

CSC 447 - Concepts of Programming Languages

Course Overview

Instructor: Stefan Mitsch



Autonomous and Intelligent Systems

-  Want a computer to perform a task (transfer money, order goods)
-  Interact with the physical world (drive a car, fly an airplane)
- Requires clear **instructions**: a programming language
 - move left, right, up, down
 - ⋮ do one step and then another step
 - ↺ repeat steps



PL Syntax, Semantics, Pragmatics

- AB ✓ Syntax:** structure of a programming language and rules for writing valid instructions (grammar)
- ⚙️ Semantics:** meaning of valid instructions
- </> Pragmatics:** how to use the language, best practices for writing code



PL Syntax and Semantics

Different Syntax, Same Semantics

Java

```
1 final int x = y>0 ? 1 : -1;
```

Scala

```
1 val x = if y>0 then 1 else -1
```

Same Syntax, Different Semantics

Java

```
1 for (var i = 0; i < 10; i++) {  
2     var x = i;  
3 }  
4 // Here i and x are both undefined
```

JavaScript

```
1 for (var i = 0; i < 10; i++) {  
2     var x = i;  
3 }  
4 // Here i is 10 and x is 9
```



PL Concepts by Example

Java

```
1 class Intro {
2     private static int isEven(int n) {
3         return n % 2 == 0 ? 1 : 0;
4     }
5
6     public static int countEven(List<Integer> xs) {
7         int result = 0;
8         if (!xs.isEmpty()) {
9             result =
10                 isEven(xs.getFirst()) +
11                 countEven(xs.subList(1, xs.size()));
12         }
13         return result;
14     }
15
16     public static void main(String[] args) {
17         List<Integer> numbers =
18             Arrays.stream(args).
19                 map(str -> Integer.parseInt(str)).
20                 toList();
21         System.out.println("You entered " + countEven(numbers) +
22             " even numbers");
23     }
24 }
```

Concepts

- Statements and expressions
- Strict and nonstrict evaluation
- Collection processing
- Recursion
- L-Values
- Argument passing
- Parametric polymorphism



PL Concepts by Example

C

```
1 int isEven(int n) { return n % 2 == 0 ? 1 : 0; }
2
3 int countEven(int argc, const int* const xs) {
4     int result = 0;
5     if (argc > 0) {
6         result =
7             isEven(xs[0]) +
8             countEven(argc-1, &xs[1]);
9     }
10    return result;
11 }
12
13 int main(int argc, char **args) {
14     int *vals = malloc((argc-1)*sizeof(int));
15     for (int i = 1; i < argc; ++i) {
16         char* end;
17         vals[i-1] = strtol(args[i], &end, 10);
18         // forgot to check whether all characters
19         // are consumed (i.e., end == '\0')
20     }
21     int numEven = countEven(argc-1, vals);
22     printf("You entered %d even numbers\r\n", numEven);
23     // forgot to free the dynamically allocated memory
24     return 0;
25 }
```

More Concepts

- Safety and undefined behavior
- Static and dynamic memory allocation
- Scope and lifetime



PL Concepts by Example

Scala

```
1 enum List:
2   case Empty
3   case Cons (head:Int, tail:List)
4
5   def length (result:Int = 0): Int = this match
6     case List.Empty      => result
7     case List.Cons(a,as) => length(as, result+1)
8
9 object List:
10   def create(elems: Int*) =
11     elems.foldRight(List.Empty)((e, l) => List.Cons(e, l))
```

More Concepts

- Algebraic datatypes
- Functional programming
- Pattern matching
- Tail recursion
- Type inference
- Currying



PL Pragmatics and Large Language Models



- 1 Implement a Java program that counts
- 2 how many even even numbers are in
- 3 the command line inputs



```
1 public static void main(String[] args) {
2     int evenCount = 0;
3
4     for (String arg : args) {
5         try {
6             int number = Integer.parseInt(arg);
7             if (number % 2 == 0) {
8                 evenCount++;
9             }
10        } catch (NumberFormatException e) {
11            System.out.println(
12                arg + " is not a valid integer.");
13        }
14    }
15    System.out.println(
16        "Count of even numbers: " + evenCount);
17 }
```



- 1 Implement a Scala program that counts
- 2 how many even even numbers are in
- 3 the command line inputs



```
1 def main(args: Array[String]): Unit = {
2     val evenCount = args.foldLeft(0) {
3         (count, arg) =>
4             try {
5                 val number = arg.toInt
6                 if number % 2 == 0 then count + 1
7                 else count
8             } catch {
9                 case e: NumberFormatException =>
10                    println(
11                        s"$arg is not a valid integer.")
12                    count
13            }
14    }
15    println(
16        s"Count of even numbers: $evenCount")
17 }
```



What are you doing to convince yourself that the code is correct?



Why is it Important to Understand PL Well?

- Write high-quality code
- Assess the correctness of (auto-generated) code
- Assess the quality of (auto-generated) code
- Assess the performance of (auto-generated) code
- Generalize (auto-generated) code
- Instruct tools to refactor and improve code
- Read, maintain, and extend existing code



Programming Languages

- **1970:** assembly, FORTRAN, COBOL, Lisp
- **1980:** C, Pascal, BASIC, ML, Smalltalk
- **1990:** C++, Perl, Objective C, Erlang
- **2000:** Java, JavaScript, Python, Ruby, Lua
- **2005:** C#
- **2010:** Scala, F#, Clojure, Go
- **2015:** Rust, Swift, Kotlin, Elm, Elixir, TypeScript, PureScript
- **2020:** ReasonML, Crystal, Pony, Zig
- Many more!



Programming Languages

In just a couple of minutes, we explored different implementations of the same feature in different languages!

- Programming languages keep popping up
- You will have to keep learning new languages
- And keep up with changes to current languages
 - Java 8 and C++ 11 added Lambda expressions (nested, anonymous functions)
- 💡 Lots of common concepts!



Learning Objectives

Learn PLs more easily by recognizing concepts

- "it has conditionals, recursion, loops"
- "it has closures"
- "it has list comprehensions"
- "it has dynamic dispatch"
- Deeper understanding of PL concepts / paradigms
- Impact of PL on program development, modularity, correctness, runtime efficiency, etc.
- Alternative way of thinking to double-check generated code



Non-Goals

- ✗ Learn a PL similar to one you already know
- ✓ Instead, learn concepts to facilitate quick assimilation of new PLs as they appear
- ✗ Learn an IDE
- ✓ Instead, become familiar with a range of tools and build systems



Course Overview

! This is a challenging course

- deepen your understanding about programming while also learning a new language
- study guides and extra credit will help you succeed!
- take advantage of office hours!

? What are different ways of expressing computations?

- **Programming paradigms and styles:**
 - functional vs object-oriented
 - mutability vs immutability
 - iteration vs recursion
 - pattern matching vs visitor pattern
- **Concepts:** lexical and dynamic scope; stack layout; inheritance and dynamic dispatch; nested structures (functions or objects); dynamic vs static type checking; subtyping; parametric polymorphism



Course Approach

- Hands on: write many programs and experiments
- Use AI coding support responsibly: write contracts and tests, develop language extensions to help analyze auto-generated code
- **Scala** as main language:
 - carefully designed *multi-paradigm* language
 - textbook explains PL concepts in context
- Also bits of: C, C++, C#, Java, JavaScript, ... chosen as exemplars of concepts



Discussion Forum

- On D2L
- Use appropriate language

Asking Questions

- For non-personal messages use D2L
 - "I think there is a mistake with grading of Question 3 of Homework 2." - **direct message/email to instructor**
 - "I cannot run program `foo`. I have tried running it from the command line on OS X. See below for a transcript of what I typed and the error message I received. Could you please help?" - **ask in D2L discussion forum**



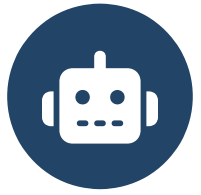
Asking Questions

- Include name, student number, and class in messages directly to the instructor
- Include enough context to answer your question:
 - "program `foo` doesn't work." - *starts discussion*
 - "program `foo` fails with output pasted below" - *good*
 - "program `foo` fails with input and output pasted below" - *better*
 - "program `foo` fails with input and output pasted below; using OS X; I have run extra commands to show current working directory, the version of `foo` in use, and other relevant information" - *best*

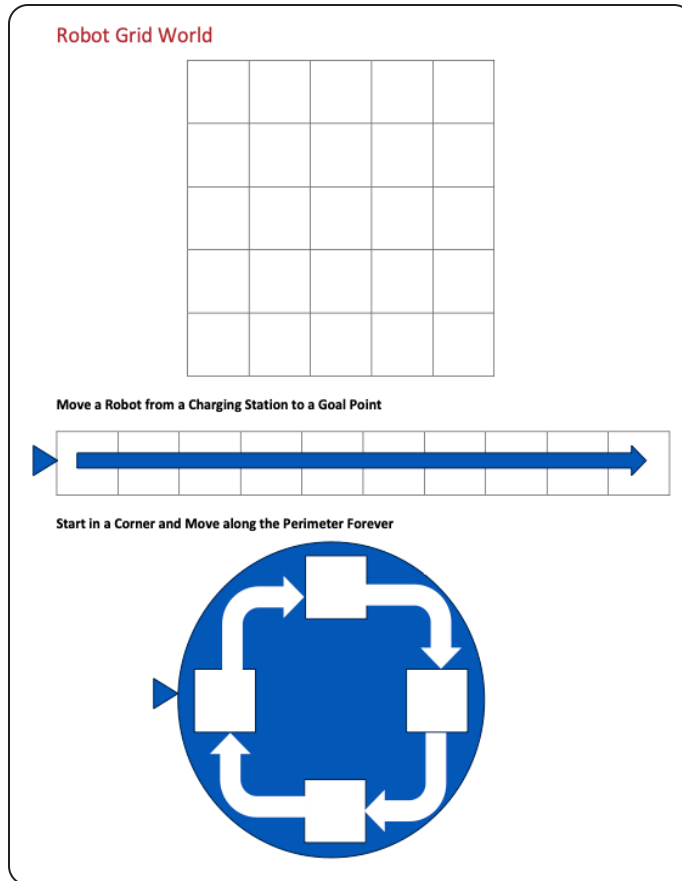


Course Syllabus

- Review the [Syllabus](#) linked from the [course homepage](#)
- Programming in Scala, First Edition is available for free [online](#)
- **Get Programming in Scala 5th edition**
- Earlier editions use a different version of Scala than class; you can use them but may need to look up new syntax

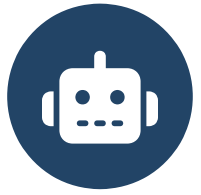


Program a Robot in a Grid World

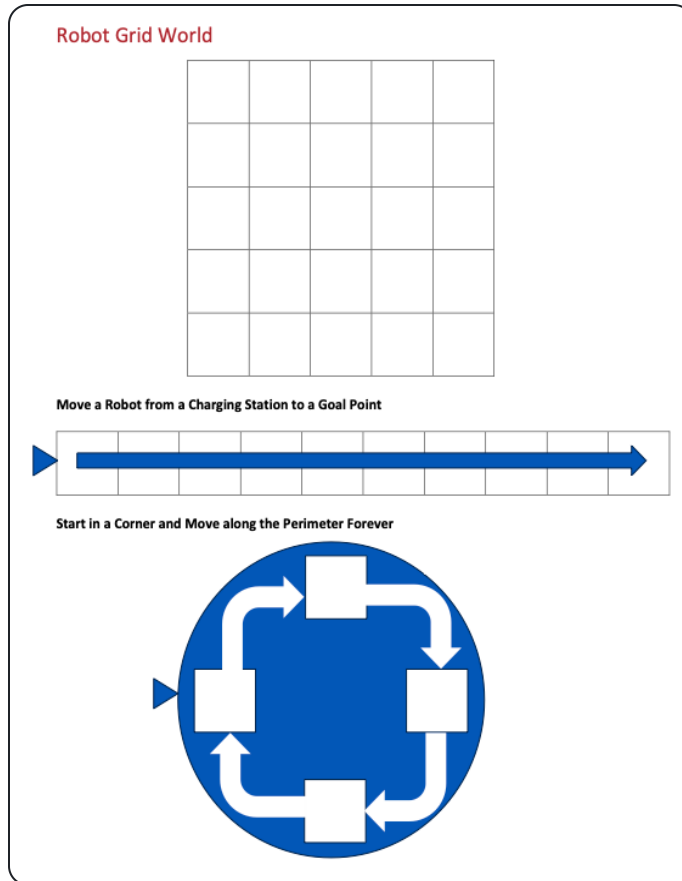


Move Robot according to Instructions

- Mark the robot's initial position on the grid
- Mark a goal on the grid
- Create instructions to express how the robot moves
- Move the robot according to your instructions: fill in the *sequential* blue arrow instruction list



Program a Robot in a Grid World



Move Robot along the Perimeter

- Place the robot in a corner
- Use your instructions to express your program
 - The loop  has 4 slots for instructions that repeat forever
 - Place your elementary instructions into the slots
- Move the robot according to your instructions



Computers that Program



Teach a programming language to an LLM (ChatGPT)



Syntax (grammar) of the language

```
1 step ::= down n | up n | left n | right n | seq step then step (then step)* | repeat(step)
```



Semantics of the language



Ask LLM to write a program

```
1 Use this language to move the robot in a 5 by 5 grid along the perimeter, starting in the top-left corner.
```



Our programming language has



Parser to turn text into instructions



Interpreter to execute the instructions