

1. Programming Basics



420-141-VA - GAME PROGRAMMING 1 - VANIER COLLEGE

Programming Basics Overview

Variables

- Types
- Member vs Local Variables

Classes vs. Objects

- Class diagrams, Subclasses
- Instances

Methods

- Constructors
- Member Methods
- Parameters
- Return Value

Algorithms

The faster you understand these concepts, the faster you can make games!

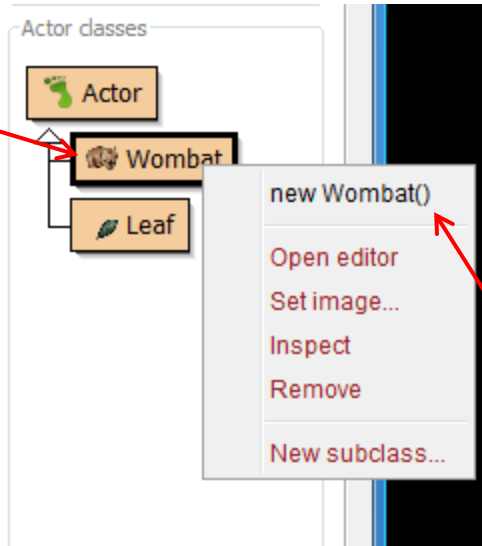
You may not master these concepts at the end of the class, but you must understand them within a few weeks.

These concepts apply to most Programming Languages (including Stride and Java)

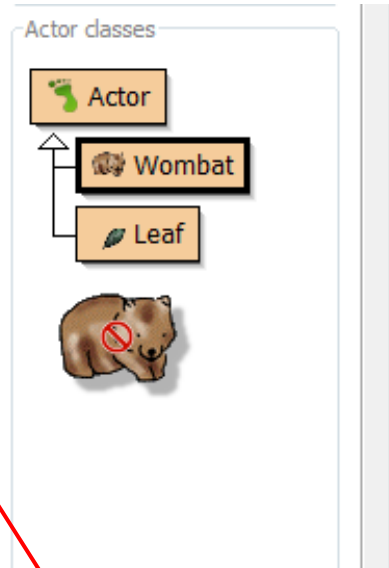
In this lecture, we use Greenfoot and Stride to illustrate the concepts

Classes vs. Objects

Right Click on
Wombat



Click New
Wombat()

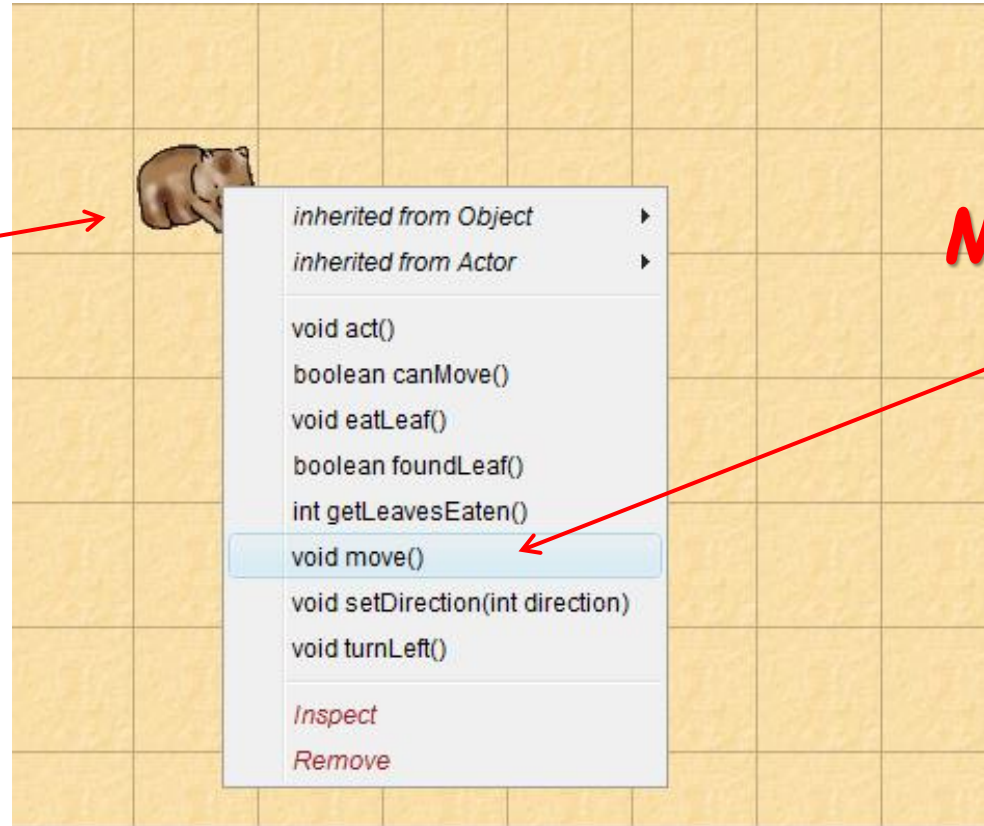


Drag to
World



Interacting with Actor Objects (calling methods)

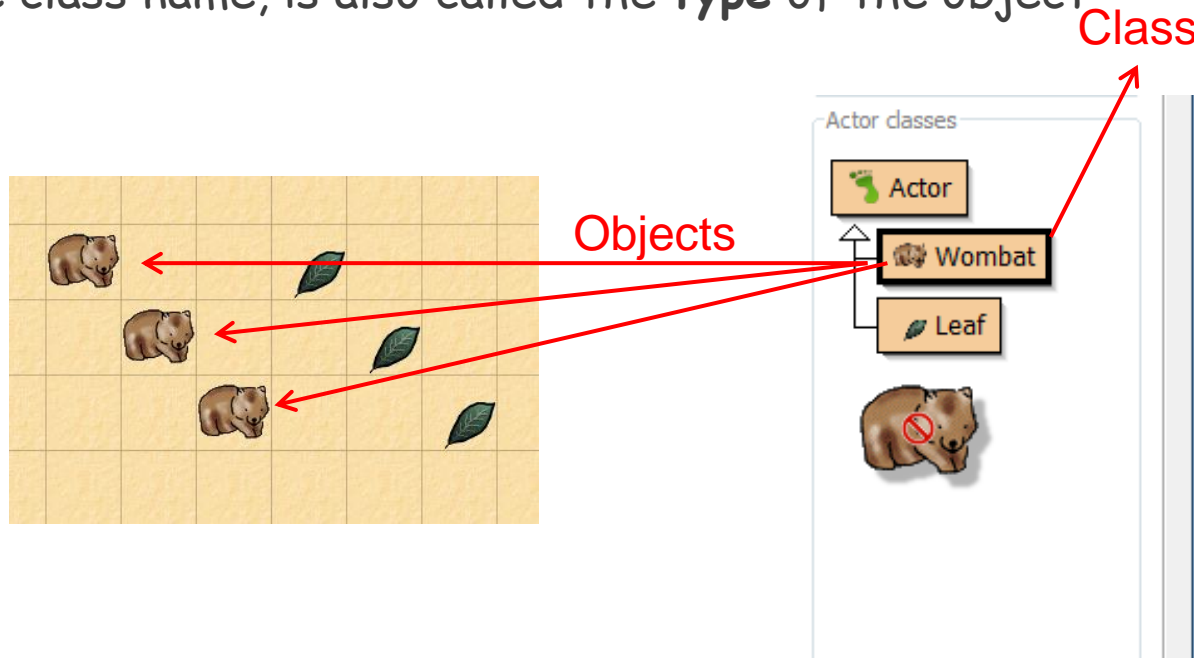
**Right Click
on the
Wombat**



**Invoke the
Move method**

Classes and objects

- A class provides the blueprints for creating objects
- When we code, we write classes
 - When we use the "new" operator, we create object instances
 - The class name, is also called the **type** of the object



```
Wombat 🦘 X
Imports ▶
Wombat. A Wombat moves forward until it hits the edge of the world,
at which point it turns left. If a wombat finds a leaf, it eats it.
@author Michael Kölling @version 2.0
class Wombat extends Actor ▶
Fields
(World, Actor, GreenfootImage, and Greenfoot)
private int leavesEaten
Constructors
Describe your constructor here...
public Wombat()
leavesEaten = 0
Methods
Do whatever the wombat likes to do just now.
public void act() overrides method from Actor
if ( foundLeaf() )
eatLeaf()
else if ( canMove() )
move()
else
turnLeft()
```

Variables

Variables are used to store information, they have a type and a name.

Types

- boolean : True or false, or from boolean expression ($7 < 3$)
- int : Integer value (Whole value, no decimals)
- double : Numeric value (including decimals)
- String : Some Text
- Actor : Greenfoot Game Object
- Wombat : User-defined class, specific to the Wombat game

Local variable : Variable declared within a method, visible locally

Member variables (fields) : Variable belonging to an object

Classes and objects

Local variables vs. Member variables (fields)

Local variables

- Methods may contain variables internally, such as parameters and variables declared within the method.

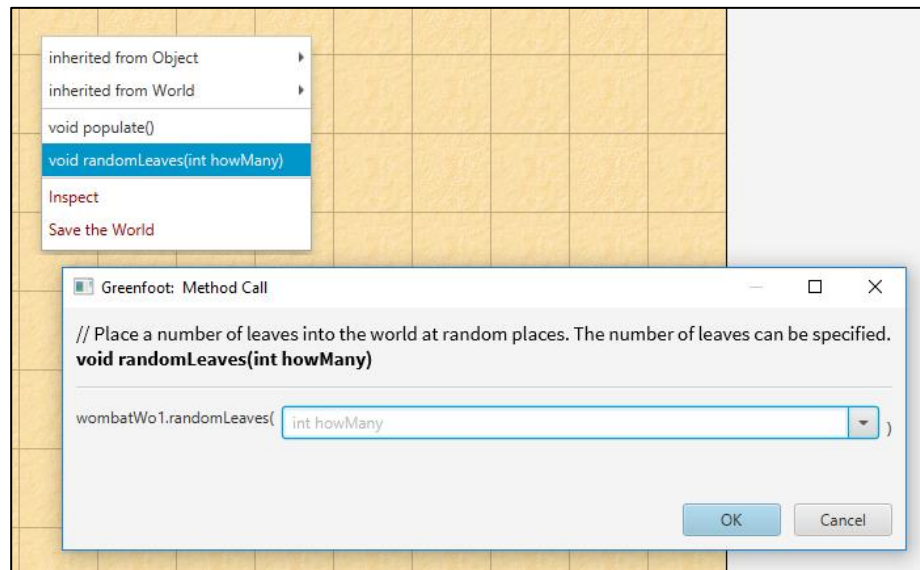
Member variables, or fields

- A class will contain **member variables**, or **fields**, used to define the state of an object.
- **Fields values** for different **objects** are generally independent

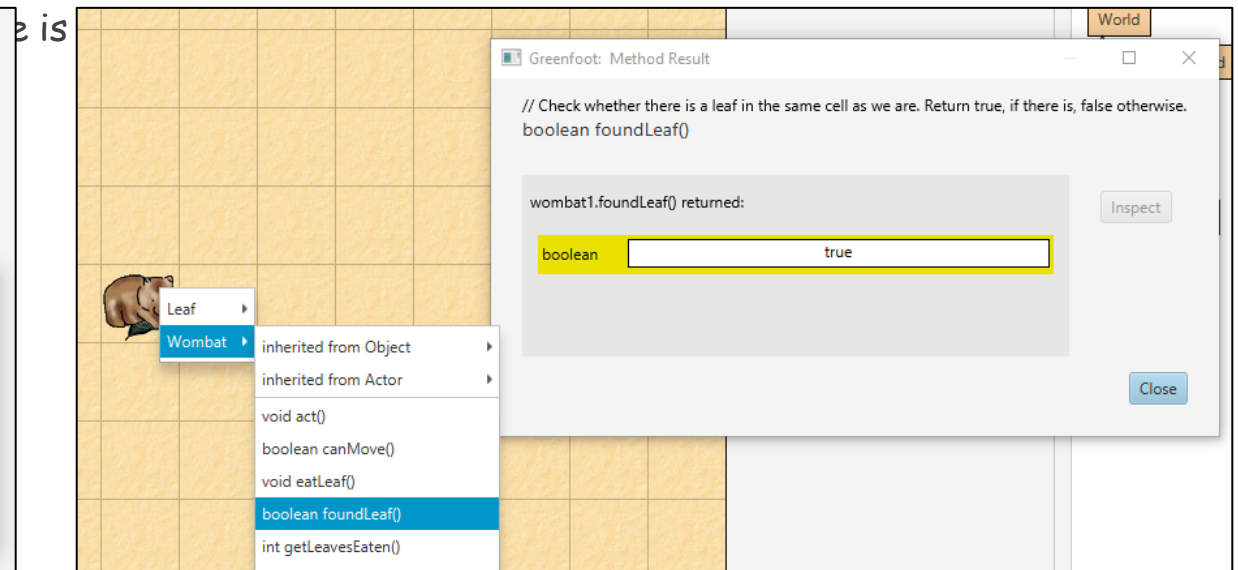
Classes and objects - Member Methods

A class will implement **member methods**, that will define behaviors for objects

- Methods may require one or many parameters (example on the left)
- Methods may **return** a value, which must be consistent with the **return type** (example on the right)



Parameter required for randomLeaves method



Method foundLeaf returns a boolean value (true or false)

Classes and objects - Constructors

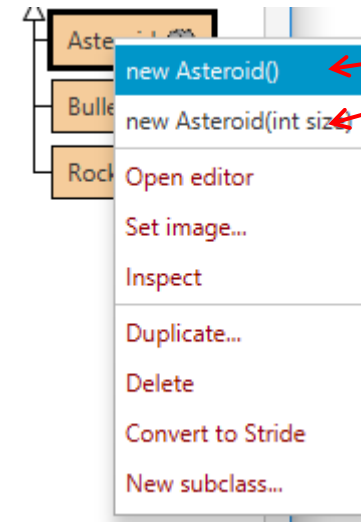
A class will implement a **constructor**, or many constructors, which are methods used to initialize instance objects member variables (fields)

The constructor method doesn't return anything, and has the same name as the class

To instantiate an object, we must call the **new** operator with the name of the class.

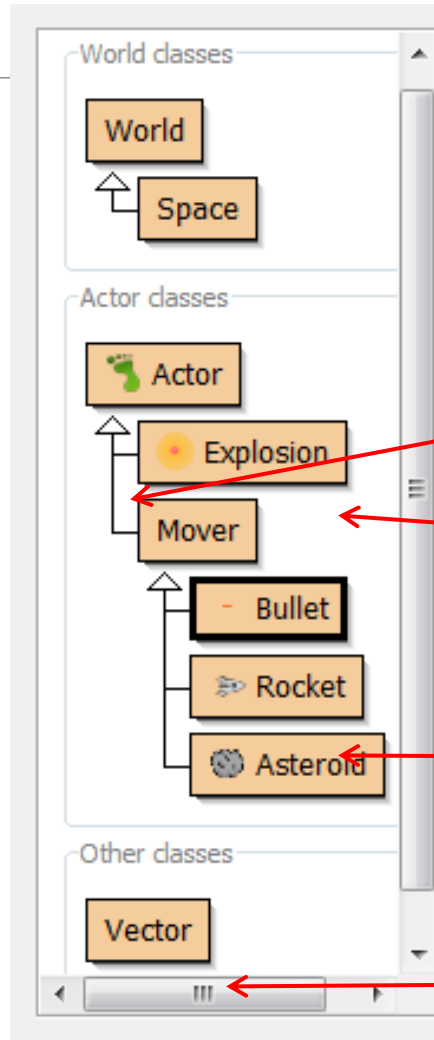
```
public void randomLeaves(int howMany)
{
    var int i = 0
    while ( i < howMany )
    {
        var Leaf leaf = new Leaf( )
        int x = Greenfoot.getRandomNumber( getWidth( ) )
        int y = Greenfoot.getRandomNumber( getHeight( ) )
        addObject( leaf, x, y )
        i = i + 1
    }
}
```

2 Constructors



Asteroid implements two constructors. One constructor initializes the size based on the parameter value

Understanding the Class Diagram



World Class is always there in Greenfoot scenarios, it is built-in.
Space represents a specific world for this scenario

Arrows show relationships

Explosion and Mover are subclasses of Actor

Bullet, Rocket, and Asteroid are subclasses of Mover.

Vector is a helper class

Algorithms

An algorithm is a sequence of instructions (or code statements) that will accomplish a specific task.

For example, the method **void randomLeaves(int howMany)** will add the number of leaves specified as parameter at random locations in the world.

```
public void randomLeaves(int howMany)
{
    var int i ← 0
    while ( i < howMany )
    {
        var Leaf leaf ← new Leaf( )
        int x ← Greenfoot.getRandomNumber (getWidth( ) )
        int y ← Greenfoot.getRandomNumber (getHeight( ) )
        addObject( leaf, x, y)
        i ← i + 1
    }
}
```

Comments

Reading code is sometimes difficult and cryptic, comments help clarify!

Comments can be inserted in a block of code to clarify it

Comments are also used to provide (or generate) documentation in a project



Java Packages - Greenfoot Package

Java includes vast amount of reusable code and features organized as packages.

Anyone can write packages. Some packages are built-in Java, while others are external.

Greenfoot provides classes for making games in the external Greenfoot package.

Right click on the **World** class, or **Actor** class to access the greenfoot package documentation

Package greenfoot

Class Summary

Class	Description
Actor	An Actor is an object that exists in the Greenfoot world.
Color	A representation of a Color.
Font	A representation of a Font.
Greenfoot	This utility class provides methods to control the simulation and interact with the system.
GreenfootImage	An image to be shown on screen.
GreenfootSound	Represents audio that can be played in Greenfoot.
MouseInfo	This class contains information about the current status of the mouse.
UserInfo	The UserInfo class can be used to store data permanently on a server, and to share this data between different users, when the scenario runs on the Greenfoot web site.
World	World is the world that Actors live in.

Class Documentation

Package `greenfoot`

Class `World`

`java.lang.Object`
`greenfoot.World`

```
public abstract class World
extends java.lang.Object
```

World is the world that Actors live in. It is a two-dimensional grid of cells.

All Actor are associated with a World and can get access to the world object. The size of cells can be specified at world creation time, and is constant after creation. Simple scenarios may use large cells that entirely contain the representations of objects in a single cell. More elaborate scenarios may use smaller cells (down to single pixel size) to achieve fine-grained placement and smoother animation.

The world background can be decorated with drawings or images.

Version:

2.6

Author:

Poul Henriksen, Michael Kolling

See Also:

`Actor`

Package `greenfoot`

Class `Actor`

`java.lang.Object`
`greenfoot.Actor`

```
public abstract class Actor
extends java.lang.Object
```

An Actor is an object that exists in the Greenfoot world. Every Actor has a location in the world, and an appearance (that is: an icon).

An Actor is not normally instantiated, but instead used as a superclass to more specific objects in the world. Every object that is intended to appear in the world must extend Actor. Subclasses can then define their own appearance and behaviour.

One of the most important aspects of this class is the 'act' method. This method is called when the 'Act' or 'Run' buttons are activated in the Greenfoot interface. The method here is empty, and subclasses normally provide their own implementations.

Version:

2.5

Author:

Poul Henriksen

Advice: Do not memorize all classes and methods, work with documentation!

Math Operators in Java (+ * / - %)

Addition

$3 + 4$

Multiplication

$3 * 4$

Division

$3 / 4$

Subtraction

$3 - 4$

Negation

-4

Modulo (Remainder)

$10 \% 2$ and $11 \% 2$

What is $3/2$?

- A. 1.5
- B. 1
- C. 0
- D. None of the above

Why is the result of $3 / 2 = 1$?

Java is a **strongly typed** language

- Each value has a type associated with it
- Tells the computer how to interpret the number
 - It is an integer, floating point, letter, etc

The compiler determines the type if it isn't specified (literals)

- 3 is an integer
- 3.0 is a floating point number (has a fractional part)

The result of an operation is in the same type as the operands

- 3 and 2 are integers so the answer is an integer 1

Casting

There are other ways to solve the problem of $3 / 2$ has a result of 1

You can make one of the values floating point by adding .0

- $3.0 / 2$
- $3 / 2.0$

The result type will then be floating point

Or you can cast one of the values to the primitive types: float or double

- $(\text{double}) 3 / 2$
- $3 / (\text{float}) 2$

Casting Exercise


Use casting to get the values right for splitting up a bill for 3 people of 19 dollars.

Try it first with a calculator

What would it be in Java using integer arithmetic without casting?

With casting?

What are the three data types that you have learned so far?

- A. cat, dog, goat
-  B. int, double, boolean
- C. String, int, short
- D. float, char, double

Java Primitive Data Types

- Integers (numbers without fractional parts) are represented by
 - The types: int or short or long
 - 235, -2, 33992093, etc.
- Floating point numbers (numbers with fractional parts) are represented by
 - The types: double or float
 - 3.233038983, -423.9, etc.
- A single character is represented by
 - The type: char
 - 'a' 'b' 'A' etc.
- True and false values are represented by
 - The type: boolean
 - true or false

Why so Many Different Types?

They take up different amounts of space

They have different precisions

Usually use int, double, and boolean

- byte uses 8 bits (1 byte) 2's complement
- short uses 16 bits (2 bytes) 2's complement
- int uses 32 bits (4 bytes) 2's complement
- long uses 64 bits (8 bytes) 2's complement
- float uses 32 bits (4 bytes) IEEE 754
- double uses 64 bits (8 bytes) IEEE 754
- char uses 16 bits (2 bytes) Unicode format

Sizes of Primitive Types

byte	8 bits								
short	8 bits	8 bits							
int	8 bits	8 bits	8 bits	8 bits					
long	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	
float	8 bits	8 bits	8 bits	8 bits					
double	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	
char	8 bits	8 bits							

Types Exercise

Which type(s) take up the most space?

Which type(s) take up the least space?

What type would you use for

- The number of people in your family
- A grade
- The price of an item
- The answer to do you have insurance
- The number of people in the class
- The number of people in your school
- The number of people in your state

Floating Point Numbers

Numbers with a fractional part

- 6170.20389

Stored as binary numbers in scientific notation -52.202 is $-.52202 \times 10^2$

- The sign (1 bit)
- The digits in the number (mantissa)
- The exponent (8 bits)

Two types

- float - 6-7 significant digits accuracy
- double - 14-15 significant digits accuracy

Comparison (Relational) Operators

Greater than >

- 4 > 3 is true
- 3 > 3 is false
- 3 > 4 is false

Less than <

- 2 < 3 is true
- 3 < 2 is false

Equal ==

- 3 == 3 is true
- 3 == 4 is false

Not equal !=

- 3 != 4 is true
- 3 != 3 is false

Greater than or equal >=

- 3 >= 4 is true
- 3 >= 3 is true
- 2 >= 4 is false

Less than or equal <=

- 2 <= 3 is true
- 2 <= 2 is true
- 4 <= 2 is false

$$-5 < -6$$

A. True

B. False

Operator Order

The default evaluation order is

- Negation -
- Multiplication *, Division /, and Modulo (remainder) %
 - Go left-to-right at the same level
- Addition + and Subtraction -
 - Go left-to-right at the same level

The default order can be changed

- By using parenthesis
- $(3 + 4) * 2$ versus $3 + 4 * 2$

What is $2 + 3 * 4 + 5$?

- A. 18
- B. 19
- C. 25
- D. 3.14159265

A Semicolon (;) ends a Statement

Java programs are made up of statements

- Like sentences in English

Java statements end in a semicolon not a period

- The period is used to send a message to an object like
 - `System.out.println();`

Strings

Java has a type called: String

A string is an object that has a sequence of characters in Unicode

- It can have no characters (the null string "")
- It can have many characters
 - "This is one long string with spaces in it. "
- Everything in a string will be printed out as it was entered
 - Even math operations "128 + 234"

Java knows how to add strings

- It returns a new string with the characters of the second string after the characters of the first
 - With no added space

Java is Case Sensitive

Some programming languages are case sensitive

- Meaning that **double** isn't the same as **Double**
- Or **string** isn't the same as **String**

In Java primitive types are all lowercase

- double, float, int,

Class names start with an uppercase letter

- So String and System are the names of classes

Java Naming Conventions

In Java **only** Class names start with an uppercase letter

- System, BufferedImage, Picture

All other names start with lowercase letters but uppercase the first letter of each additional word

- picture, fileName, thisIsALongName

Questions

?