

AP COMPUTER SCIENCE A

IMPORTANT REVIEW TOPICS

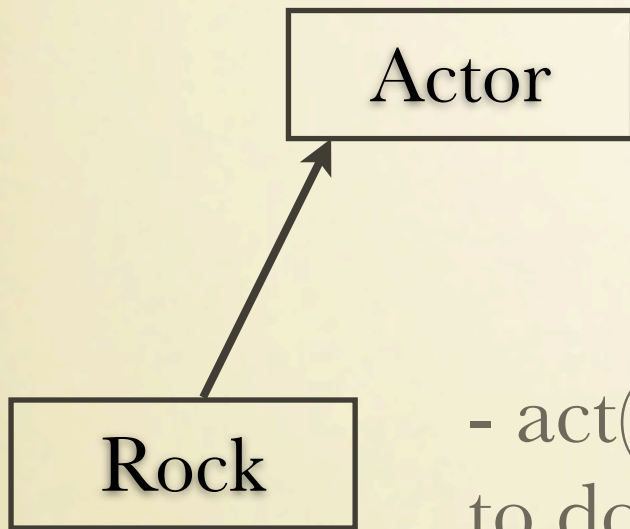
GRIDWORLD

Actor

- act() method always turns actor 180° to the right

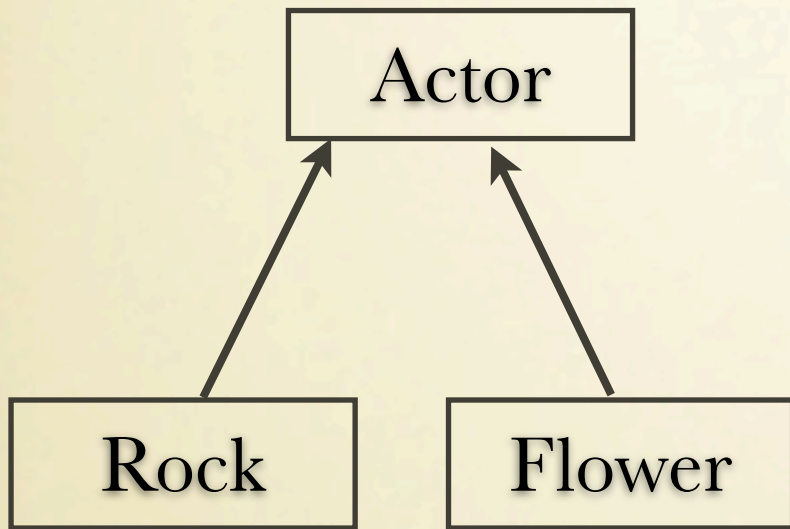
Every Actor in GridWorld is told to do things through the act() method. If you want to change the behavior of an Actor, you must either override the act() method, or a method called by the act() method.

GRIDWORLD



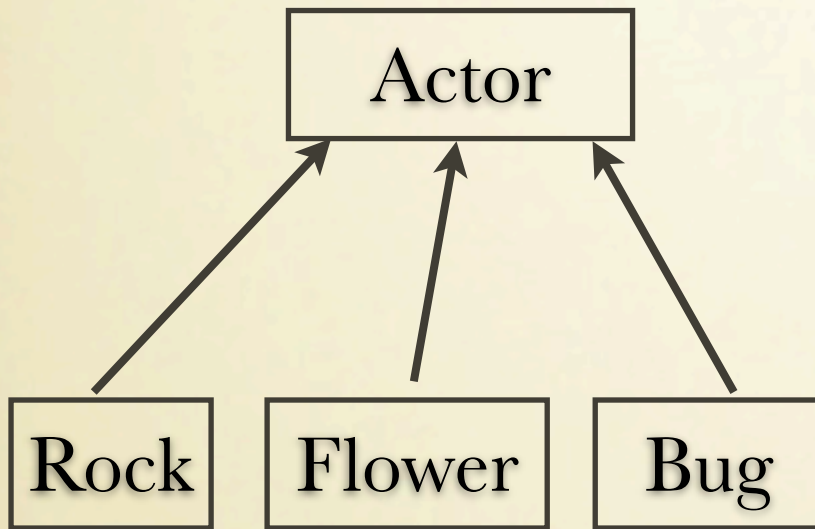
- `act()` is overridden
to do nothing

GRIDWORLD



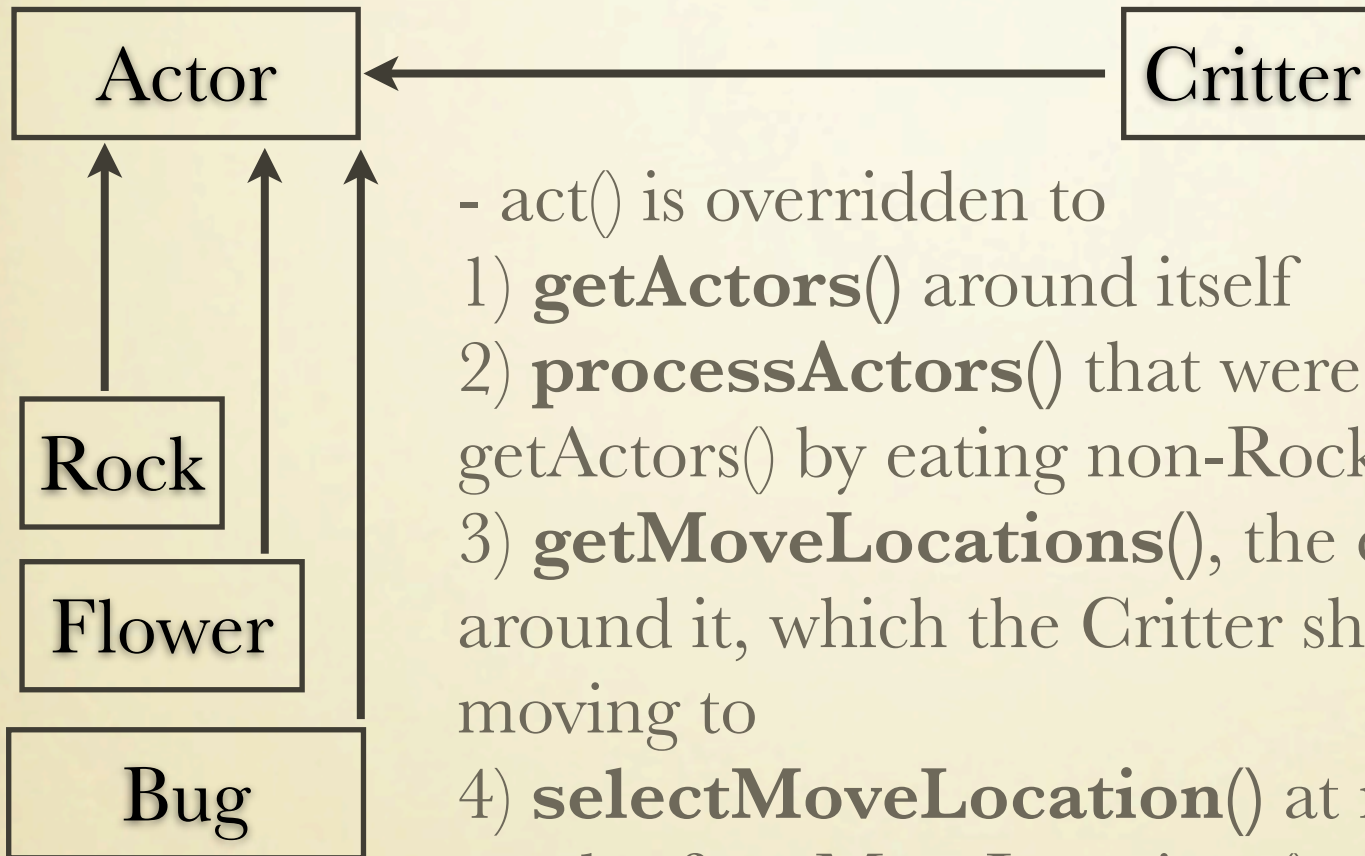
- `act()` is overridden
to darken color

GRIDWORLD



- `act()` is overridden to
 - 1) Check if `canMove()` to spot in front of Bug - if so, it move `()`'s
 - 2) If `canMove()` is false, then it `turn()`'s

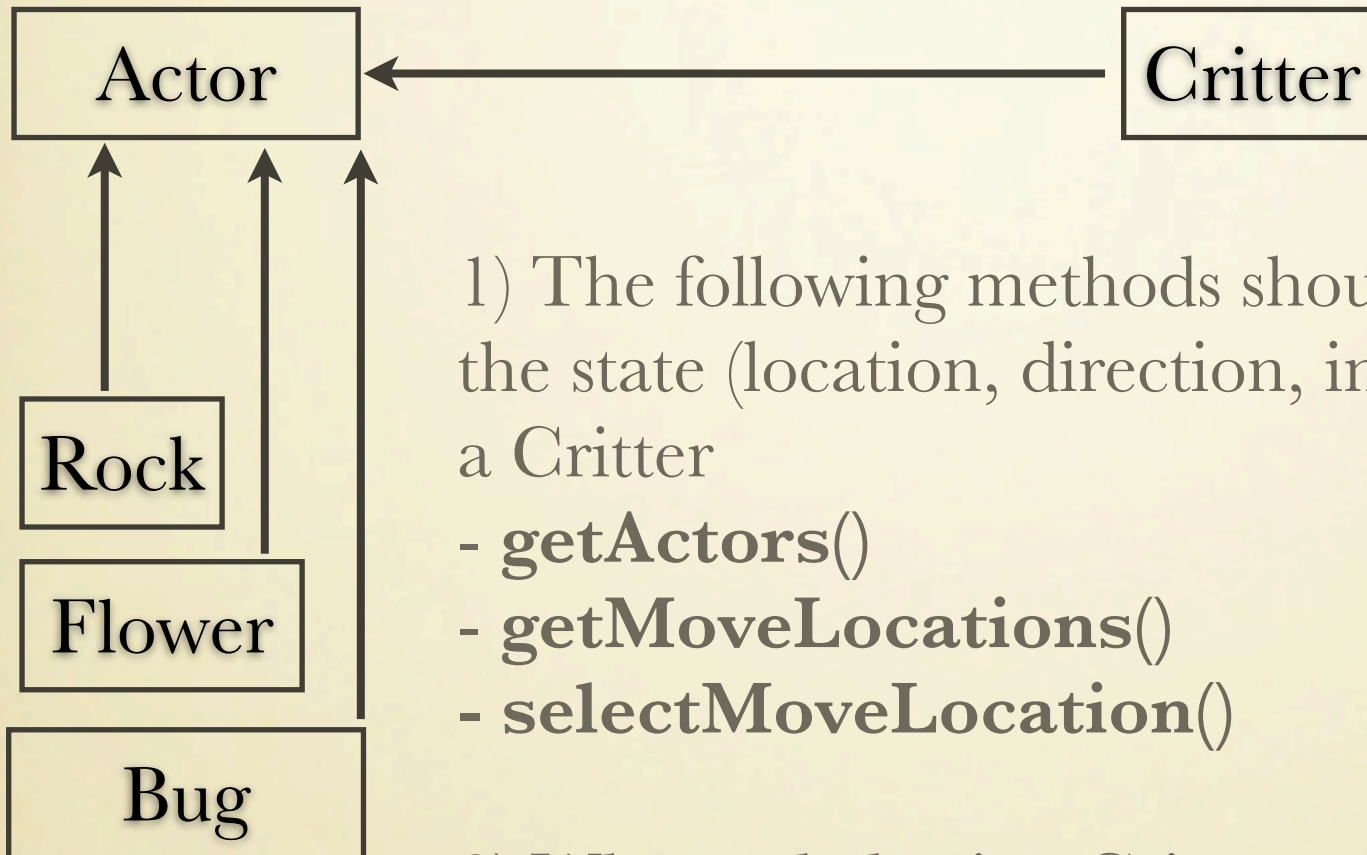
GRIDWORLD



- act() is overridden to

- 1) **getActors()** around itself
- 2) **processActors()** that were the result of calling getActors() by eating non-Rock's and non-Critter's
- 3) **getMoveLocations()**, the empty Location's around it, which the Critter should consider moving to
- 4) **selectMoveLocation()** at random from the result of getMoveLocations()
- 5) **makeMove()** to the Location chosen by selectMoveLocation()

CRITTER CONTINUED



1) The following methods should **NOT** change the state (location, direction, instance fields, etc) of a Critter

- **getActors()**
- **getMoveLocations()**
- **selectMoveLocation()**

2) When subclassing Critter, you should **NOT** override the **act()** method

GRIDWORLD OPERATIONS

- Create a location
 - `Location temp = new Location(2,3);`
- Create a location dependent on another location
 - `Location temp2 = new Location(temp.getRow(), temp.getCol()+1);`
- Create a location that is in a direction from a location
 - `Location temp3 = temp2.getAdjacentLocation(45);`
 - `//45°` is the same as `Location.HALF_RIGHT` (i.e. Northeast)

GRIDWORLD OPERATIONS

- How to check if a location is in a grid

```
Grid<Actor> grid = getGrid();  
if(grid.isValid(temp3)) {  
    //code  
}
```

- How to get what Actor is at a Location in the grid

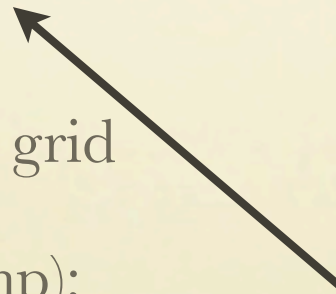
```
Actor a = grid.get(temp3); //null if no such actor
```

GRIDWORLD OPERATIONS

- How to move an actor manually
 - `a.moveTo(temp2);`
- How to remove an Actor from this grid
 - `a.removeSelfFromGrid();`
- How to put an Actor into the grid
 - `a.putSelfInGrid(grid, temp);`

Can use **this** instead of **a** if you are inside an Actor and you want to command it to do something!

After this line of code, `a.getGrid()` and `a.getLocation()` will return null!



GRIDWORLD OPERATIONS

- How to override a method and make it do the old method code sometimes.

```
public void act() {  
    if(...)   
        doSomethingDifferent();  
    else   
        super.act();  
}
```

- How to compare Location's

`temp.equals(temp2)` not `temp == temp2`

compare values

compare memory bubbles

STRING OPERATIONS

String a = "abcdefg";

String b = a.substring(1,3); //bc

Has no effect on **a**

start

stop before

String c = a.substring(4); //efg

start at 4, and go to end

String d = a.substring(4, a.length()); //efg

number of char's in the String

STRING OPERATIONS

```
String a = "abcdefg";
```

```
a.substring(1,2);
```

← Statement has no effect!

```
System.out.println(a.substring(1,2));
```

← prints **b**

```
a = a.substring(0,3) + "!" + a.substring(3); // abc!defg
```

↑
Technique for inserting in a String
Same idea for

- "inserting" in a String
- "modifying" a String

ARRAYLIST OPERATIONS

```
ArrayList<Integer> list = new ArrayList<Integer>();
```

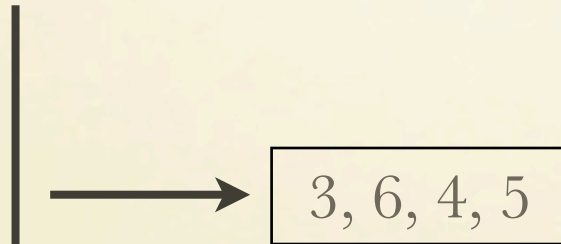
```
list.add(3);
```

```
list.add(4);
```

```
list.add(5);
```

```
list.add(1,6);
```

index value



ARRAYLIST OPERATIONS

list: 3, 6, 4, 5

list.set(2,9); → 3, 6, 9, 5

index value

list.remove(0); → 6, 9, 5

index

int v = list.get(0); //6

index

ARRAYLIST OPERATIONS

Print all values in a list

```
for(int v: list)
    System.out.println(v);
```

number of items in the list

```
for(int i=0; i<list.size(); i++)
    System.out.println(list.get(i));
```

ARRAYLIST OPERATIONS

Remove all evens from a list

```
for(int i=list.size()-1; i>=0; i--)  
    if(list.get(i) % 2 == 0)  
        list.remove(i);
```

Backwards remove is safer!
Forwards remove causes
shifting... so you must be
careful.

Cannot use advanced for-
loop when modifying a list
at the same time.

ARRAY OPERATIONS

Creation

length of array

```
double[] ar = new double[5];
```

Print all values

length of array

```
for(int i=0; i<ar.length; i++)  
    System.out.println(ar[i]);
```

```
for(int v: ar)  
    System.out.println(v);
```

v is a **value**,
not an **index**

ARRAY OPERATIONS

Swap

```
int v = array[1];  
array[1] = array[2];  
array[2] = v;
```

Shifting Left

```
for(int i=0; i<ar.length-1; i++)  
    array[i] = array[i+1];
```

Shifting Right

```
for(int i=array.length-1; i>=0; i--)  
    array[i] = array[i-1];
```

DATASTRUCTURE CHART


	ARRAY	ARRAYLIST	STRING
NUMBER OF ITEMS	<code>.length</code>	<code>.size()</code>	<code>.length()</code>
ACCESS BY INDEX	<code>[i]</code>	<code>.get(i)</code>	<code>.charAt(i)</code>
CHANGE VALUE	<code>[i] = v</code>	<code>.set(i,v)</code>	Need substring() command!
ADD TO END	Need a new array!	<code>.add(v)</code>	<code>+= v</code>
ADD TO BEGINNING	Need a new array!	<code>.add(0,v)</code>	<code>str = v + str;</code>

2D ARRAY

- Creation

```
int[] [] matrix = new int[5][4];
```

rows cols



0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

- Accessing Values

```
matrix[0][0] = 3;  
int x = matrix[2][2];
```

2D ARRAY

- Accessing Dimensions

```
int rows = matrix.length;  
int cols = matrix[0].length;
```

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

- Print all values

```
for(int row = 0; row < matrix.length; row++)  
{  
    for(int col = 0; col < matrix[0].length; col++)  
    {  
        System.out.print(matrix[row][col]);  
    }  
    System.out.println();  
}
```

2D ARRAY FANCINESS

- Print all values

```
for(int[] row: matrix)
{
    for(int val: row)
    {
        System.out.print(val + " ");
    }
    System.out.println();
}
```

Don't have to code
this... maybe read it