CSE 390B, Winter 2022

Building Academic Success Through Bottom-Up Computing

Stress & Wellness, Project 7 Overview

Stress and Wellness Discussion, Project 7 Overview, Project 7 Tips and Tricks

Lecture Outline

- **Stress and Wellness Discussion**
- Project 7 Overview
- Project 7 Tips and Tricks

Stress & Wellness Discussion

- Reading was to listen/read Episode 1 of the Feminist Survival Project podcast
- Topic was the stress response cycle and dealing with stress vs. dealing with stressors
- Lots of information to process despite being a relatively short episode!
  - Discussion with others can be a great way to debrief something that introduces a potentially new way of thinking about things

Stress & Wellness Discussion Prompt 1

- We can often trick ourselves into thinking that stress is something that can be willed away by our minds, a notion challenged by Emily and Amelia's theme “Wellness is not a state of mind nor is it a state of being. It is a state of action”
  - How much time do you set aside for taking action towards wellness?
  - How much have you reflected on what actions do/don't contribute to your wellness?
  - What might be an example of treating wellness as a state of mind/being vs. treating wellness as a state of action?

Stress & Wellness Discussion Prompt 2

- One focus of the podcast was the importance in distinguishing between addressing our stress response and addressing the stressors in our life
  - Reflecting on your own life, identify some of the common stressors that come up for you both externally (e.g., traffic) and internally (e.g., self-criticism)
  - How do you typically respond to stressors – physically, emotionally and cognitively?
  - Where do you feel you could focus on addressing your stress response and situations where you could focus on addressing your stressors? Why might those situations be more suitable to focusing on one or the other?

Lecture Outline

- Stress and Wellness Discussion
- **Project 7 Overview**
- Project 7 Tips and Tricks

Project 7 Overview

- Buggy Compiler
  - Code Generation Implementation/Debugging

- Project 7 will be due Tuesday (3/8)
  - Note the shift in due date!
  - We expected to start project 7 until after midterm corrections are turned in this Thursday

Project 7: Buggy Compiler

- You will be given starter code for a compiler that reads a micro version of Jack and spits out Hack

- The Scanner & Parser are working
  - Task A: Read through to understand what's going on. There are comments to help you out!
- The Code Generation is buggy and half-finished
  - Task B: Find the bugs by practicing deliberate debugging strategies. Remember you can step through generated Hack code using the CPUEmulator tool
  - Task C: Finish the compiler! These slides will be a helpful reference

Project 7: Micro Jack

- Stripped-down version of Jack language
  - More manageable but enough features to be interesting!
- Has:
  - Types: Int and Int[], Variable Assignment, If, While, +, -, ==, !=
- Doesn't Have:
  - Functions/Function Calls, Classes/Objects, Strings, For, Array Bounds Checking, etc.

Any number of variable declarations

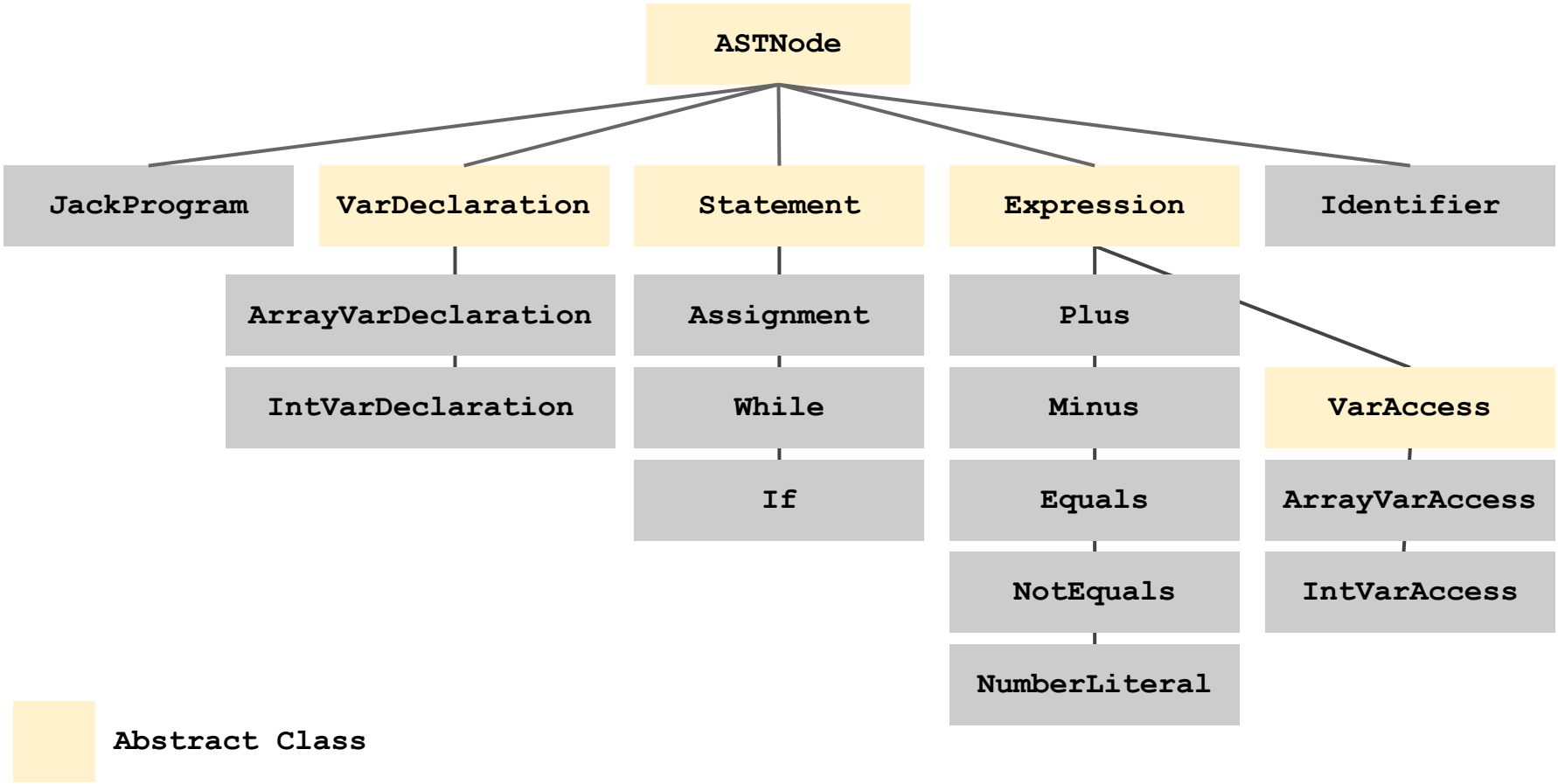
Basic.jack

```
var int a, b[1], c;
var int d[10], e;

let a = 1;
let b[0] = 1;
let n = 9;
while (n != 0) {
  let d[n] = a;
  let n = n - 1;
}
let screen[100] = d[0];
```

Then any number of statements

Project 7: The AST Nodes

- You are provided with all AST Node classes needed
  - All of your code will be implemented within these classes



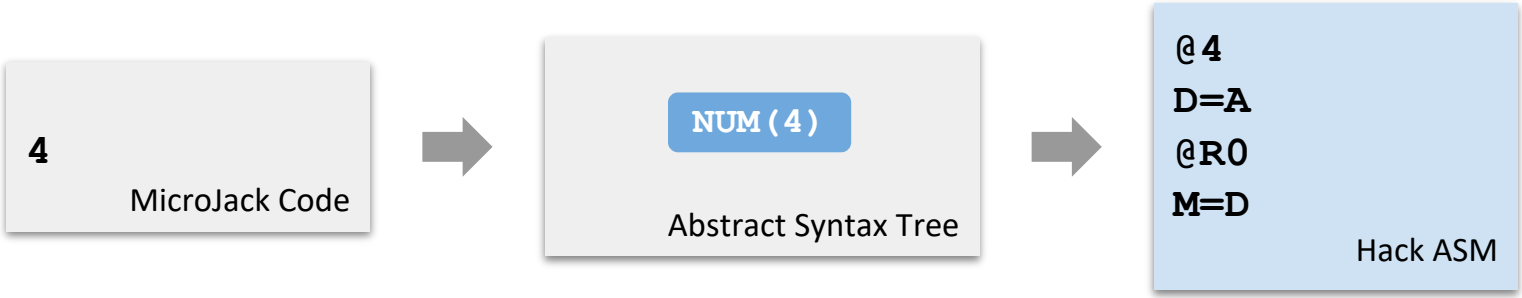
Project 7: Generating Code

- Each AST Node has a printASM method that should print out Hack instructions to System.out (and recursively call printASM on children)
  - You're provided with instr("@R0") and label("LOOP") convenience functions
  - Each can take a comment as a second argument -- HIGHLY recommended!

```
public class If extends Statement {
  public Expression condition;
  public List<Statement> statements;

  ...

@Override
public void printASM(symbolTable) {
  condition.printASM(symbolTable);
  instr("@R0", "Get cond result");
  instr("D=M");

  ...

}
```

Example: Number Literal (Step 1)

- Called a "literal" because it's a literal value embedded in the MicroJack code
  - Generated Hack ASM should simply put that value in R0



Example: Number Literal (Step 1)

```
public class NumberLiteral extends Expression {
    public int value;

    public NumberLiteral(String value) {
        this.value = Integer.parseInt(value);
    }

    @Override
    public void printASM() {
        comment("Start Number Literal");
        instr( ? );
        instr("D=A");
        instr("@R0");
        instr("M=D");
        comment("End Number Literal");
    }

    @Override
    public String toString() {
        return Integer.toString(value);
    }
}
```

4

```
// Start Number Literal
@4
D=A
@R0
M=D
// End Number Literal
```

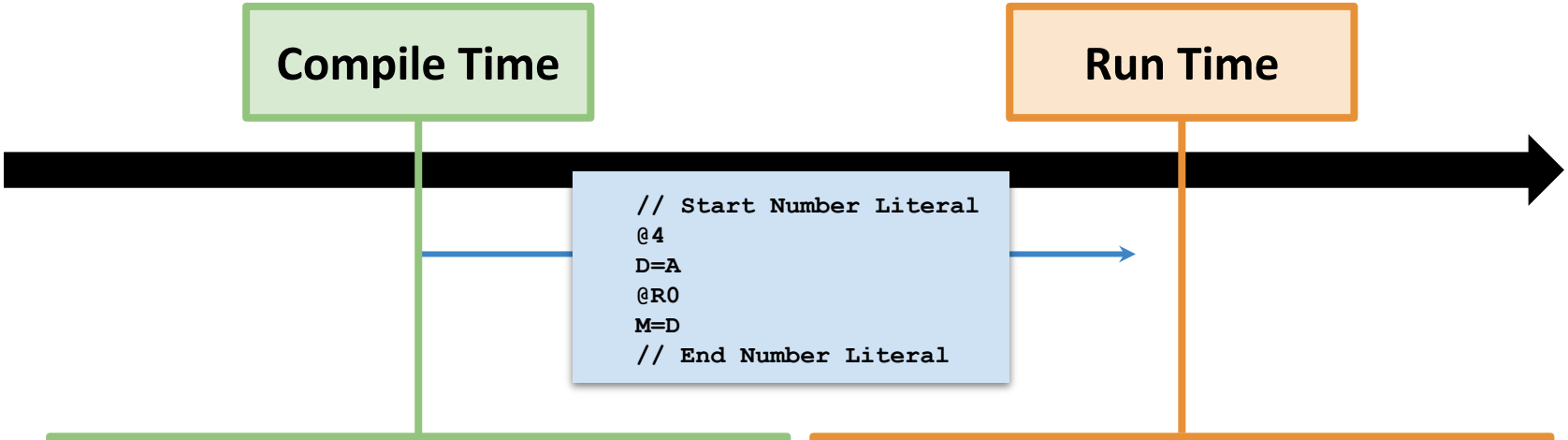
Example: Number Literal (Step 1)

```
public class NumberLiteral extends Expression {
    public int value;

    public NumberLiteral(String value) {
        this.value = Integer.parseInt(value);
    }

    @Override
    public void printASM() {
        comment("Start Number Literal");
        instr("@" + toString());
                instr("D=A");
        instr("@R0");
        instr("M=D");
        comment("End Number Literal");
    }

    @Override
    public String toString() {
        return Integer.toString(value);
    }
}
```

4

```
// Start Number Literal
@4
D=A
@R0
M=D
// End Number Literal
```

Example: Number Literal (Step 1)



- Java (the compiler) is running
- 4 is stored as value field inside an NumberLiteral ASTNode
- When executed, **prints code that stores it another way**!

```
public class NumberLiteral extends Expression {
    public int value;

    public NumberLiteral(String value) {
        this.value = Integer.parseInt(value);
    }

    @Override
    public void printASM() {
        comment("Start Number Literal");
        instr("@" + toString());
        instr("D=A");
        instr("@R0");
        instr("M=D");
        comment("End Number Literal");
    }

    @Override
    public String toString() {
        return Integer.toString(value);
    }
}
```

- Hack ASM (the output) is running
- 4 is stored as constant inside an assembly instruction
- When executed, loads 4 from instruction into A register

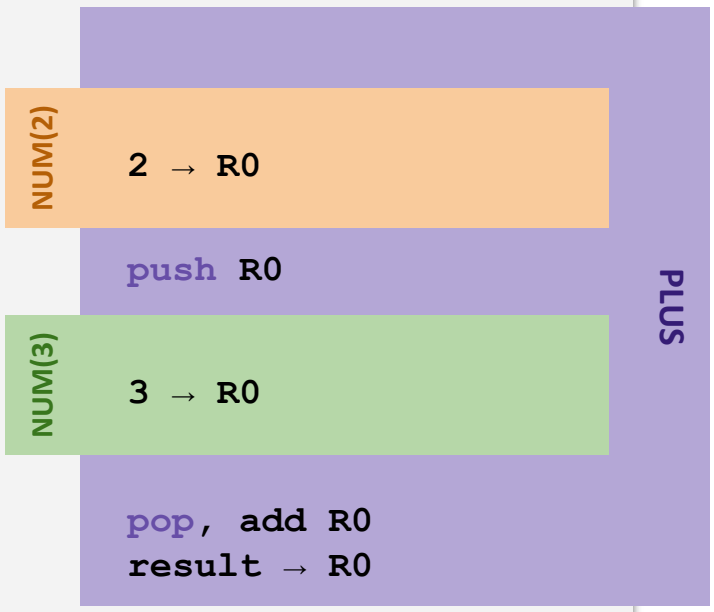
Example: Plus (Step 2)

```
public class Plus extends Expression {
    public Expression left;
    public Expression right;

    @Override
    public void printASM() {
        comment("Start Plus");

        left.printASM();

        instr("@R0");
        instr("D=M");

        right.printASM();

        push();

        instr("@R1");
        instr("A=M");

        instr("D=D+A", "perform the addition");

        ...
```

Example: Plus (Step 2)

1 Structural Bug: Map to abstract diagram for Plus:

```
public class Plus extends Expression {
    public Expression left;
    public Expression right;

    @Override
    public void printASM() {
        comment("Start Plus");

        left.printASM();

        instr("@R0");
        instr("D=M");

        right.printASM();

        push();

        instr("@R1");
        instr("A=M");

        instr("D=D+A", "perform the addition");

        ...
```



1 Detail Bug: Step through generated code, Check state at each step

Project 7 Buggy Compiler Overview

0	Read starter code	
1	**Implement** NumberLiteral.java	~4 Lines
2	**Debug** Plus.java	2 Bugs
3	**Implement** Minus.java	~13 Lines (similar to Plus)
4	**Implement** NotEquals.java	~21 Lines (similar to Equals)
5	**Implement** ArrayVarAccess.java	~3 Lines
6	**Debug** If.java	2 Bugs
7	**Implement** While.java	~14 Lines

Lecture Outline

- Stress and Wellness Discussion
- Project 7 Overview
- **Project 7 Gotchas and Tips**

Project 7: MicroJack Language Gotchas

- Can't write a negative integer literal
  - Instead, use subtraction from zero: 0 - 1
- All variable declarations must come before all regular statements
  - (Why? Simplifies concept of a "defined" variable)
- No defined operator precedence
  - If order matters for an operation, use parentheses

Project 7: MicroJack Language Gotchas

- Arrays are very simple
  - arr[index] is really just calculating an address: take address of arr variable and add index to it as an offset
  - No array bounds checking -- just lets you run off the end

- "Booleans" are just 0 (false) and non-zero (true)

Project 7: Debugging Tips

- Try walking through the general printASM code to understand why each line is there
  - Add comments to the assembly as you go! Much easier to understand resulting file
- Find the smallest example you can
  - Provided tests get progressively more complex, but you may want to write your own tiny test case to isolate
  - ASM gets long fast—we've added comments so you can isolate to the section you're working on
- "Play Computer": as you step through the code, write down the state you expect after each instruction, then advance and see if the CPUEmulator agrees

Project 7 Tools Demo

Project 7 Tools Practice

- Practice using the project 7 tools! Try doing the following:
  - Run git pull to grab the project 7 starter code
  - Navigate to the src/ directory: **`cd src/`**
  - Compile the Java source code of the compiler by running: **`javac $(find . -name "*.java")`**
  - Use your compiler to compile the Jack file for the OnlyVars program: **`java compiler/Compiler compile ../test/OnlyVars.jack`**
  - Load/run OnlyVars.tst in the CPUEmulator

- The above steps were taken from the "How to Run Tests" portion of the spec
  - Can refer to this when needed as you work through the project

Post Lecture-16 Reminders

- What's in store for Week 9?
  - Design Processes
  - Networking
  - Final Project Overview

- Reminders
  - Midterm Corrections due tonight (2/24) at 11:59pm PST (No late days)
  - Professor Meeting Report Due next Thursday, May 27th
  - Project 7 Released, due Tuesday, March 8th