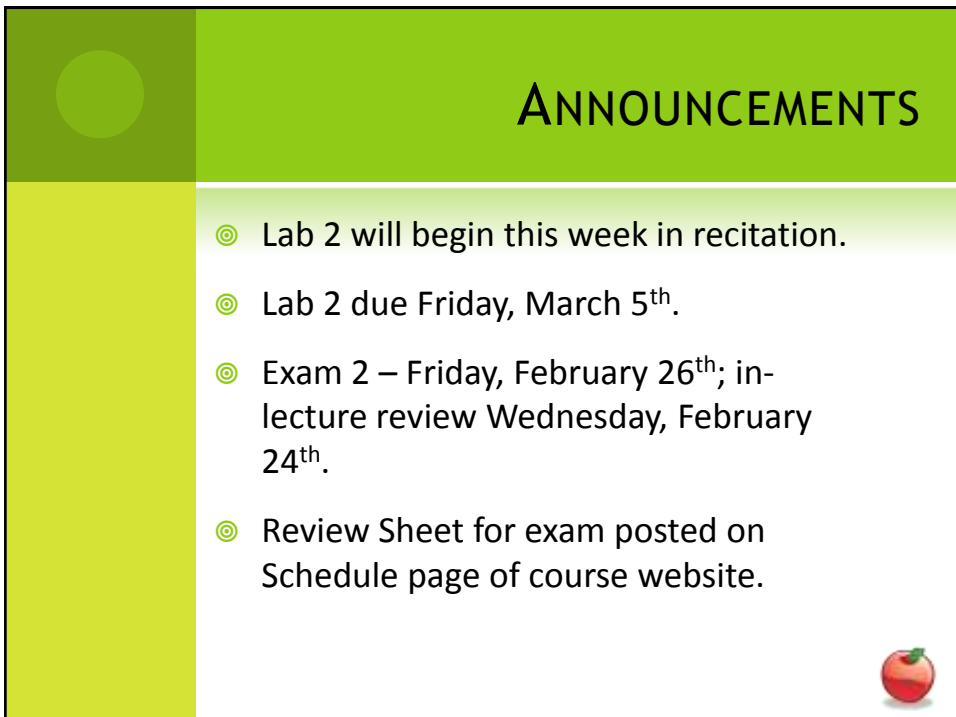



**CSE 113 A**  
February 15-19, 2010



## ANNOUNCEMENTS

- ⦿ Lab 2 will begin this week in recitation.
- ⦿ Lab 2 due Friday, March 5<sup>th</sup>.
- ⦿ Exam 2 – Friday, February 26<sup>th</sup>; in-lecture review Wednesday, February 24<sup>th</sup>.
- ⦿ Review Sheet for exam posted on Schedule page of course website.



3


## MONDAY LECTURE CODE

- Some things to note:
- In checkEdges method: We needed to get the width of the world, so we first needed to get the world and then needed to ask the world for its width.

`getWorld().getWidth()`

returns a world object

ask the world object for its width



4

## MONDAY LECTURE CODE

- In move method: I do not expect you to be able to derive formulas to create movement taking into account rotation. You can simply reuse this code as needed to move based on degree of rotation.
- What is important is that you can move something simply by taking its current location and changing that slightly. Doing this repeatedly moves the actor.



5

## MONDAY LECTURE CODE

- ⦿ When checking edges, we can avoid them by turning or wrap – see example in code



6

## MONDAY LECTURE CODE

- ⦿ Detecting intersecting objects can be done using `getOneIntersectingObject` method.
- ⦿ This method can take as an argument a class that represents the type of object we are looking for (like `canSee` in Crab example).
- ⦿ This method returns an Actor object that represents what the current actor is intersecting with. If there is no intersecting actor, the method returns null. Null is a keyword in Java that represents the value of a null reference (can be thought of as “no object”).



7

## MONDAY LECTURE CODE

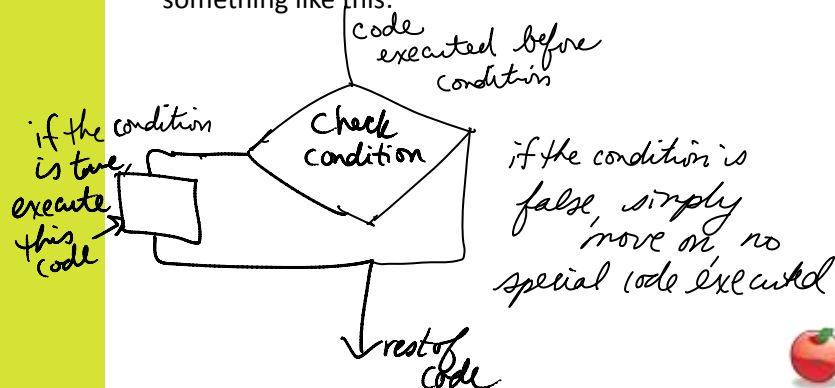
- Change the image of an actor using setImage. Takes as an argument a GreenfootImage which is created with a string that represents the name of the file where the image is stored. Can simply use any of the built-in images, or can add your own to images folder of the scenario and it is accessible within the scenario.



8

## IF-STATEMENTS

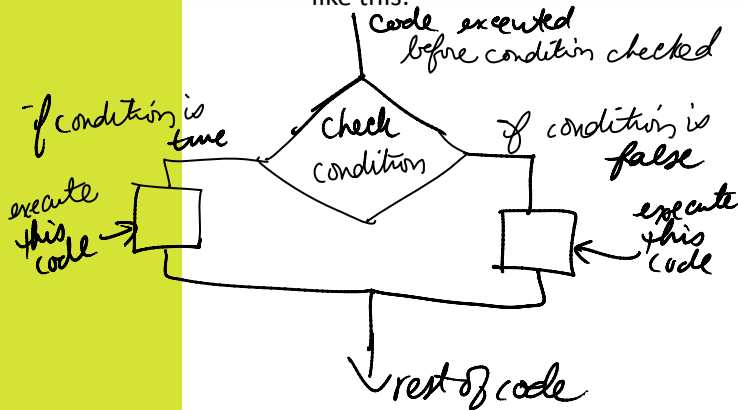
- We have been working a lot with if-statements to determine choices in our programs. If we look at our program execution with if-statements, it would look something like this:



9

## MORE WAYS TO CHOOSE

- ⊙ We could create choice in programs that looks like this:



10

## MORE WAYS TO CHOOSE

- ⊙ That would be the notion of a choice when there is a definitive path when a condition is true and another path when the condition is false.
- ⊙ In order to do this type of choice in code, we would need to use if-else statements instead of just if-statements.



11

## IF-ELSE SYNTAX

```
if( /*boolean expression*/ )  
{  
    //code to be executed if boolean expression is true  
}  
else  
{  
    //code to be executed if boolean expression is false  
}
```



12

## WEDNESDAY LECTURE CODE

- ⊙ Create a method named `checkForCars` in the `Ambulance` class that checks to see if the `Ambulance` intersects with a car.
- ⊙ If the `Ambulance` does intersect with a car, then check to see if that car has hit more than 5 barrels. If the car has hit more than 5 barrels, then the car should be removed from the scenario. Otherwise the car should be turned into a flower.



13

## WEDNESDAY LECTURE CODE

- ⦿ So, in the checkForCars method, we first wrote the code to getOneIntersectingObject of type Car and stop the scenario when it happens. This code is a copy/edit of the code we used to determine if the ambulance was intersecting with a barrel.
- ⦿ Then, we removed the line that stopped the scenario to replace it with the code we want to happen when an ambulance and a car collide.



14

## WEDNESDAY LECTURE CODE

```
if(/*car has hit more than 5 barrels*/)
{
    //remove car from world
}
else
{
    //turn car into flower
}
```



15

## WEDNESDAY LECTURE CODE

- ⦿ We can use the code we had before for turning a barrel into a flower to turn a car into a flower (copy/paste).
- ⦿ We know about a method to add an object to the world. There is a similar method to remove an object. We need to make sure that we get the world first and then remove the object:

```
getWorld().removeObject(car);
```



16

## WEDNESDAY LECTURE CODE

- ⦿ Now we need to figure out how many barrels the car has hit.
- ⦿ We need to create a method inside the Car class that will report on how many barrels a car has hit. Recall that cars are already keeping track of how many barrels they hit in an instance variable. The method we write simply reports the value of that variable.
- ⦿ So, in the if-statement we can call that method after we write it:

```
if(car.getBarrelsHit() > 5)
```





17

## WEDNESDAY LECTURE CODE

- ⦿ In order to call the new method on the car, we needed to make one change to the way we treat the “actor” that is returned from the call to `getOneIntersectingObject(Car.class)`.

- ⦿ Originally the code looked like this:

```
Actor car = getOneIntersectingObject(Car.class)
```

- ⦿ Now it looks like this:

```
Car car = (Car)getOneIntersectingObject(Car.class)
```



18

## WEDNESDAY LECTURE CODE

- ⦿ The `(Car)` is a typecast. We are taking the Actor that is returned and telling Java to treat it as though it were a Car (which it is – we asked for intersecting objects of type Car after all).



19

## IF-ELSE IF STATEMENTS

- ⊙ We have to make a change to the checkForEdges code from last time.
- ⊙ We are going to create an if-else if structure in the top/bottom and left/right edge checks.

```

if(/*actor at right*/) { /* do something */}
else if(/*actor at left*/) { /* do something */}
}

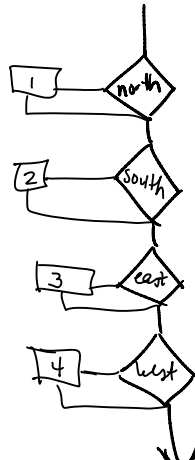
```



20

## WEDNESDAY CODE

- ⊙ Originally, we had all the edges as if's. This created a picture like this:



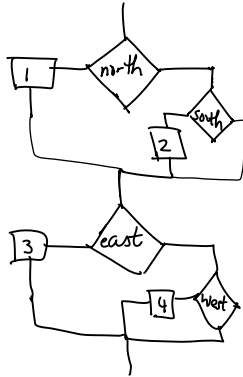
*You could execute 1 & 2  
or 3 & 4 but the  
actor can never be both  
at north & south edge or  
at the east & west edge  
at the same time.*



21

## WEDNESDAY CODE

- ⊙ When we put the if-else ifs in, we have this picture:



*so only one of 1 or 2  
can be executed at any  
given time.*

*same thing with 3 or 4.*



22

## FRIDAY LECTURE CODE

- ⊙ Write a method so that when the ambulance reaches a certain point on the screen (let's say 137), all of the barrels are removed from the world.
- ⊙ How would we write the code for this?
  - ⊙ First, we can notice that there is a condition that must be met, so we need an if-statement



23

## FRIDAY LECTURE CODE

```
if( )  
{  
}
```

- ⊙ We need to determine what goes into the () and what goes into the { }
- ⊙ Tip: Write them out in English first and then translate into Java code.



24

## FRIDAY LECTURE CODE

- ⊙ The condition is looking for when our ambulance's x-coordinate is 137.
- ⊙ The code we execute removes all barrels from the world.
- ⊙ Tip: Be sure to refer to the documentation for the World and Actor classes when we are trying to do something new – there may be methods defined that can help us. (This was the case with removing the barrels from the world).



25

## EXERCISE (WILL BE ANSWERED ON MONDAY)

- ⦿ Make the ambulance add 5 flowers to the screen when the ambulance is at  $y = 36$ .

