorithm used:

If the customer looks at least 30, sell them beer.
Otherwise, check the buyer's ID.
If it says they are at least 21, sell them beer, otherwise send them home.

• Draw the decision tree:

The Algorithm in Java

• Here's the enhanced algorithm again:

If the customer looks at least 30, sell them beer.
Otherwise, check the buyer's ID.
If it says they are at least 21, sell them beer, otherwise send them home.

• Implement it in Java:

Checking a Series of Conditions

• If you're checking a series of conditions, and taking the first one that's true, then we can write it more compactly.

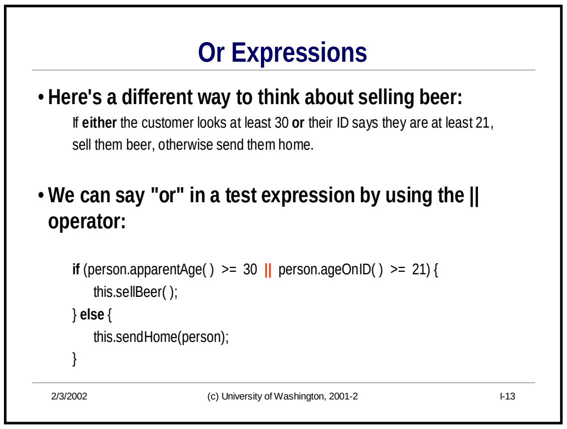
```
if (person.apparentAge( ) >= 30) {
    this.sellBeer( );
} else if (person.ageOnID( ) >= 21) {
    this.sellBeer( );
} else {
    this.sendHome(person);
}
```

The if-else if-else Statement Pattern

• We use this pattern for testing a series of conditions:

```
if (<first-test-expression>) {
    <statements to do if first test is true>
} else if (<next-test-expression>) {
    <statements to do if previous test is false but this test is true>
} else if (<next-test-expression>) {
    <statements to do if previous tests are false but this test is true>
...
} else {
    <statements to do if all tests are false>
}
```

• Draw decision tree.



Or Expressions

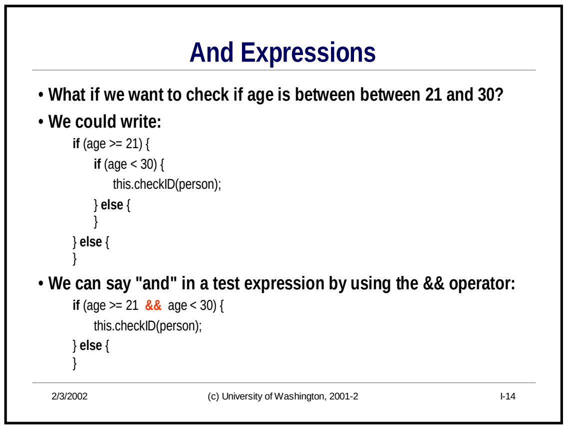
- Here's a different way to think about selling beer:

  If **either** the customer looks at least 30 **or** their ID says they are at least 21, sell them beer, otherwise send them home.

- We can say "or" in a test expression by using the || operator:

```
if (person.apparentAge( )  >=  30  ||  person.ageOnID( )  >=  21) {
    this.sellBeer( );
} else {
    this.sendHome(person);
}
```

2/3/2002 (c) University of Washington, 2001-2 I-13



And Expressions

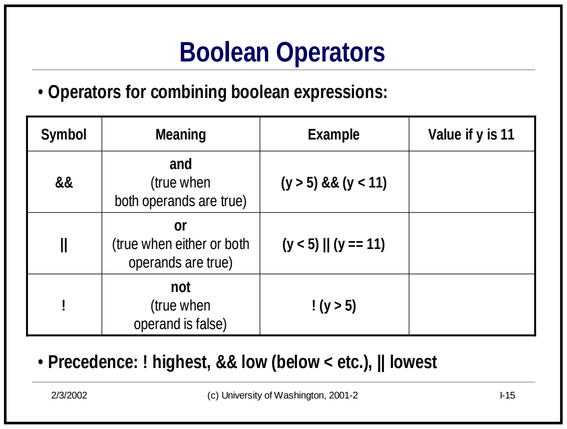
- What if we want to check if age is between between 21 and 30?
- We could write:

```
if (age >= 21) {
    if (age < 30) {
        this.checkID(person);
    } else {
    }
} else {
}
```

- We can say "and" in a test expression by using the && operator:

```
if (age >= 21  &&  age < 30) {
    this.checkID(person);
} else {
}
```

2/3/2002 (c) University of Washington, 2001-2 I-14



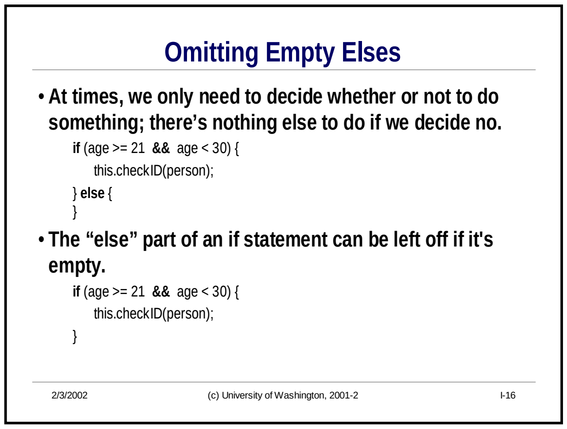
Boolean Operators

- Operators for combining boolean expressions:

Symbol	Meaning	Example	Value if y is 11
&&	**and** (true when both operands are true)	(y > 5) && (y < 11)	
||	**or** (true when either or both operands are true)	(y < 5) || (y == 11)	
!	**not** (true when operand is false)	! (y > 5)	

- Precedence: ! highest, && low (below < etc.), || lowest

2/3/2002 (c) University of Washington, 2001-2 I-15



Omitting Empty Elses

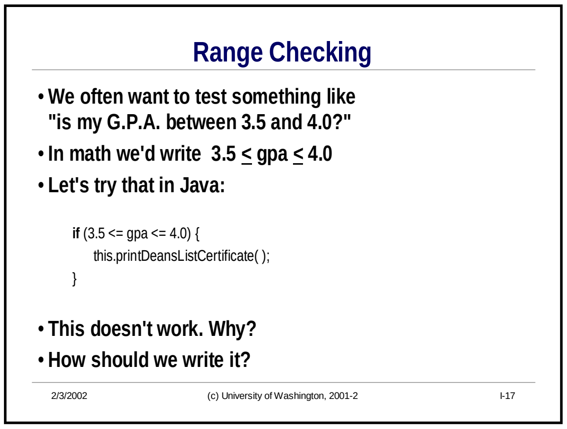
- At times, we only need to decide whether or not to do something; there's nothing else to do if we decide no.

```
if (age >= 21  &&  age < 30) {
    this.checkID(person);
} else {
}
```

- The "else" part of an if statement can be left off if it's empty.

```
if (age >= 21  &&  age < 30) {
    this.checkID(person);
}
```

2/3/2002 (c) University of Washington, 2001-2 I-16



Range Checking

- We often want to test something like "is my G.P.A. between 3.5 and 4.0?"
- In math we'd write 3.5 ≤ gpa ≤ 4.0
- Let's try that in Java:

```
if (3.5 <= gpa <= 4.0) {
    this.printDeansListCertificate( );
}
```

- This doesn't work. Why?
- How should we write it?

2/3/2002 (c) University of Washington, 2001-2 I-17