# Basics of Java: Expressions & Statements

Nathaniel Osgood CMPT 858 February 15, 2011

## Java as a Formal Language

- Java supports many "constructs" that serve different functions
  - Class & Interface declarations
  - Importing references to classes from other code libraries
  - Defining methods

# Methods

- Methods are "functions" associated with a class
- Methods can do either or both of
  - Computing values
  - Performing actions
    - Printing items
    - Displaying things
    - Changing the state of items
- Consist of two pieces
  - Header: Says what "types" the method expects as arguments and returns as values, and exceptions that can be thrown
  - Body: Describes the algorithm (code) to do the work (the "implementation")

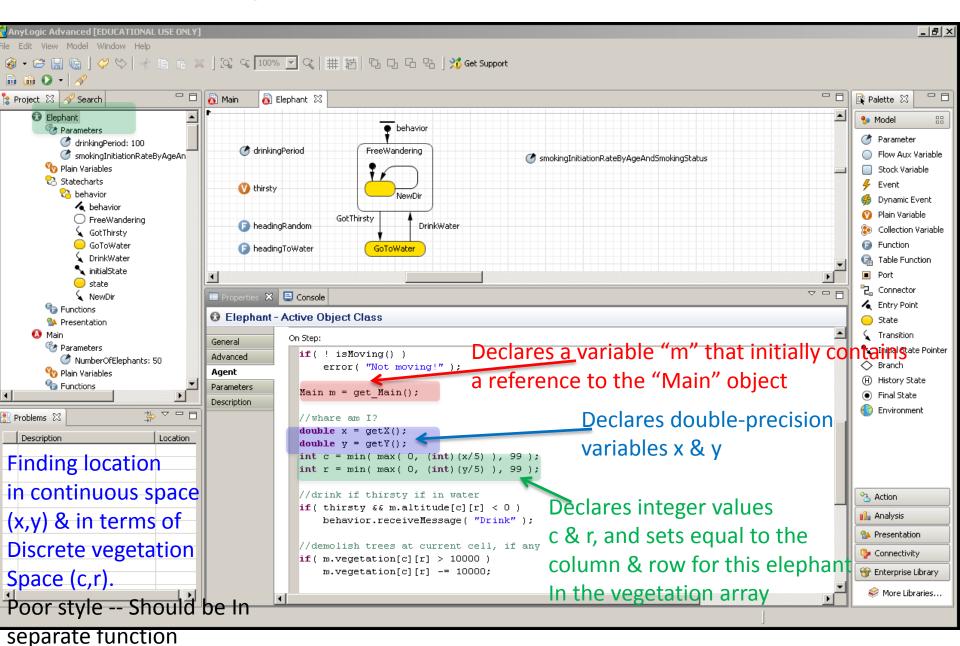
# **Method Bodies**

- Method bodies consist of
  - Variable Declarations
  - Statements
- Statements are "commands" that do something (effect some change), for example
  - Change the value of a variable or a field
  - Return a value from the function
  - Call a method
  - Perform another set of statements a set of times
  - Based on some condition, perform one or another set of statements

# Variable Declarations

- Variables in Java are associated with "types" and can contain values
  - The types describe the sort of values that a variable can contain (the set of possible values)
  - E.g.
    - double: Double precision floating point numbers
    - int: (positive & negative): Integer values within some range
    - String: A (reference to a) text sequence
    - boolean: A dichotomous value, holding "true", or "false"
- When we "declare" a variable, we indicate its name & type – and possibly an initial value

### **Example Variable Declarations**



## **Common Java Statements**

- if
- for
- while / do-while
- Try-Catch-Finally
- Throw (Trigger) exception
- An expression (typically side-effecting)
  - Assignment
  - Call to a function
- Composite statement block (multiple statements enclosed in a "{}")

## **Common Java Expressions**

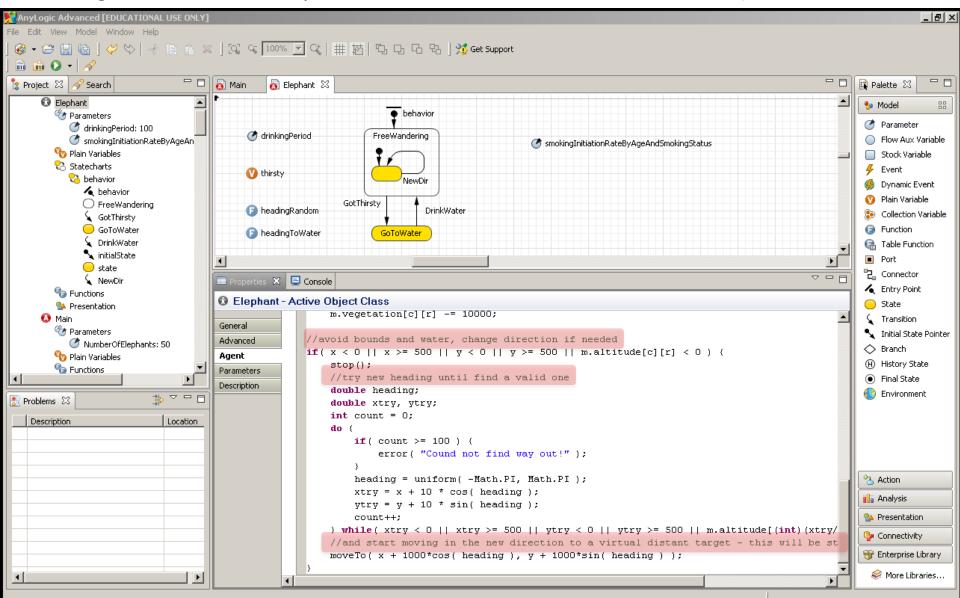
- Literal (3.5, 1, "my string", null)
- Causes changes Side effecting)
  - Assignment (a=b) Left hand side is some location (variable, field, etc.)
- Comparison (a>b,a==b)
- Mathematical Operators (+,-,/,\*) Can be "overloaded" to mean other things (e.g. + as concatenation)
- Method call (function call): this.get\_Main()
- "Dereferencing": Looking up field or value *b* in an object expression *a*: (*a*.*b*)
- Ternary operator: (predicate ? a : b)

## Comments

- Comments in Java are indicated in two different ways
  - Arbitrarily long: Begun with /\* and ended with \*/
    - These can span many lines
  - Within a line: after a //
- Use comments to describe your intentions!

#### Rerouting Around Barriers (Boundaries & Water) Poor Style – entire logic, conditions (checks on boundaries, whether water) & rerouting

Logic should all be in separate functions from this & from each other). Remove constants



#### For statements

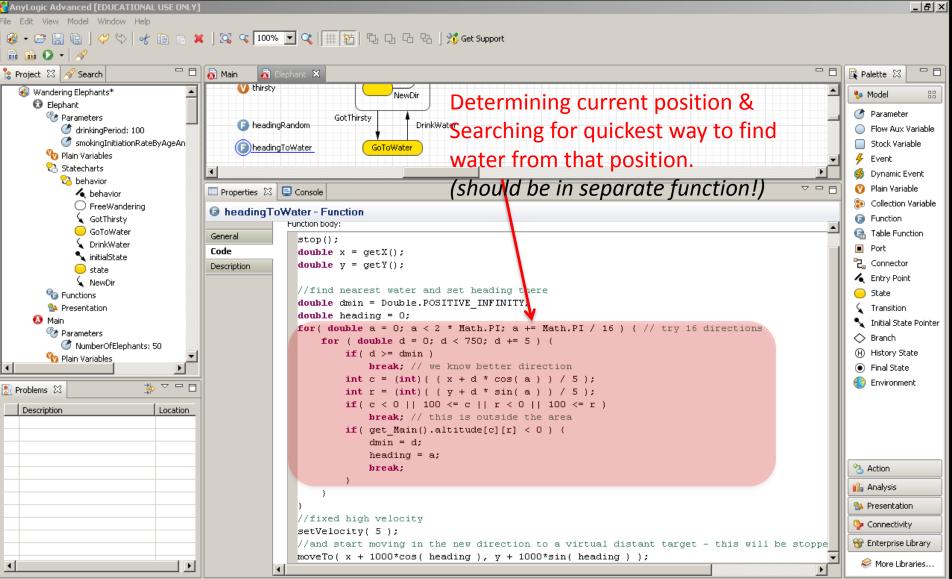
- "For" statements "iterate", repeatedly executing some inner statement many times
- Several variants are available

for (int i = 0; i < 100; i++)

statement

for (int i : *collection*) *statement* 

### **Heading Towards Resource**



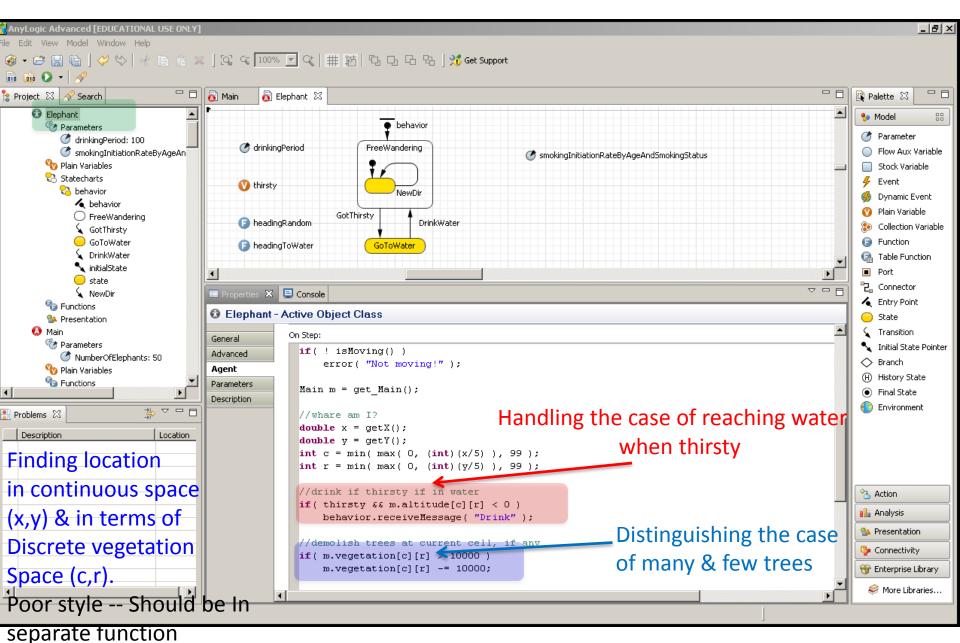
Selection

# If Statements

 With an if statement, one tests a condition ("predicate"), and – based on the result – either executes one statement or another (possibly empty) statement

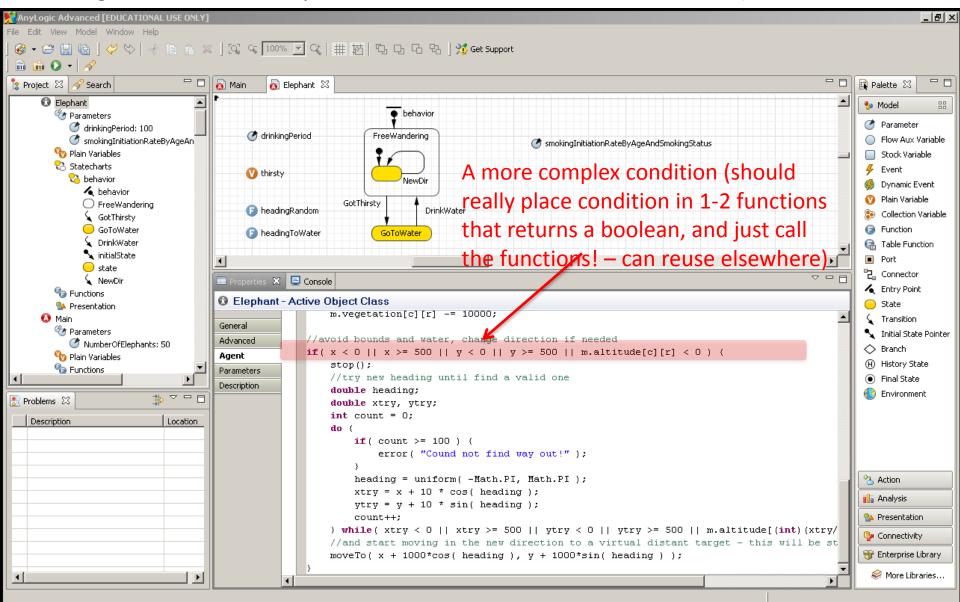
if (condition) if (condition) true-statement or true-statement else false-statement "falls through" to later code if condition is false

# Handling of Movement Logic

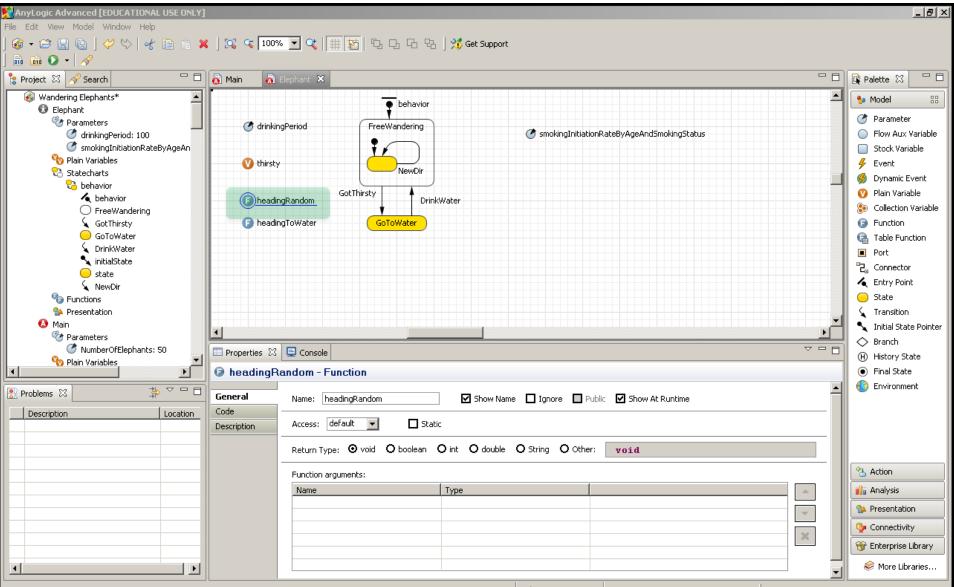


#### Rerouting Around Barriers (Boundaries & Water) Poor Style – entire logic, conditions (checks on boundaries, whether water) & rerouting

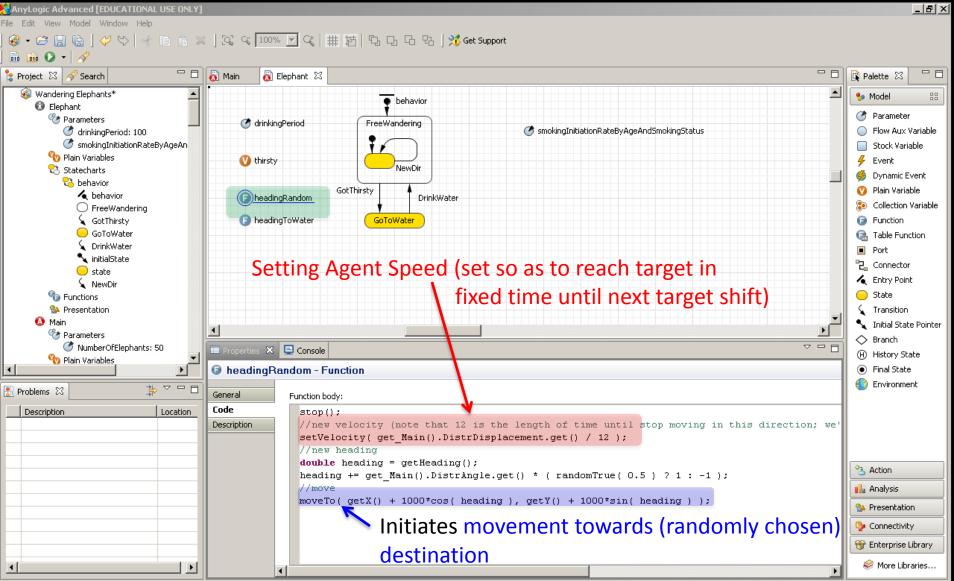
Logic should all be in separate functions from this & from each other). Remove constants



#### **New Direction Change Function Info**



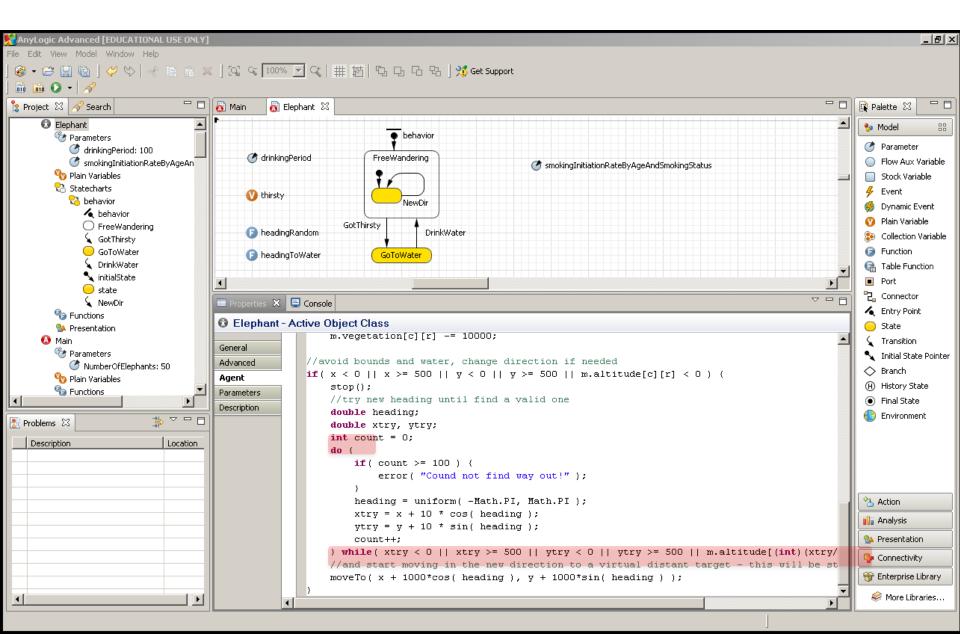
#### New Direction Change: Function "Body"



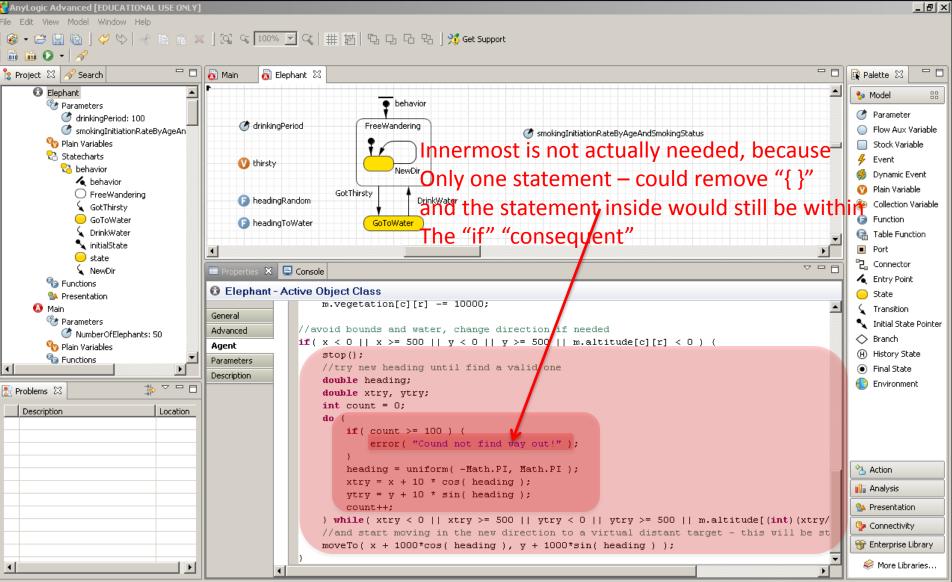
# "While"/"Do while" loop

- Executes a statement as long as some condition is true
- The classic "While" loop has the test at the beginning
- The "do while" has the test at the end of the loop

## While loops



#### Compound Statements (Delineated by "{ }")



#### **Expression Statements**

