CSE 142 Summer 2001

Making Decisions

Introduction

- Today:
  - Making decisions
  - Decision trees
  - Relational operators, boolean operators

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
- Java, like any interesting programming language allows statements to be executed *conditionally*, depending on the value of some *boolean (logical, true/false) expression*

Buying Beer

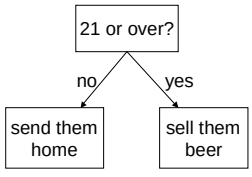
- Convenience store owners use this algorithm:

  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 in Java

- We can write this in Java:

```
int age = input.readInt("Enter customer's age: ")
if (age >= 21) {
  System.out.println("Sell beer");
} else {
  System.out.println("Send person home");
}
```

The if-statement Pattern

- We use this pattern for our decisions:

```
if (<condition-expression>) {
  <then-statements>
} else {
  <else-statements>
}
```

- <condition-expression> is a boolean expression
- (Reminder: boolean is a type in Java; boolean expressions are anything that evaluates to true or false)

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	y >= 11	
≤	<=	less than or equal	y <= 10	
≠	!=	not equal	y != 5	
=	==	equal	y == 5	

Refinement

- This is closer to the real algorithm used:

  If the customer looks over 31, sell them beer. Otherwise, check their ID. If it says they are over 21, sell them beer, otherwise send them home.

  - Draw the decision tree:

The Refinement in Java

- Here's the algorithm again:

  If the customer looks over 31, sell them beer. Otherwise, check their ID. If it says they are over 21, sell them beer, otherwise send them home.

  - Implement it in Java:

Combining Boolean Expressions

- What if we want to check if a number is between 20 and 45?

```
if (x > 20) {
  if (x < 45) {
    // do something. . .
  } else { . . . }
} else { . . . }
```

- Java allows us state this more directly:

```
if (x > 20 && x < 45) {
  // do something.
}
```


Boolean Operators

- Operators for combining boolean expressions:

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

Checking for Range

- We often want to express something like: 10 < x < 20
- Let's try it:

```
if (10 < x < 20) { . . .}
```

  - This doesn't work, why?
- We need to use boolean operators:

```
if (10 < x && x < 20) { . . . }
```


The if-statement Pattern - Variation

- At times, we only need to decide whether or not to do something; there's nothing else to do if we decide no
- The "else" part of the if statement can be omitted in this case

```
if (<condition-expression>) {
  <then-statements>
}
```

  - Meaning: if <condition-expression> is true, execute <then-statements>, otherwise do nothing

```
if (temperature > 80) {
    System.out.println("Too hot for Seattle natives!");
}
```