Week 13 - Wednesday

COMP 1800

Last time

- What did we talk about last time?
- Simulations
- UML
- Fish and bears example

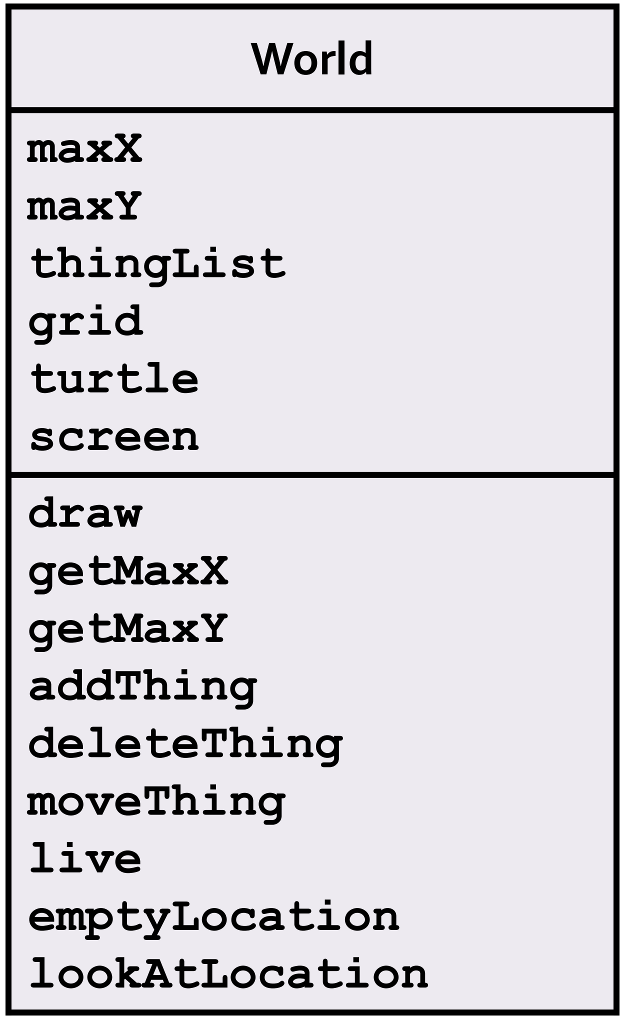
Questions?

Assignment 9

Fish and Bears

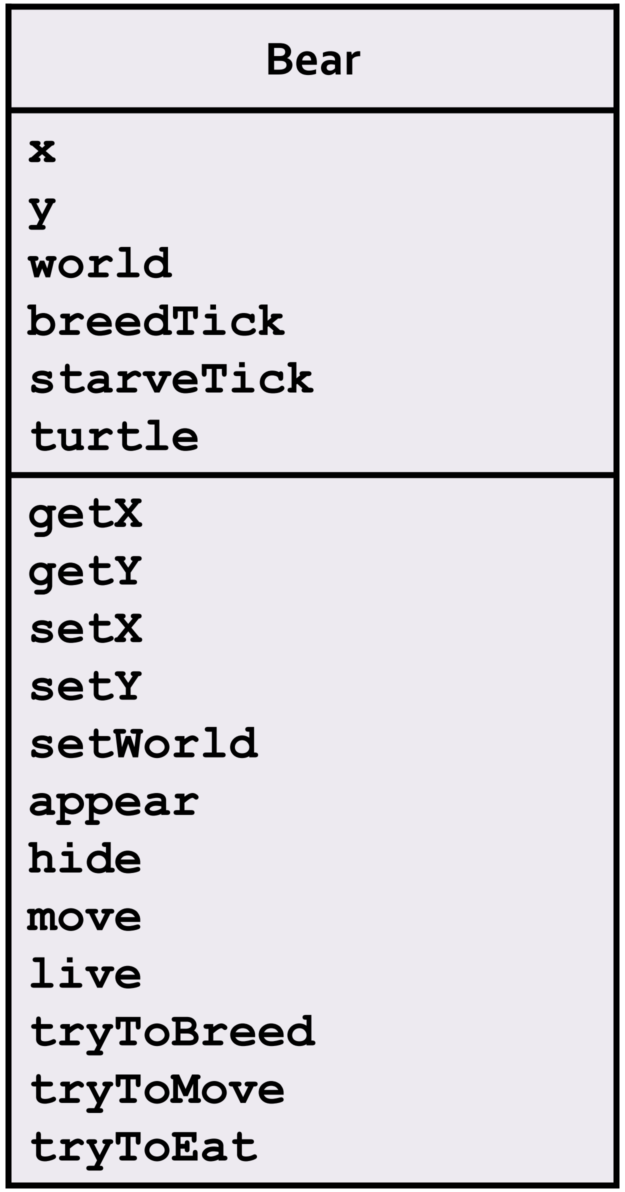
Class diagram for `World`

- Here is a UML class diagram for the **`World`** class



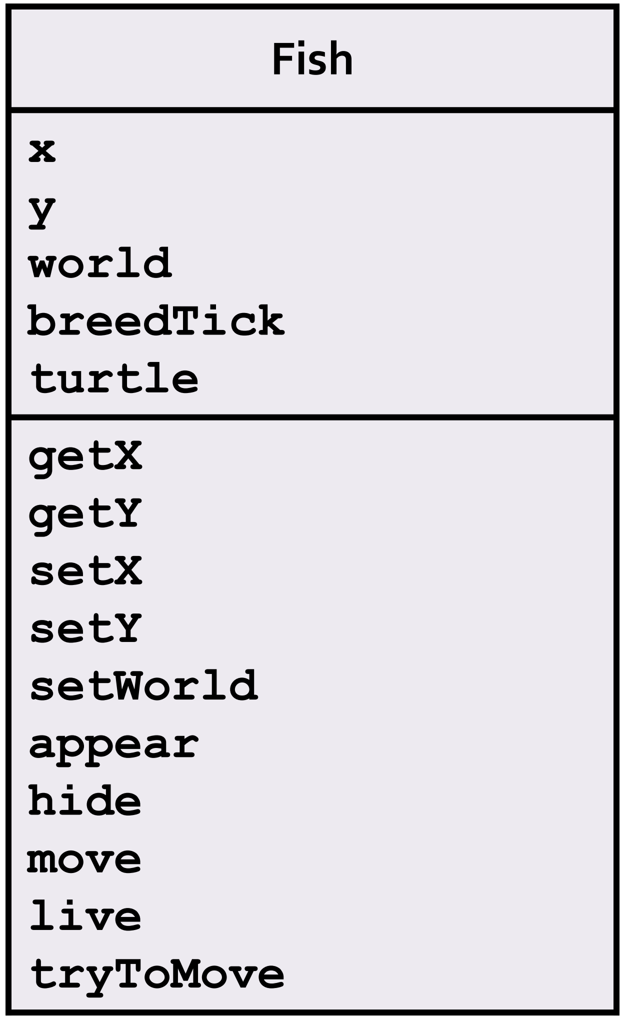
Class diagram for **Bear**

- Here is a UML class diagram for the **`Bear`** class



Class diagram for `Fish`

- Here is a UML class diagram for the **`Fish`** class



`Bear` Class

Implementing Bear

- Create a constructor for **Bear** with the following header:

```
def __init__(self):
```

- It should:
  - Create a turtle
  - Put the turtle's tail up
  - Hide the turtle
  - Set the turtle's shape to a square (since we don't have cool bear and fish pictures like the book does)
  - Set the **x** and **y** to 0
  - Set the **world** to **None**
  - Set the **breedTick** to 0
  - Set the **starveTick** to 0

Accessors for **Bear**

- Write the following accessors for **`Bear`**

```
def getX(self):
```

```
def getY(self):
```

Mutators for Bear

- Write the following mutators for **Bear**

```
def setX(self):
```

```
def setY(self):
```

```
def setWorld(self):
```

```
def appear(self): # move turtle to x and y and show
```

```
def hide(self): # hide turtle
```

`move()` method for `Bear`

- Write a method with the following header:

```
def move(self, newX, newY):
```

- It should:
  - Tell world to move a thing from the current x and y to the new ones
  - Set the **x** and **y** values to the new ones
  - Move the turtle the new location as well

Code to place **Fish** and **Bear** objects

- The following code creates a **World** and places **Fish** and **Bear** objects

```
bearCount = 10
fishCount = 10
worldLife = 2500
width = 50
height = 25
world = World(width, height)
world.draw()
for i in range(fishCount):
    fish = Fish()
    x = random.randrange(width)
    y = random.randrange(height)
    while not world.emptyLocation(x, y):
        x = random.randrange(width)
        y = random.randrange(height)
    world.addThing(fish, x, y)
for i in range(bearCount):
    bear = Bear()
    x = random.randrange(width)
    y = random.randrange(height)
    while not world.emptyLocation(x, y):
        x = random.randrange(width)
        y = random.randrange(height)
    world.addThing(bear, x, y)
```

Coming up

- Now we can create **`Fish`** and **`Bear`** objects and put them in a **`World`** object
- The next step is to give them behaviors:
  - Moving
  - Breeding
  - Eating
- We can even add other kinds of objects to the ecosystem

`isinstance()`

Determining types in Python

- Types are pretty loose in Python
- You can say **`x = 5`** and later **`x = 'goat'`**, and **`x`** will have a different type based on what's inside of it
- You can use the **`type()`** function to see what the type of something currently is

```
x = 5
print(type(x)) # prints <class 'int'>
x = 'goat'
print(type(x)) # prints <class 'str'>
```

`isinstance()`

- If you want to test to see if a variable has a certain type, you can also use the **`isinstance()`** function
- It's useful for **`if`** statements
- It will also help us find out if an object is a **`Fish`** or a **`Bear`**

```
x = 5
if isinstance(x, int):
    print("It's an int!")
else:
    print("What's going on?")
```

Fish Behavior

Searching the neighborhood

- There are a few ways to look at the eight neighboring spaces around a fish (or a bear)
- One way is to look at all the offsets from a list
- For example, if your location is (x, y), you can add each of the following offsets to get all possible neighboring locations:

$(x - 1, y + 1)$	$(x, y + 1)$	$(x + 1, y + 1)$
$(x - 1, y)$	(x, y)	$(x + 1, y)$
$(x - 1, y - 1)$	$(x, y - 1)$	$(x + 1, y - 1)$

```
offsets = [(-1,1), (0,1), (1,1), (-1,0), (1,0), (-1,-1), (0,-1), (1,-1)]
```

Fish life

- When a fish gets a turn to live, this is what it does:
  - Counts the fish that are near it (occupying the eight neighboring spaces)
  - If there are two or more neighboring fish, it dies (removing itself from the world)
  - Otherwise,
    - Increase its breeding counter by one
    - If its breeding counter is twelve or more,
      - Try to breed
    - Try to move

```
def live(self):
```

`tryToBreed()` method for `Fish`

- Write a method with the following header:

```
def tryToBreed(self):
```

- It should:
  - Randomly pick a location using the list of eight possible offsets
  - (Keep picking offsets if the location is out of bounds)
  - If the random location is empty,
    - Create a new fish in that location
    - Add the new object to the world
    - Set its breeding counter to 0

`tryToMove()` method for `Fish`

- Write a method with the following header:

```
def tryToMove(self):
```

- It should:
  - Randomly pick a location using the list of eight possible offsets
  - (Keep picking offsets if the location is out of bounds)
  - If the random location is empty,
    - Move to that location

Bear Behavior

Bear life

- When a bear gets a turn to live, this is what it does:
  - Increase its breeding counter by one
  - If its breeding counter is eight or more,
    - Try to breed
  - It should try to eat
  - If its starving counter is ten,
    - It dies (removing itself from the world)
  - Otherwise,
    - Try to move

```
def live(self):
```

`tryToBreed()` method for **Bear**

- Write a method with the following header (which works almost exactly like the same method for **`Fish`**):

```
def tryToBreed(self):
```

- It should:
  - Randomly pick a location using the list of eight possible offsets
  - (Keep picking offsets if the location is out of bounds)
  - If the random location is empty,
    - Create a new bear in that location
    - Add the new object to the world
    - Set its breeding counter to 0

`tryToMove()` method for Bear

- Write a method with the following header (which works exactly like the same method for **`Fish`**):

```
def tryToMove(self):
```

- It should:
  - Randomly pick a location using the list of eight possible offsets
  - (Keep picking offsets if the location is out of bounds)
  - If the random location is empty,
    - Move to that location

`tryToEat()` method for `Bear`

- Write a method with the following header :

```
def tryToEat(self):
```

- It should:
  - Look through all eight neighbors, using the list of offsets
    - If any of those neighbors is not empty and is also a fish,
      - Add that fish to a list of possible prey
  - If there is at least one fish in the list of prey,
    - Randomly pick one
    - Delete that fish from the world
    - Move to the location of that fish
    - Set the starving counter to zero
  - Otherwise,
    - Increase the starving counter

Finishing the simulation

- After all the classes have been written and the code to place the initial set of **`Bear`** and **`Fish`** objects runs, we only need one short loop to run the simulation

```
for i in range(worldLife):
    world.live()
```

Quiz

Upcoming

Next time…

- Introduce inheritance
- Work time for Assignment 9

Reminders

- Finish Assignment 9
  - **Due Friday**
- Start reading Chapter 12