WiBitNet™

Programming Core Concepts

Syntax

- A set of rules that combines symbols and formatting to define the legal structure of a computer program
- Think of it as Grammar
- Most languages have their own syntax

Variables

- Used to store a value in memory that can be used in the program
- Variable Scope (Local & Global)
  - A variable is only available to a section of a program based on
    - Where it's declared
    - Its variable type

Expressions

- Perform comparisons and operations on variable types
  - Examples
    - X = 1 + 2
      - Set variable ‘X’ equal to the addition expression *‘1 + 2’*
      - 1 + 2 = 3
    - Y = X – 1
      - Set variable ‘Y’ equal to the subtraction expression *‘X - 1’*
        - ‘X’ must already exist for this expression to be valid
      - Y = 3 – 1 = 2
    - ST = “HELLO. ” + “HOW ARE” + “ ” + “YOU?”
      - Set variable ‘ST’ equal to the concatenation expression *“‘HELLO. ” + “HOW ARE” + “ ” + “YOU?”’*
      - ST = “HELLO. HOW ARE YOU?”
    - IS_TRUE = (x == 2)
      - Set variable ‘IS_TRUE’ equal to the Boolean expression (x == 2)
      - IS_TRUE = (3 == 2) = FALSE

Arrays / Lists

- Classical
  - A list of values of the same data type consisting of a predefined length that is allocated before the program executes
- Modern (Aka LinkedList / ArrayList / Dynamic Array)
  - A list of objects of the same or different types with a length that is allocated as needed during runtime

Array

Visual representation of an array

0	1	2	3	4

Array

Visual representation of an array

0	1	2	3	4
VALUE 1	VALUE 2	VALUE 3	VALUE 4	VALUE 5

Visual representation of a list

Memory	1	2	3	4
1				
2				
3				
4				
5				
6				

Visual representation of a list

Memory	1	2	3	4
1				
2				
3				
4				
5				
6				

Visual representation of a list

Memory	1	2	3	4
1	Pointer			
2				
3				
4				
5				
6				

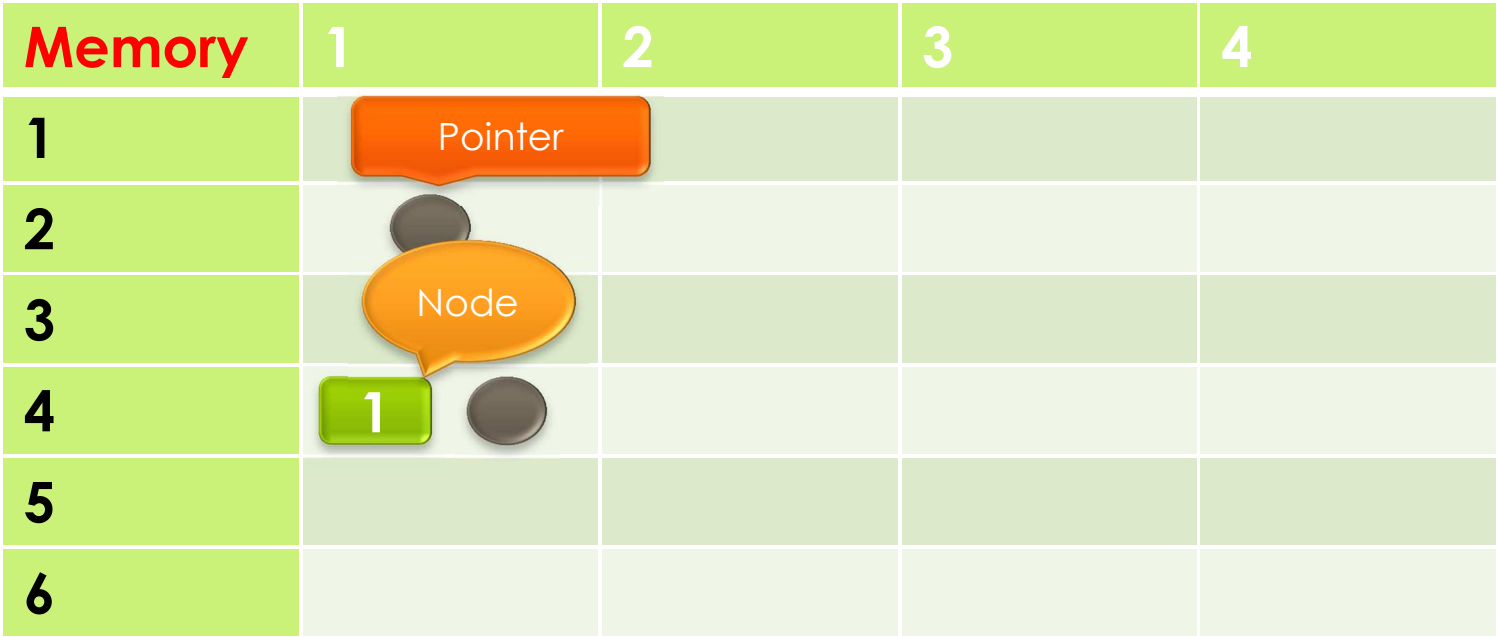
Visual representation of a list

Memory	1	2	3	4
1	Pointer			
2				
3				
4	1			
5				
6				

Visual representation of a list

Memory	1	2	3	4
1	Pointer			
2				
3	Node			
4	1			
5				
6				

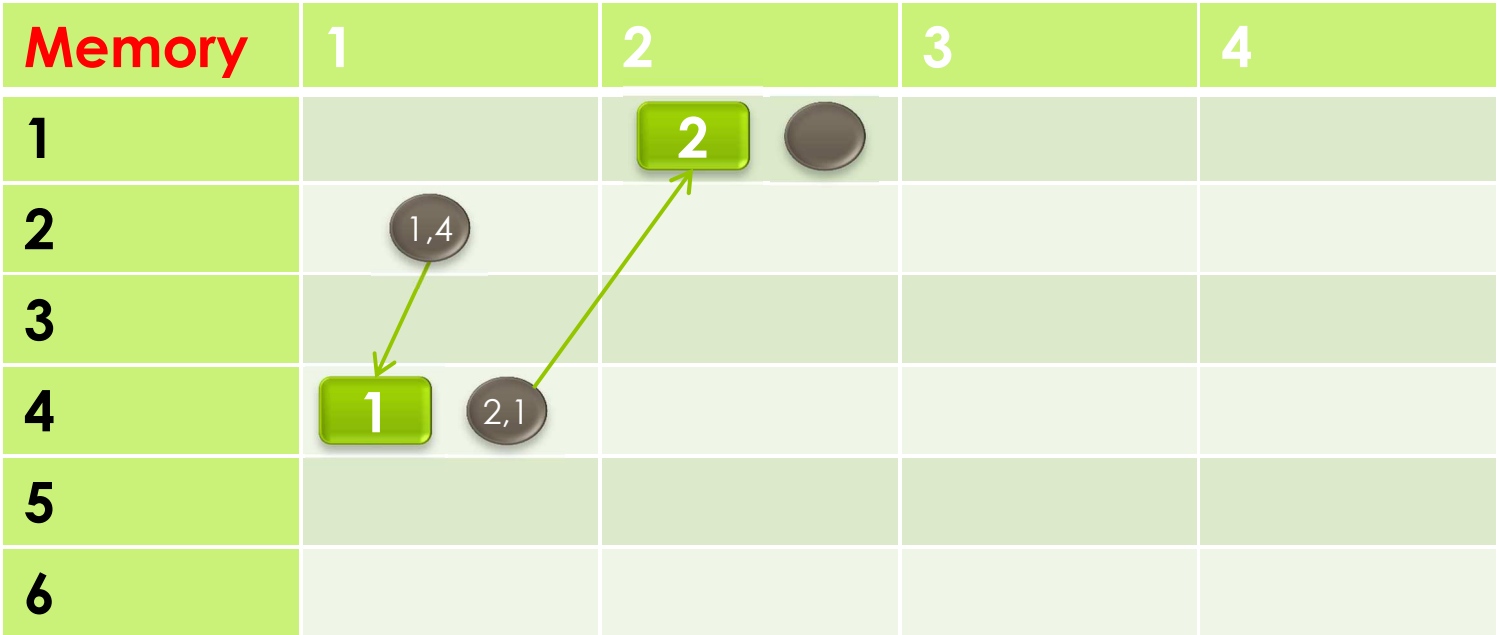
Visual representation of a list



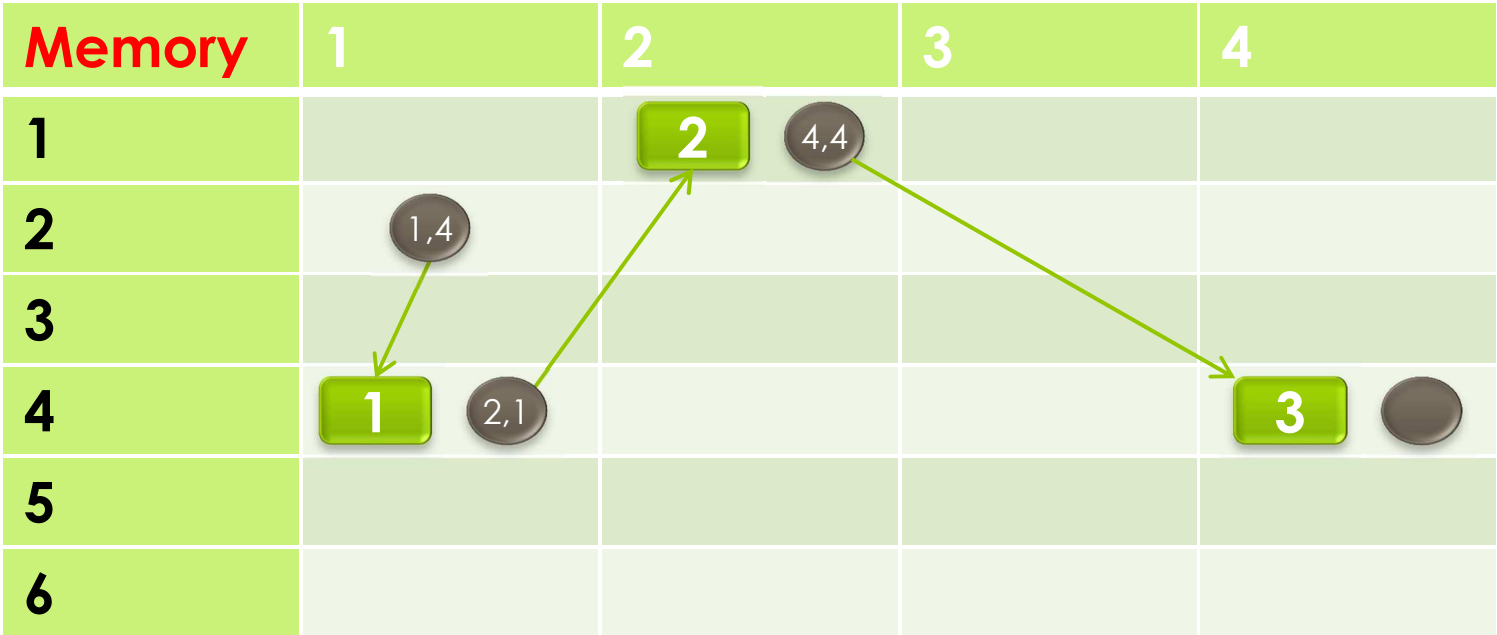
Visual representation of a list

Memory	1	2	3	4
1				
2	1,4			
3				
4	1			
5				
6				

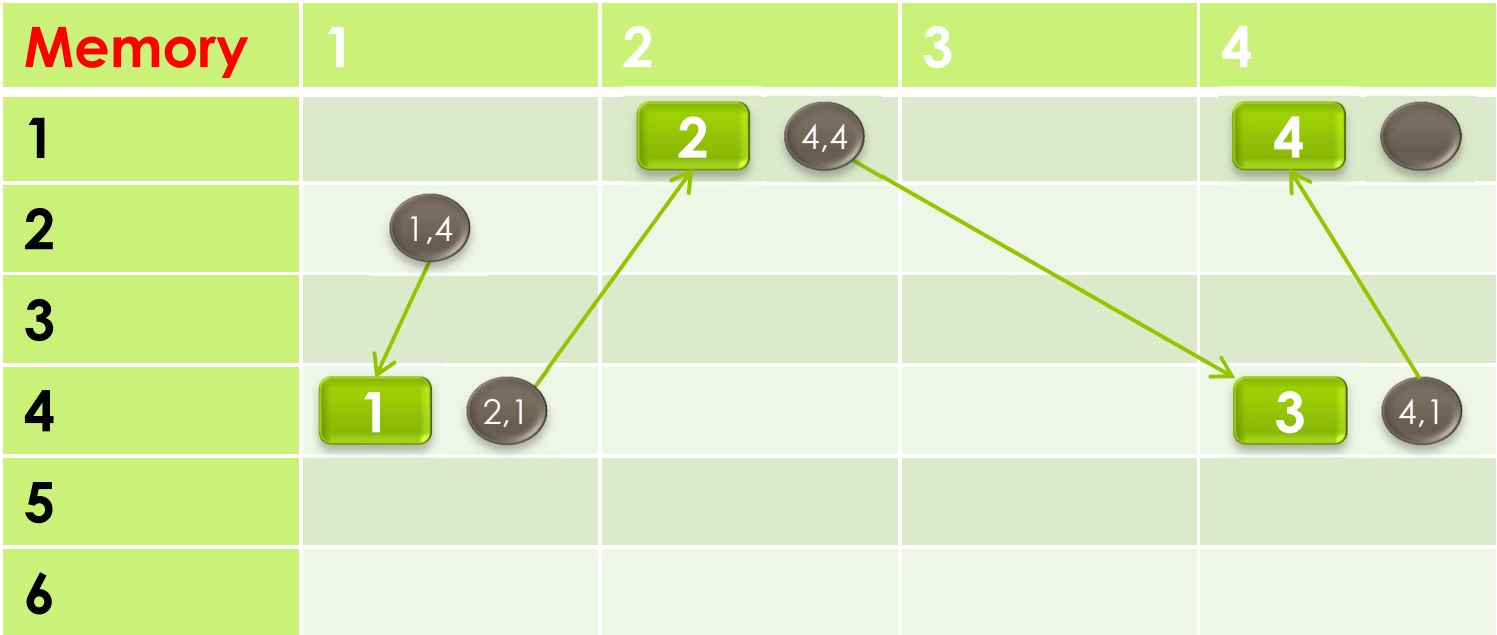
Visual representation of a list



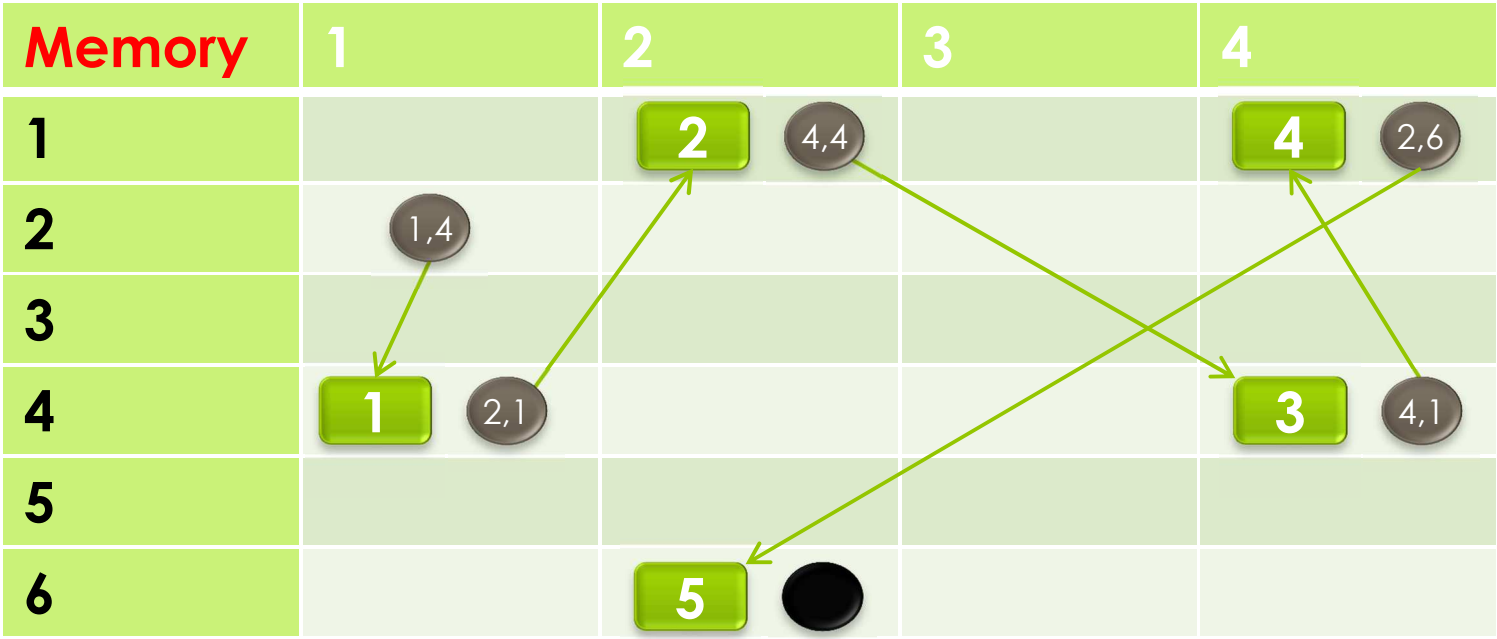
Visual representation of a list



Visual representation of a list



Visual representation of a list



Multidimensional Arrays

- Assortment of objects that relate to one another by indexing
  - Most common type
    - 2D Array

2-Dimensional Arrays

Visual representation of a 2 dimensional array

	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	4	6
3	0	3	6	9
4	0	4	8	12

Pointers

- A variable that stores a memory address location rather than a value

Variable Type	Variable Value
Pointer 0	0, 0
Pointer 1	1, 0
Pointer 2	1, 0
Pointer 3	2, 0

Memory

	0
0	x64
1	xAF
2	xFF

Input / Output

- Input
  - Any method used to give a program information
    - Command line arguments
    - Input file
    - Sockets
- Output
  - Any method used to get information from a program
    - Screen / Monitor
    - Printer
    - Output file
    - Sockets

Dependencies / Libraries

- Any Code/Library/Executable that is necessary for a program to execute or compile
- Example
  - In order for you to know how to walk, you had to learn to stand. Thus, you cannot walk unless you know how to stand.
    - To walk, you must include your knowledge about standing
  - In C++, I must declare the Input / Output Stream dependency in order to write text to the screen

Functions / Methods

- Function
  - A portion of computer code within a larger program that performs a specific task
  - AKA
    - Subroutine / Sub
    - Procedure / Proc
- Method
  - The same thing as a function, but it is exclusively related to a specific class or object (Object Oriented Programming)

Decision Statements

- … a programming structure that is used to make logical decisions based on Boolean expressions during the runtime flow of a computer program
  - Types
    - IF / THEN / ELSE
    - CASE

Decision Statements

```
If (<CONDITION IS TRUE>) THEN
   DO SOMETHING
```

```
If (<CONDITION IS TRUE>) THEN
   DO SOMETHING
Else
   Do SOMETHING ELSE
```

```
If (<CONDITION IS TRUE>) THEN
   DO SOMETHING
Else If (<CONDITION IS TRUE>) THEN
   Do SOMETHING
Else
   Do SOMETHING ELSE
```

Loops

- A programming structure that is used to process multiple iterations of computer code (the same code multiple times)
  - Types
    - While
    - Do / While
    - For
    - For Each

Loops: While

- A loop structure that executes code while a particular expression is true

```
X = 1
While ( X < 10 )
   PRINT X
   X = X + 1
```

Loops: Do / While

- A loop structure that executes code while a particular expression is true, and executed at least once

```
X = 1
Do
   PRINT X
   X = X + 1
While ( X < 10 )
```

Loops: For

- A loop structure that executes code from a designated starting and ending point

```
ARRAY[] = new Array[3]
ARRAY[0] = “VALUE 1”
ARRAY[1] = “VALUE 1”
ARRAY[2] = “VALUE 2”

For (i = 0; i < ARRAY.Length(); i++)
   PRINT ARRAY[i]
```

Loops: For Each

- A loop structure that executes code for each element in a list / array

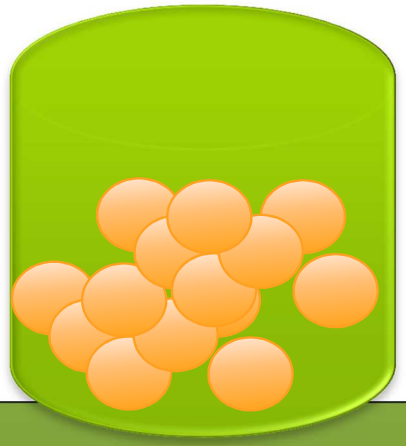
```
ARRAY[] = new Array[3]
ARRAY[0] = “VALUE 1”
ARRAY[1] = “VALUE 1”
ARRAY[2] = “VALUE 2”

ForEach (ITEM in ARRAY)
   PRINT ITEM
```

Loops In General

- Loops are typically used to iterate through a process or list of items (array)
- Very common control structure
- Almost every algorithm uses a loop of some sort

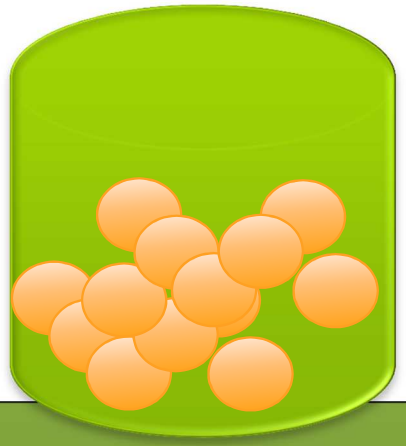
Loops In General

- Eating cookies
  - Let's say that you have decided that you are going to eat 5 cookies
  - What loop structure would you use?



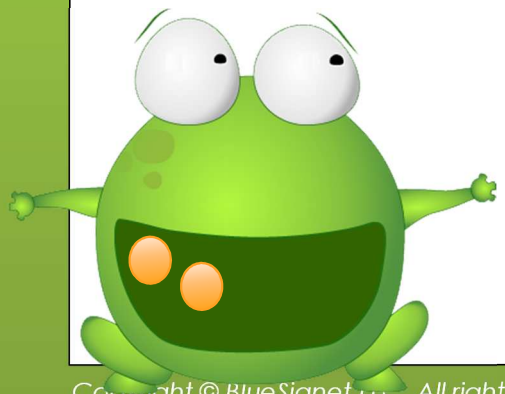
Loops In General

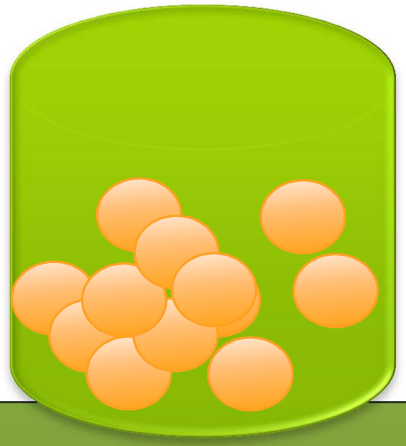
```
CookiesToEat = 5
CookiesEaten = 0
While (CookiesEaten < CookiesToEat )
  EAT COOKIE
  CookiesEaten = CookiesEaten + 1
```



Loops In General

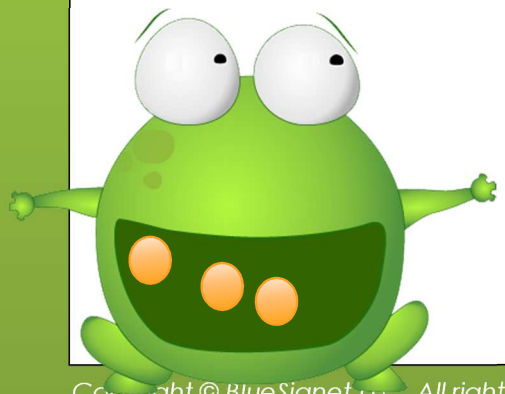
```
CookiesToEat = 5
CookiesEaten = 0
While (CookiesEaten < CookiesToEat )
  EAT COOKIE
  CookiesEaten = CookiesEaten + 1
```

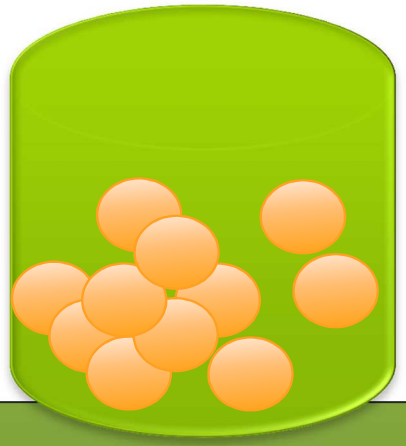




Loops In General

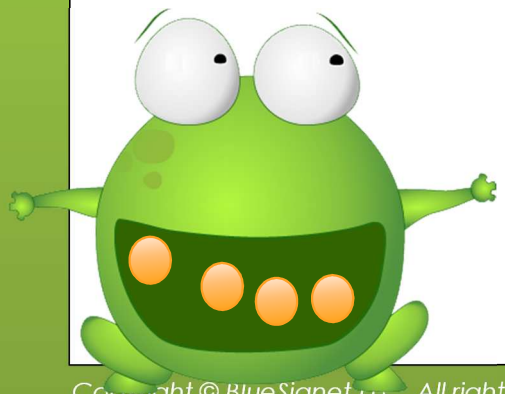
```
CookiesToEat = 5
CookiesEaten = 0
While (CookiesEaten < CookiesToEat )
  EAT COOKIE
  CookiesEaten = CookiesEaten + 1
```





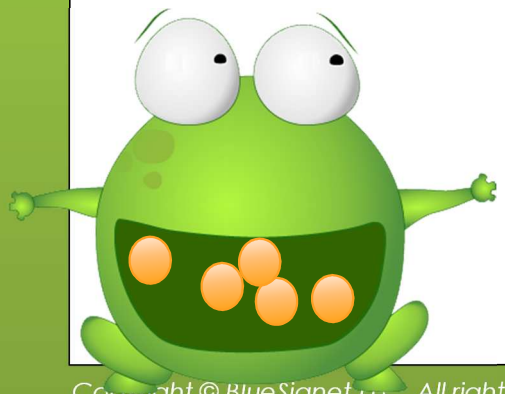
Loops In General

```
CookiesToEat = 5
CookiesEaten = 0
While (CookiesEaten < CookiesToEat )
  EAT COOKIE
  CookiesEaten = CookiesEaten + 1
```



Loops In General

```
CookiesToEat = 5
CookiesEaten = 0
While (CookiesEaten < CookiesToEat )
  EAT COOKIE
  CookiesEaten = CookiesEaten + 1
```

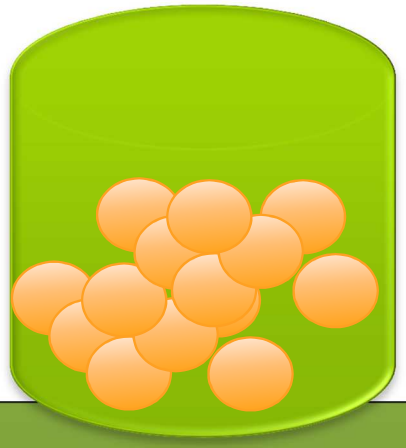


Loops In General

```
CookiesToEat = 5
CookiesEaten = 0
While (CookiesEaten < CookiesToEat )
  EAT COOKIE
  CookiesEaten = CookiesEaten + 1
```

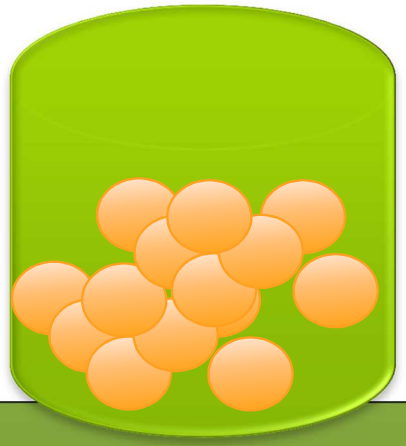
Loops In General

- Eating cookies
  - That's great! But isn't there another loop structure that we can use to eat exactly 5 cookies?



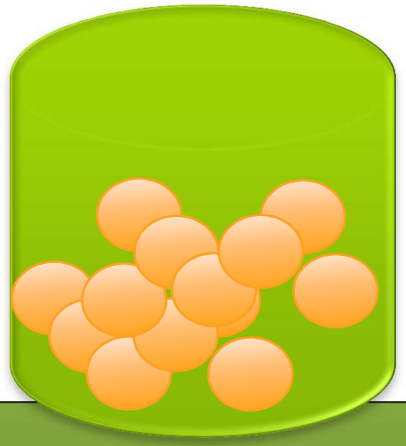
Loops In General

```
For (CookiesEaten = 0; CookiesEaten < 5;
  CookiesEaten++)
  EAT COOKIE
```



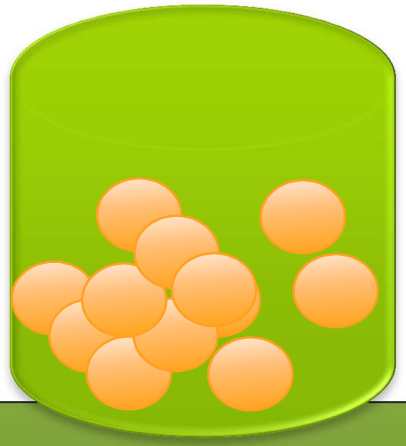
Loops In General

```
For (CookiesEaten = 0; CookiesEaten < 5;
  CookiesEaten++)
  EAT COOKIE
```



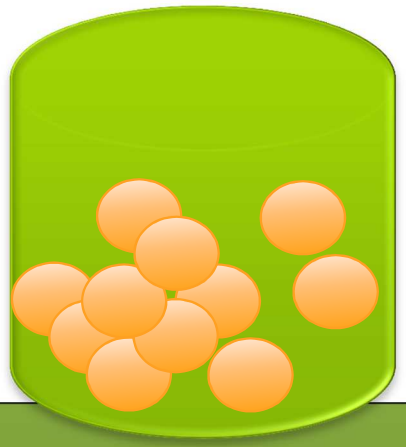
Loops In General

```
For (CookiesEaten = 0; CookiesEaten < 5;
  CookiesEaten++)
  EAT COOKIE
```



Loops In General

```
For (CookiesEaten = 0; CookiesEaten < 5;
  CookiesEaten++)
  EAT COOKIE
```



Loops In General

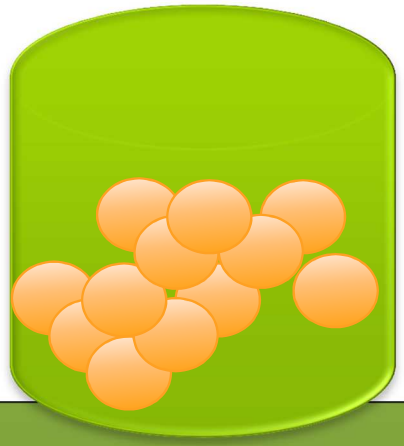
```
For (CookiesEaten = 0; CookiesEaten < 5;
  CookiesEaten++)
  EAT COOKIE
```

Loops In General

```
For (CookiesEaten = 0; CookiesEaten < 5;
  CookiesEaten++)
  EAT COOKIE
```

Loops In General

```
For (CookiesEaten = 0; CookiesEaten < 5;
  CookiesEaten++)
  EAT COOKIE
```



Loops In General

- Eating cookies
  - The next night you want to eat at least one cookie, but you will eat them until you become full
  - Which loop structure would you use?

Loops In General

```
EatMoreCookies = true
Do
  EAT COOKIE
  EatMoreCookies = AmIFull()
While ( EatMoreCookies == true )
```

Loops In General

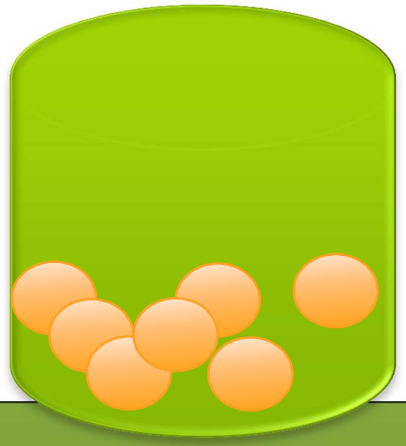
```
EatMoreCookies = true
Do
  EAT COOKIE
  EatMoreCookies = AmIFull()
While ( EatMoreCookies == true )
```

Loops In General

```
EatMoreCookies = true
Do
  EAT COOKIE
  EatMoreCookies = AmIFull()
While ( EatMoreCookies == true )
```

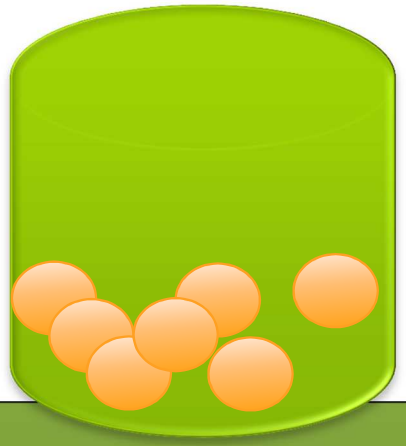
Loops In General

```
EatMoreCookies = true
Do
  EAT COOKIE
  EatMoreCookies = AmIFull()
While ( EatMoreCookies == true )
```



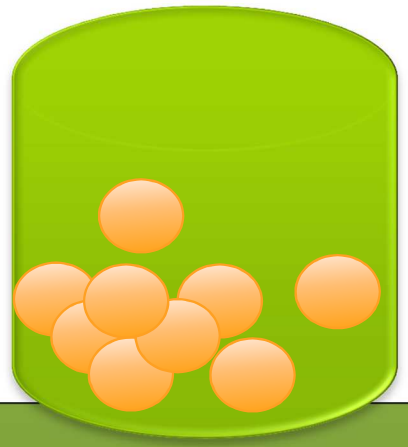
Loops In General

```
EatMoreCookies = true
Do
   EAT COOKIE
   EatMoreCookies = AmIFull()
While ( EatMoreCookies == true )
```



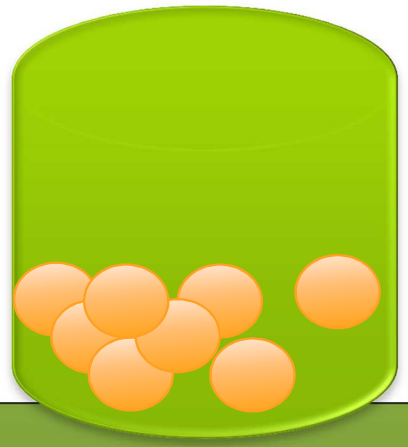
Loops In General

- Eating cookies
  - Later that night you want to eat the rest of the cookies in the jar
  - Which loop structure would you use?



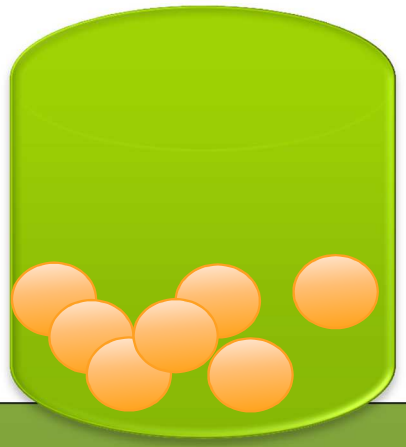
Loops In General

```
ForEach (COOKIE in JAR)
  EAT COOKIE
```



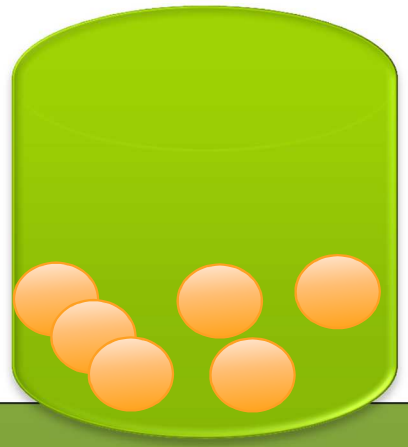
Loops In General

```
ForEach (COOKIE in JAR)
  EAT COOKIE
```



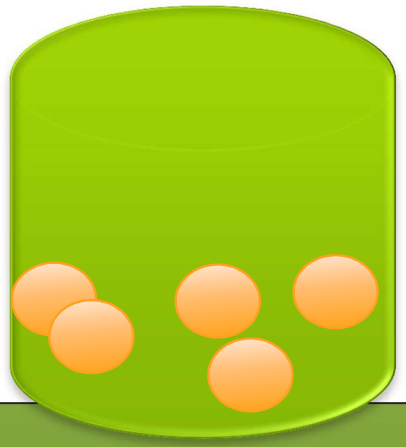
Loops In General

```
ForEach (COOKIE in JAR)
  EAT COOKIE
```



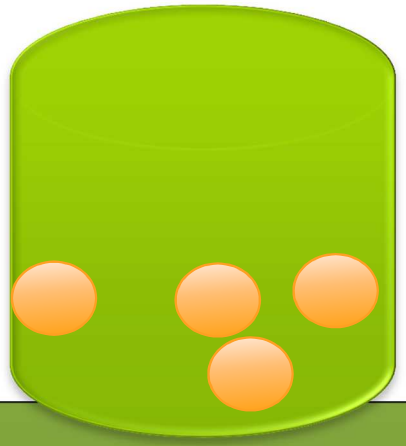
Loops In General

```
ForEach (COOKIE in JAR)
  EAT COOKIE
```



Loops In General

```
ForEach (COOKIE in JAR)
  EAT COOKIE
```



Loops In General

```
ForEach (COOKIE in JAR)
  EAT COOKIE
```

Loops In General

```
ForEach (COOKIE in JAR)
  EAT COOKIE
```

Loops In General

```
ForEach (COOKIE in JAR)
  EAT COOKIE
```

Loops In General

```
ForEach (COOKIE in JAR)
    EAT COOKIE
```

Classes

- An object oriented programming structure that consists of methods (functions) and attributes (variable types) to form a template for an object
- Object Oriented Programming
  - A programming structure that uses “object” data structures (Classes) to construct a computer program
- Object
  - An encapsulation of methods and variables (Potential)
- Instance
  - An occurrence of an object (Entity)

Person

Attributes

firstName

lastName

Person

Attributes

firstName

lastName

Methods

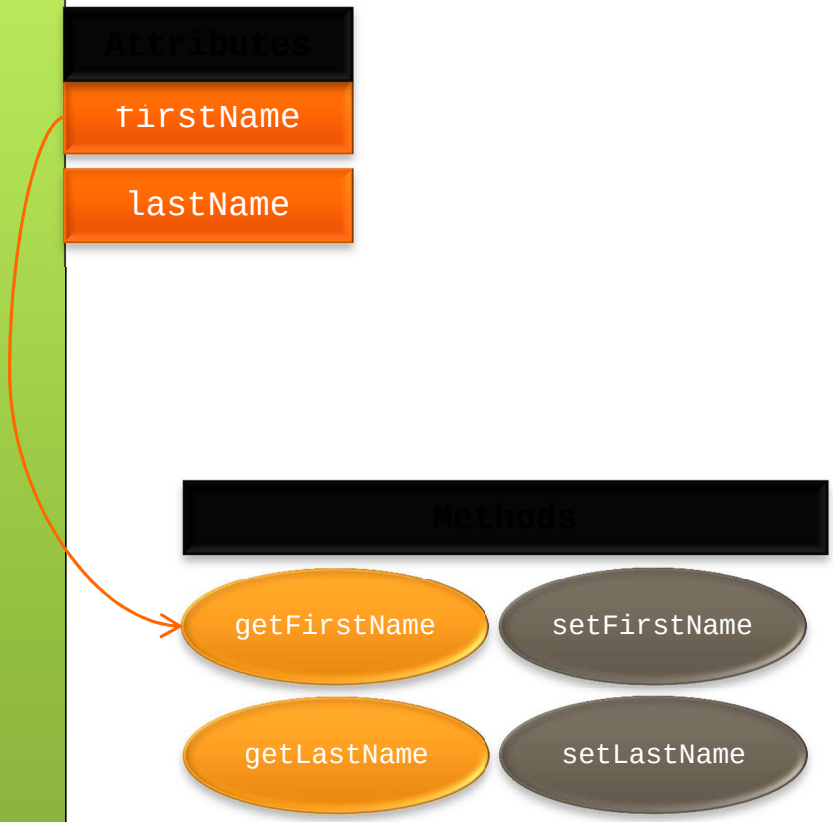
getFirstName

setFirstName

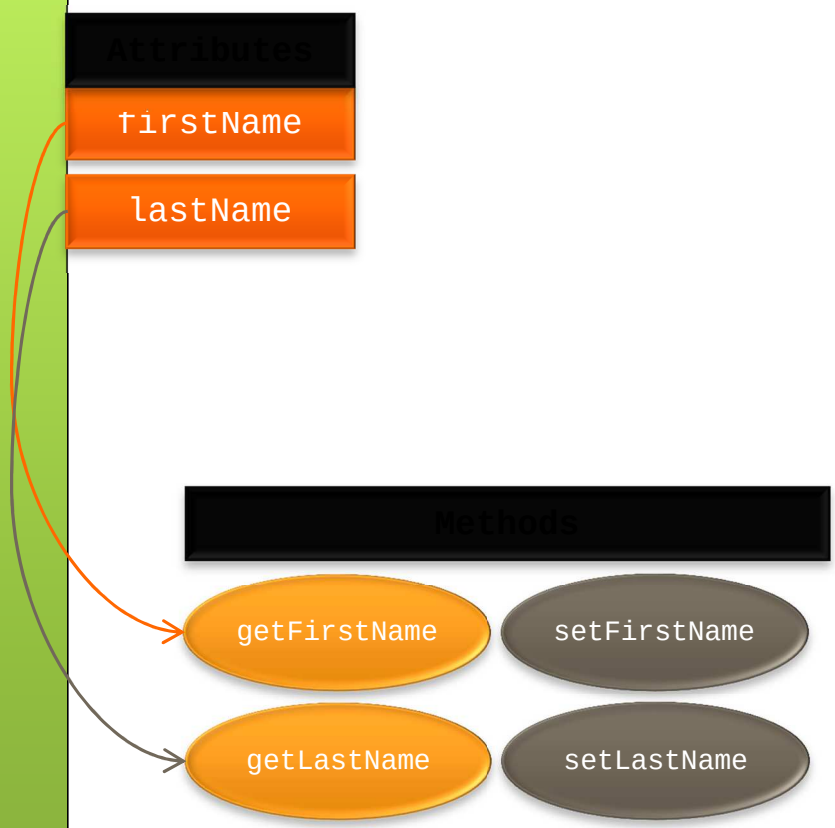
getLastName

setLastName

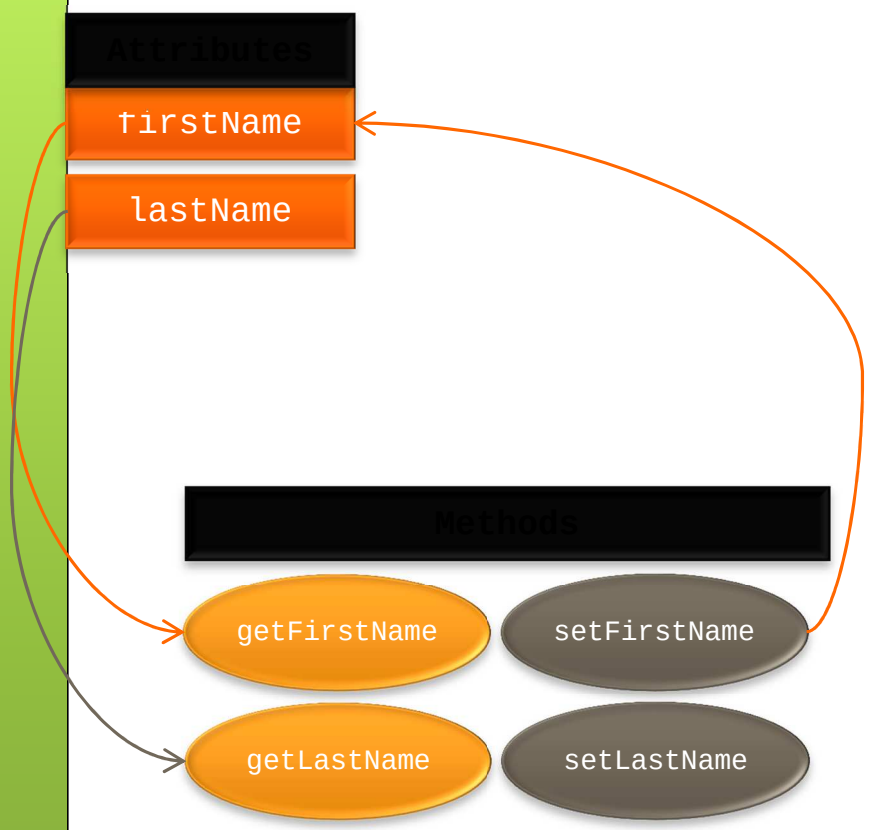
Person

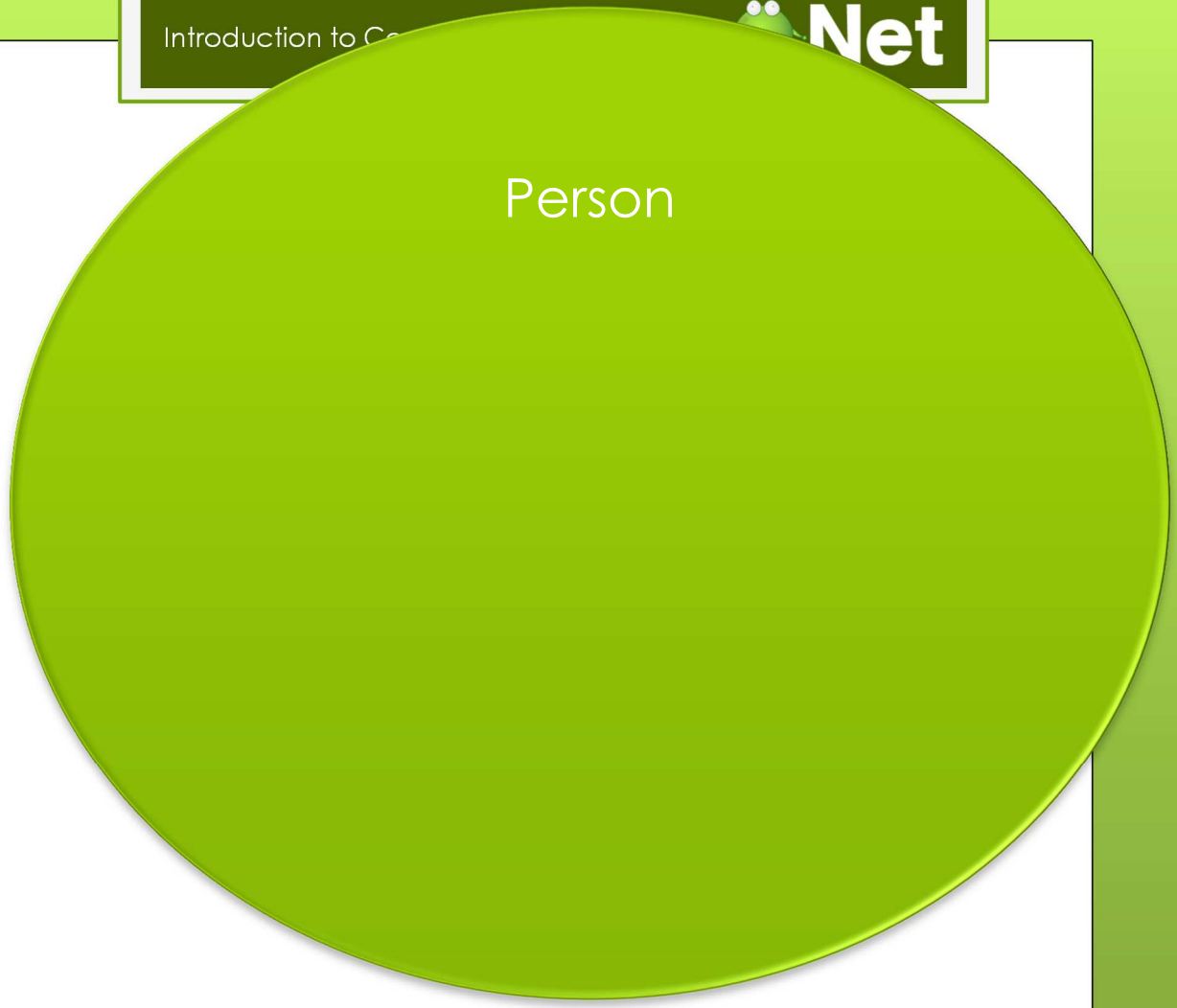


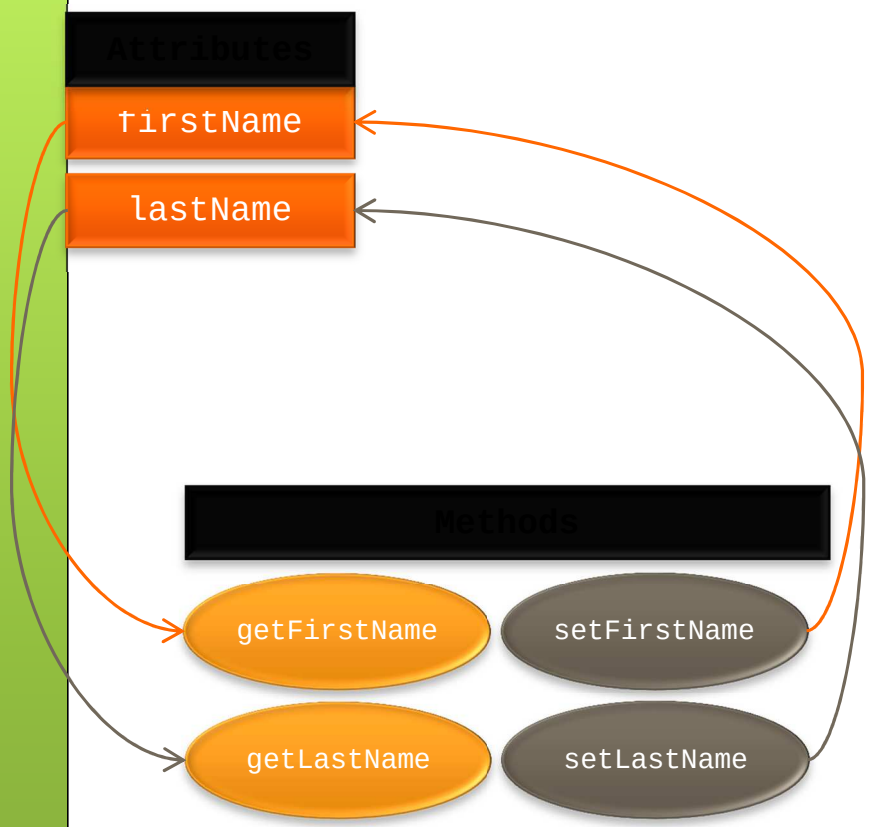
Person



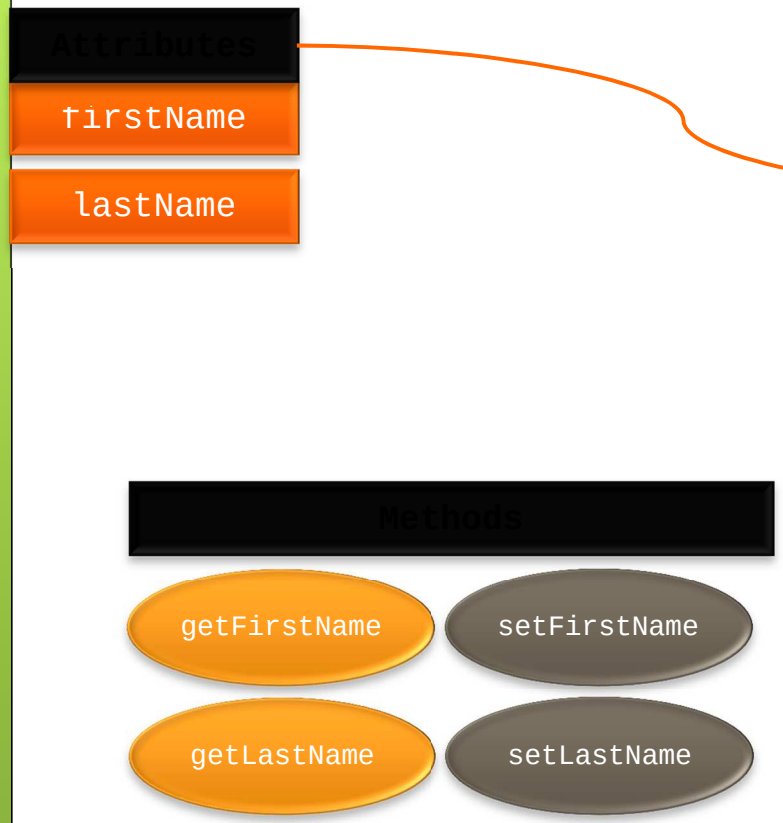
Person

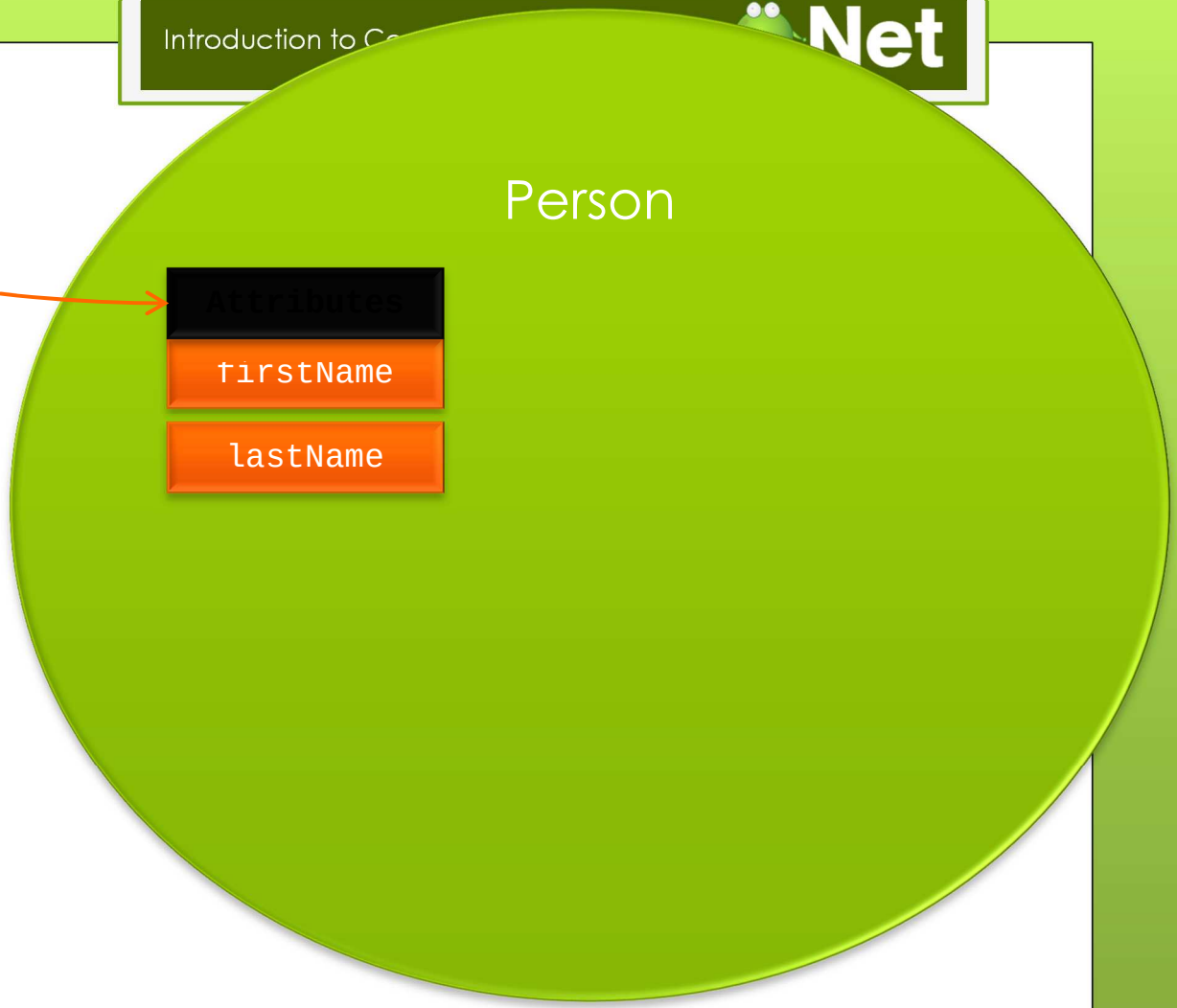


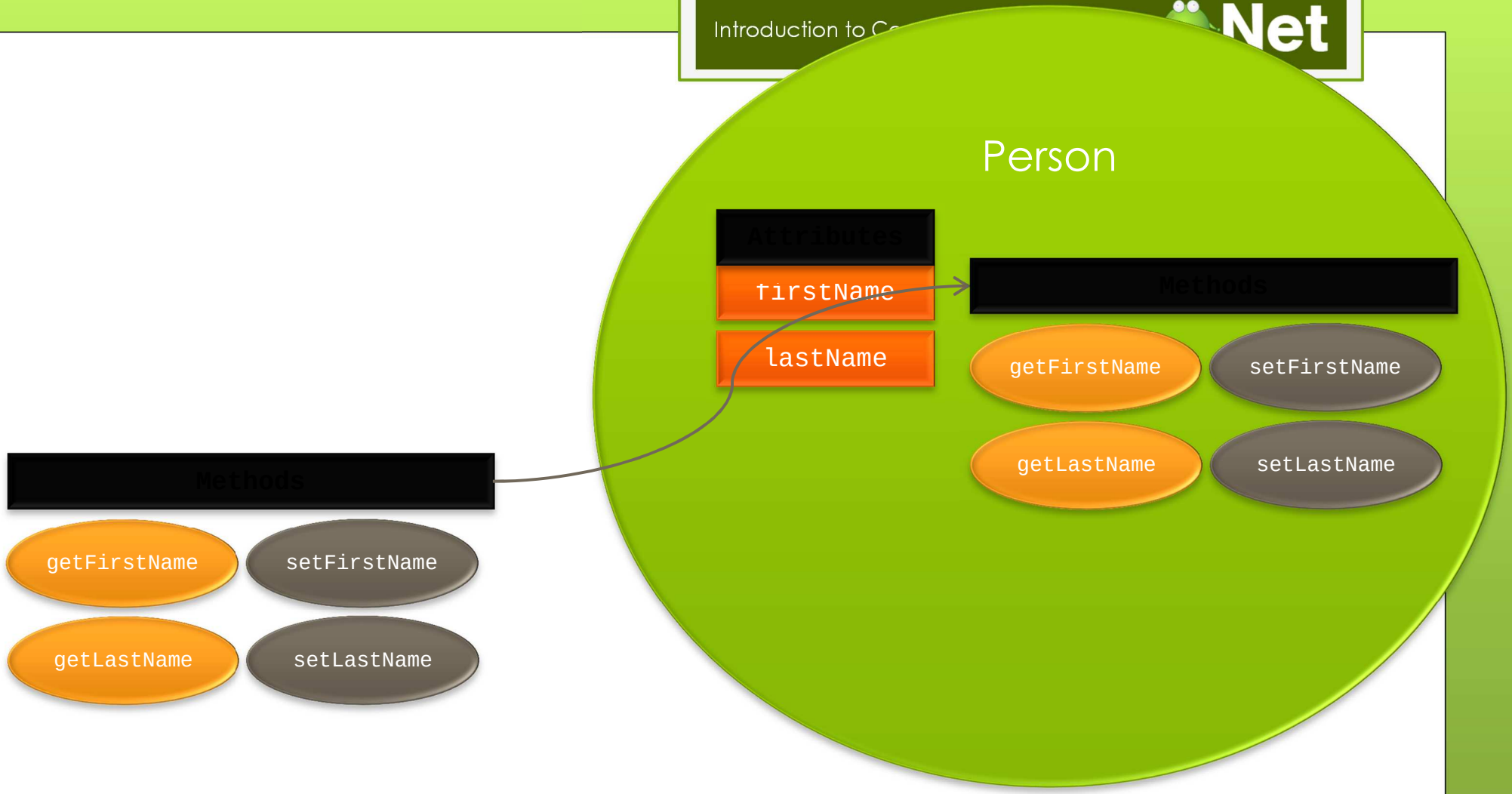


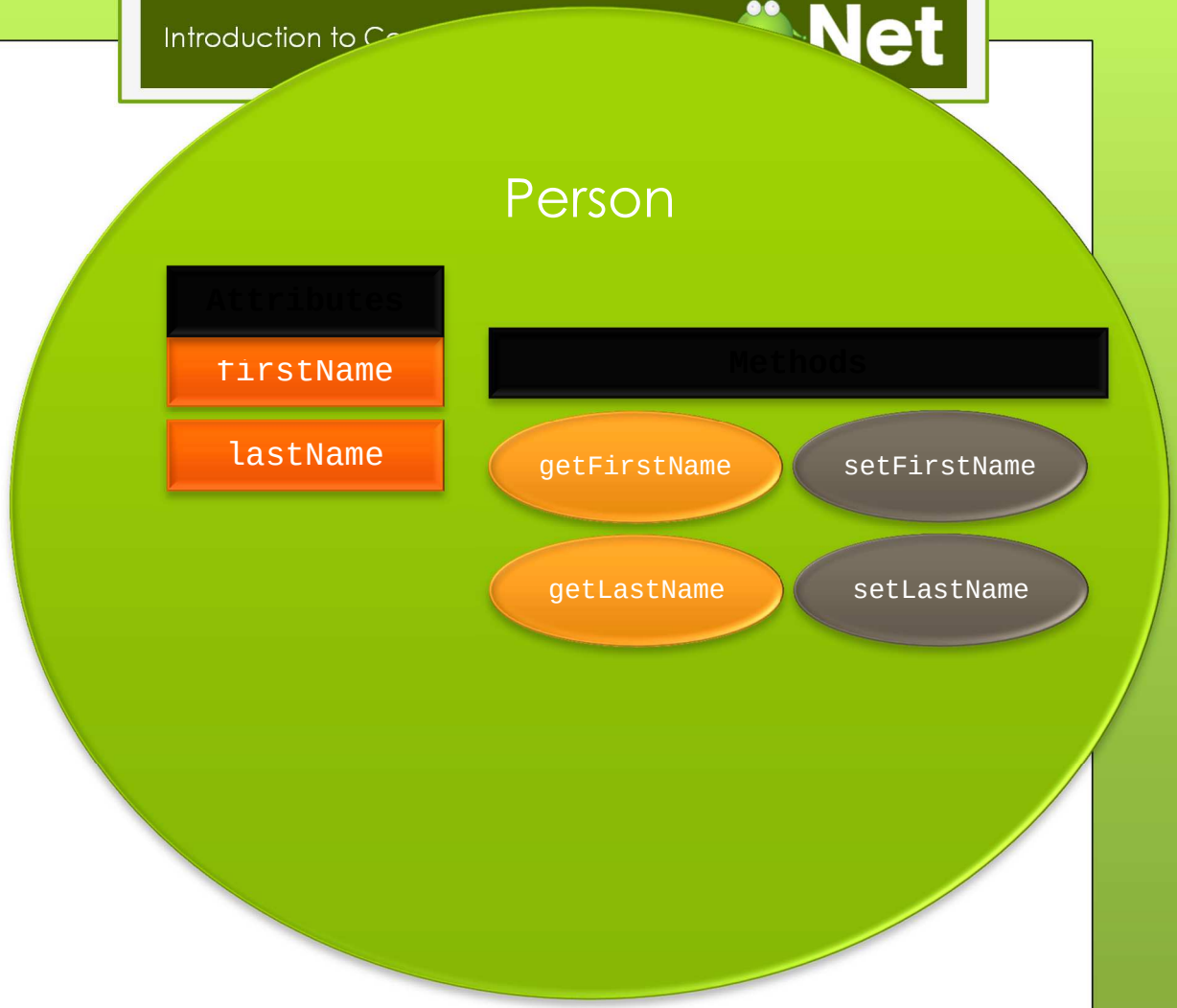


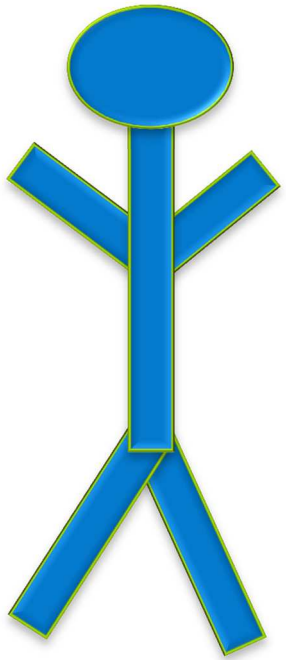
Person

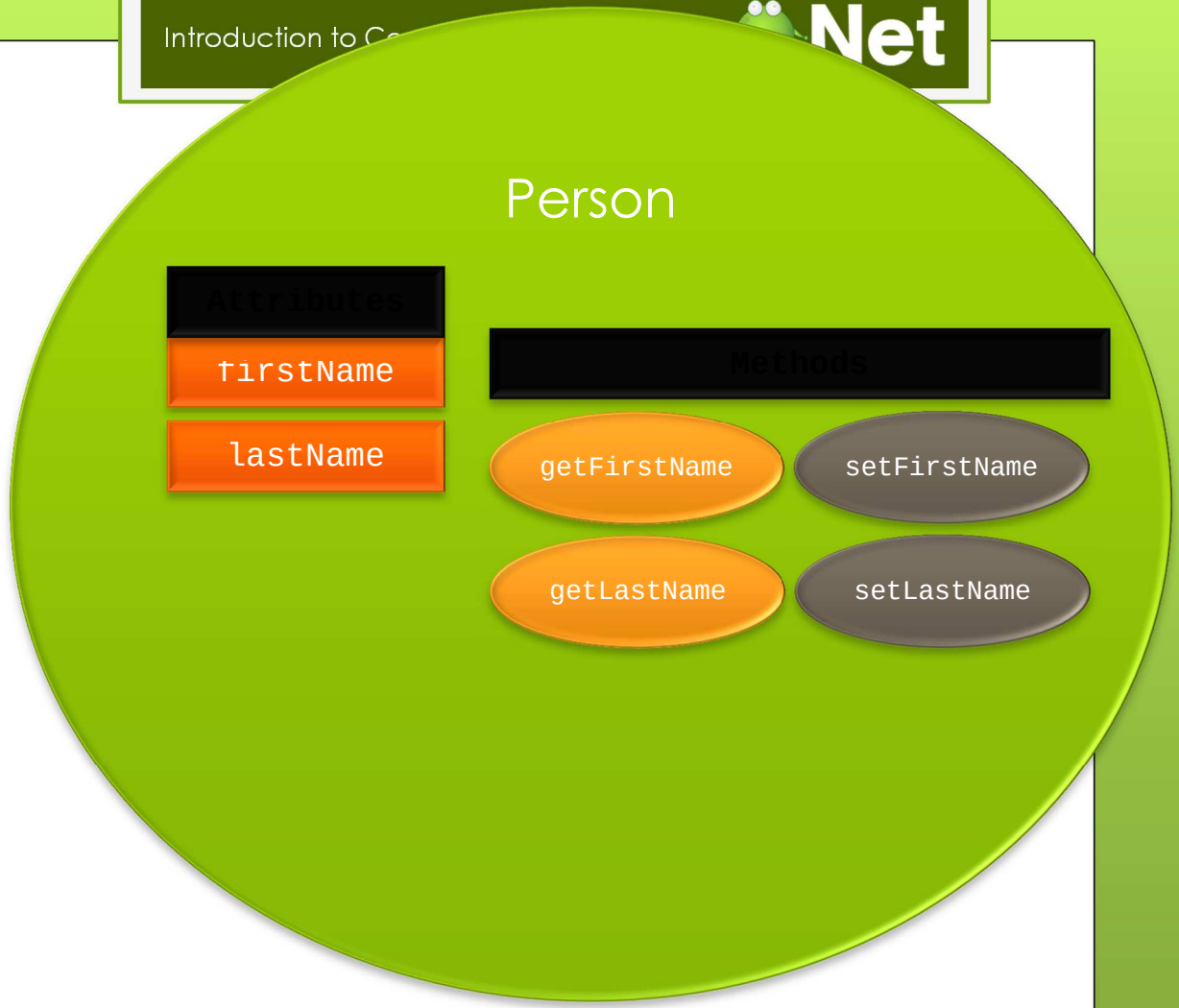


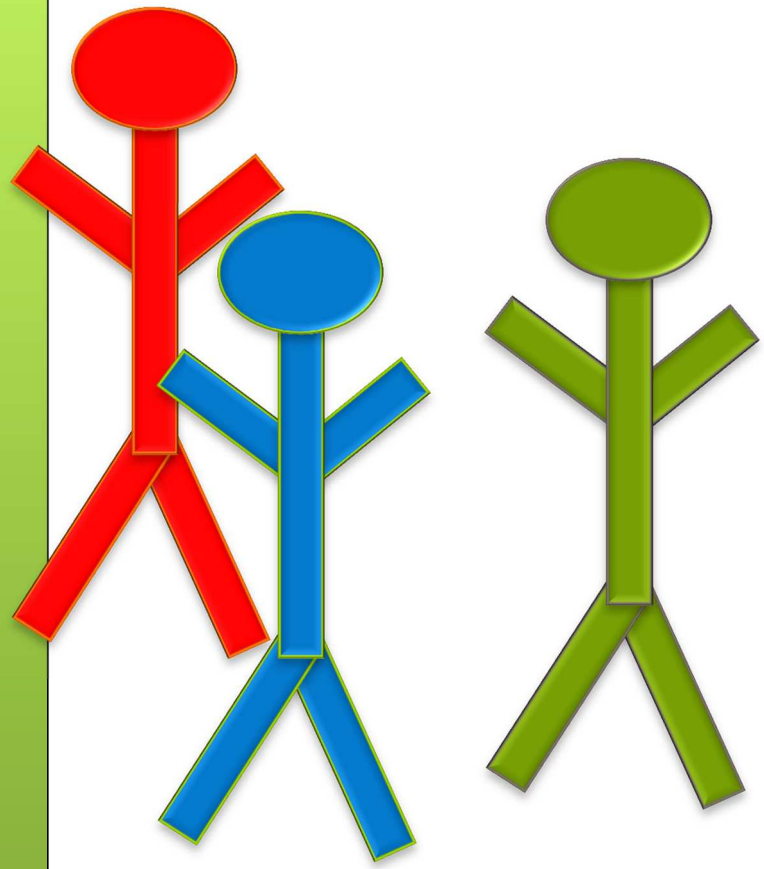


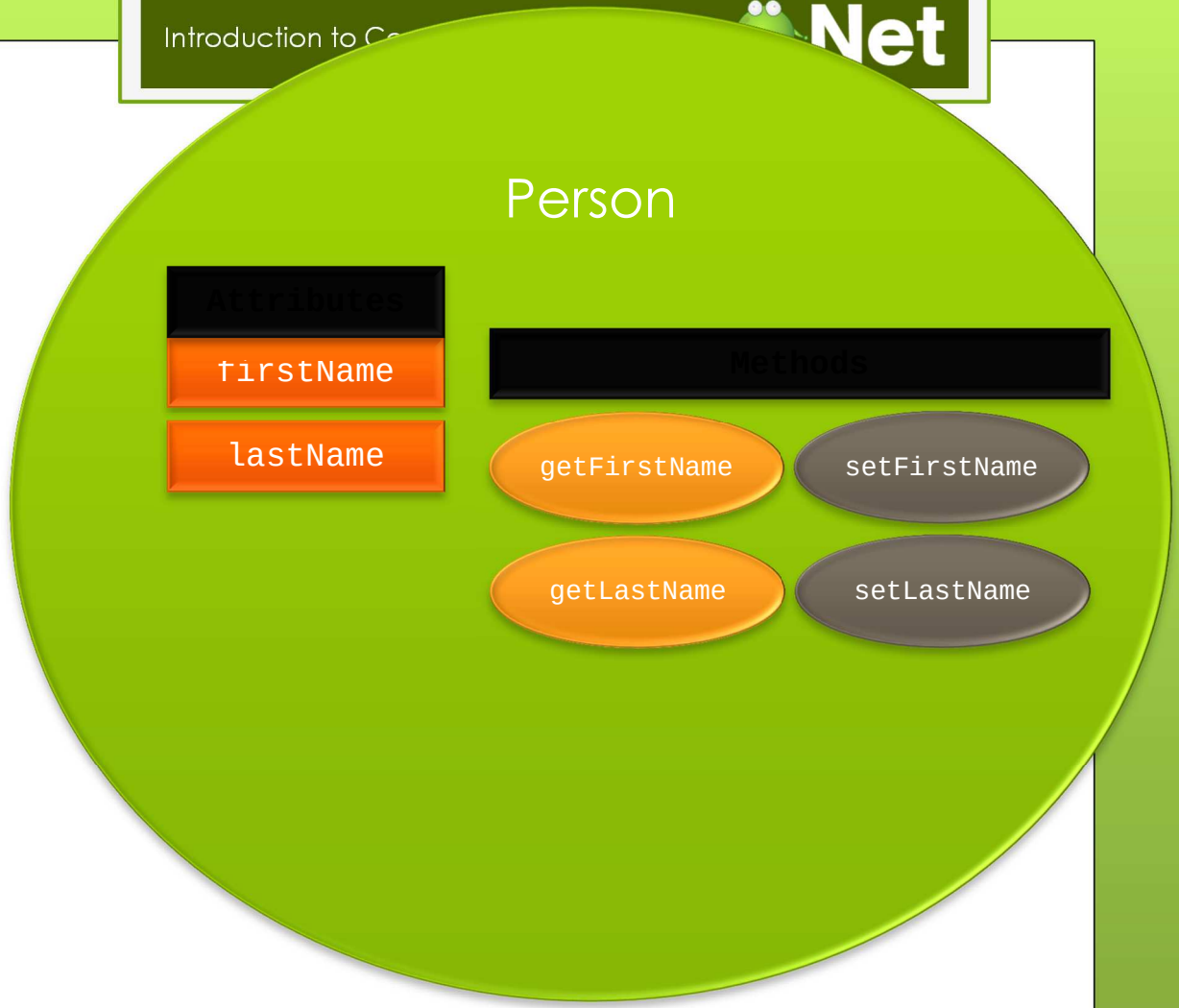












WiBitNet™

The End?