Team

- Caitlyn Chen
  Language Guru
- Tiffeny Chen
  System Architect
- Jang Hun Choi
  System Architect
- Mara Dimofte
  Manager
- Christi Kim
  Tester

TA: Xijao Li
Motivation

- Card games come in many different forms -- standard 52-card deck to UNO, Apples, and Pokemon relying on unique decks.

- Existing card languages fail to generalize the full breadth of card games

- Allows users to easily code the gameplay and the functionality of a turn-based card game
Language Tutorial - Basic

```plaintext
1 int a
2 main:
3 {
4  float b
5  b = 5.0
6  if 3<5:
7     {
8         for (a = 0 ; a < 5; a = a + 1):
9           {
10              b = b + 1.0
11              do PRINT(b)
12          }
13     }
14 }
15 }
```

*strongly typed language
- types: int, float, bool, string, void, series, player, card

*newline instead of semi colon
Language Tutorial - Advanced

```plaintext
when do series<Card> CREATEPILE():

CREATEPILE

action (func) CREATEPILE

created card object

created card object

called action

accessed 0th element of series cards

returned series literal

return [card1]

card1 = Card("R5", true)

series<Card> cards

card = cards[0]

do PRINT(card.type)

do PRINT(card.faceup)

R5, true
(respectively)
```
## Features Summary

<table>
<thead>
<tr>
<th>Feature</th>
<th>Associated Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Flow</td>
<td>if, else, while, for</td>
</tr>
<tr>
<td>Built in Objects</td>
<td>Player, Card</td>
</tr>
<tr>
<td>Series Literals</td>
<td>push, pop, get, size</td>
</tr>
<tr>
<td>Action Declarations</td>
<td>accessing params, return</td>
</tr>
</tbody>
</table>
Built-in Objects

Player

```java
1 Player player
2 main:
3 {
4     player = Player("bob", 0)
5     do PRINT(player.name)
6     do PRINT(player.score)
7 }
```

Card

```java
1 Card card
2 main:
3 {
4     card = Card("R5", true, 5)
5     do PRINT(card.type)
6     do PRINT(card.faceup)
7     do PRINT(card.value)
8 }
```
```java
series<int> a
series<string> b
int i
main:
{
    a = [2]
    a.push(1)
do PRINT(a[1])
    a.push(3)
do PRINT(a[2])
    a.pop()
    for (i = 0; i < a.size(); i = i + 1):
    {
do PRINT(a[i])
    }
    b = ["stmt1", "stmt2"]
b.push("stmt3")
b.push("stmt4")
do PRINT(b[3])
```
Action Declarations

Program

```
when do string A(string x):
{
    string z
    z = x
    return z
}
main:
{
    string w
    w = do A("Ahoy Matey, to AHOD")
    do PRINT(w)
}
```

Output

```
1  Ahoy Matey, to AHOD
```
Compiler Architecture

AHOD Program → Tokens → AST → SAST → Code Generation → LLVM IR

Scanner → Parser → Semantic Check → C library → AHOD executable
Demo
Next Steps

- Create more built-in-methods:
  - string-to-int, int-to-string, user input, etc.
- Make code more concise and flexible
  - use whitespace as delimiters (indentation, newline, etc.)
- Allow users to define classes and class attributes
- More list implementations
  - slicing, remove and find certain element
- Type inferencing
- Optimizations of register allocations
- Garbage collection
Thank you for all the guidance this semester!