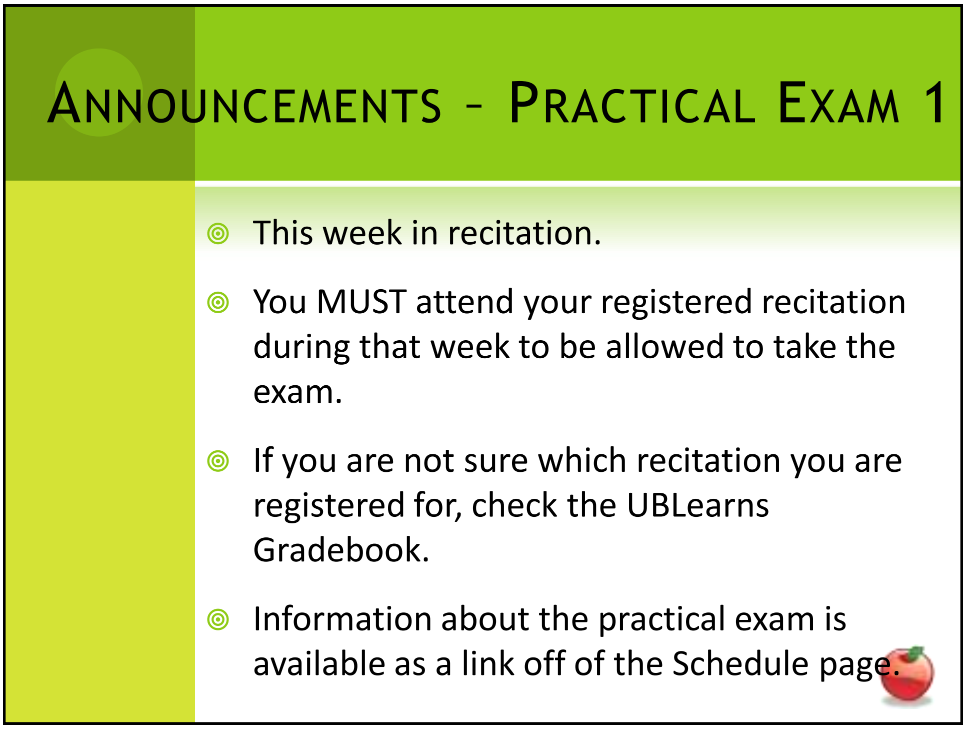
CSE 113 B

February 21-25, 2011

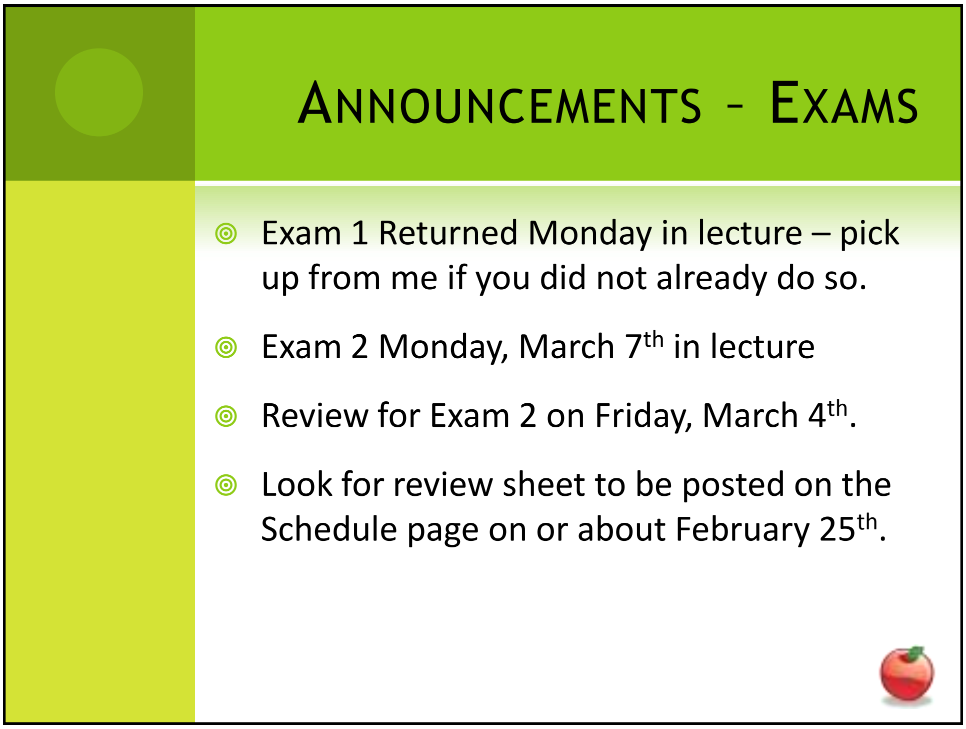
Announcements - Lab

- Lab 1, 2, 3, 4; Practice Assignment 1, 2, 3, 4 grades are available in Web-CAT – look under “Results” -> “Past Results” and if looking for Lab 1, make sure to check “closed”.
- Lab 5 & Practice Assignment 5 will be graded by Web-CAT, but the grading is not functional at this time.



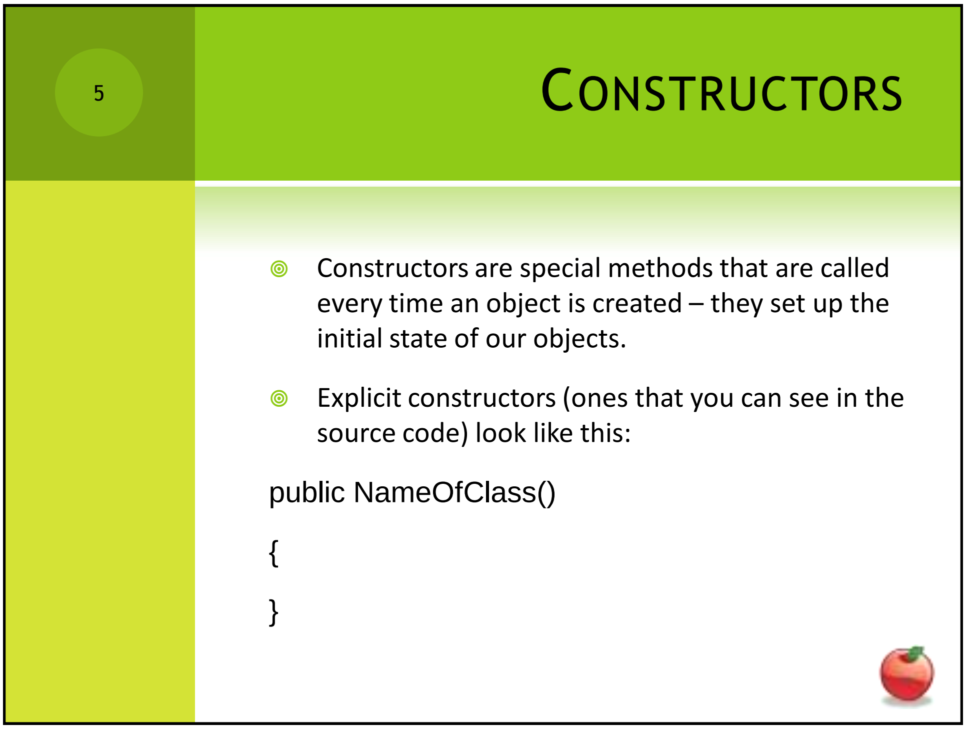
Announcements – Practical Exam 1

- This week in recitation.
- You MUST attend your registered recitation during that week to be allowed to take the exam.
- If you are not sure which recitation you are registered for, check the UBLearns Gradebook.
- Information about the practical exam is available as a link off of the Schedule page.



Announcements – Exams

- Exam 1 Returned Monday in lecture – pick up from me if you did not already do so.
- Exam 2 Monday, March 7th in lecture
- Review for Exam 2 on Friday, March 4th.
- Look for review sheet to be posted on the Schedule page on or about February 25th.

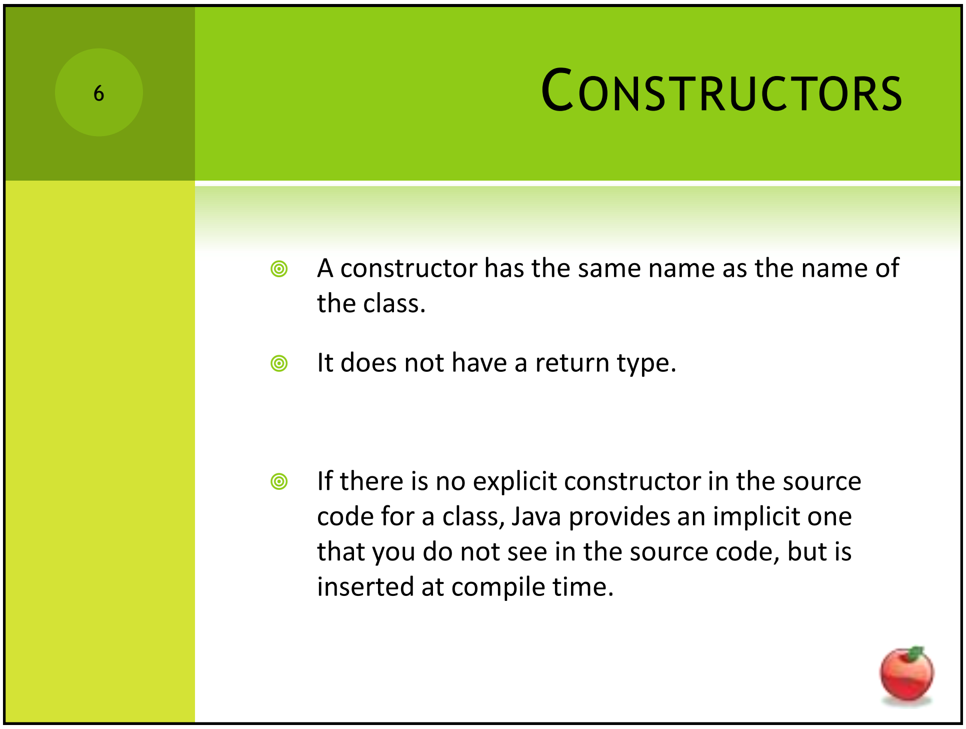


5

Constructors

- Constructors are special methods that are called every time an object is created – they set up the initial state of our objects.
- Explicit constructors (ones that you can see in the source code) look like this:

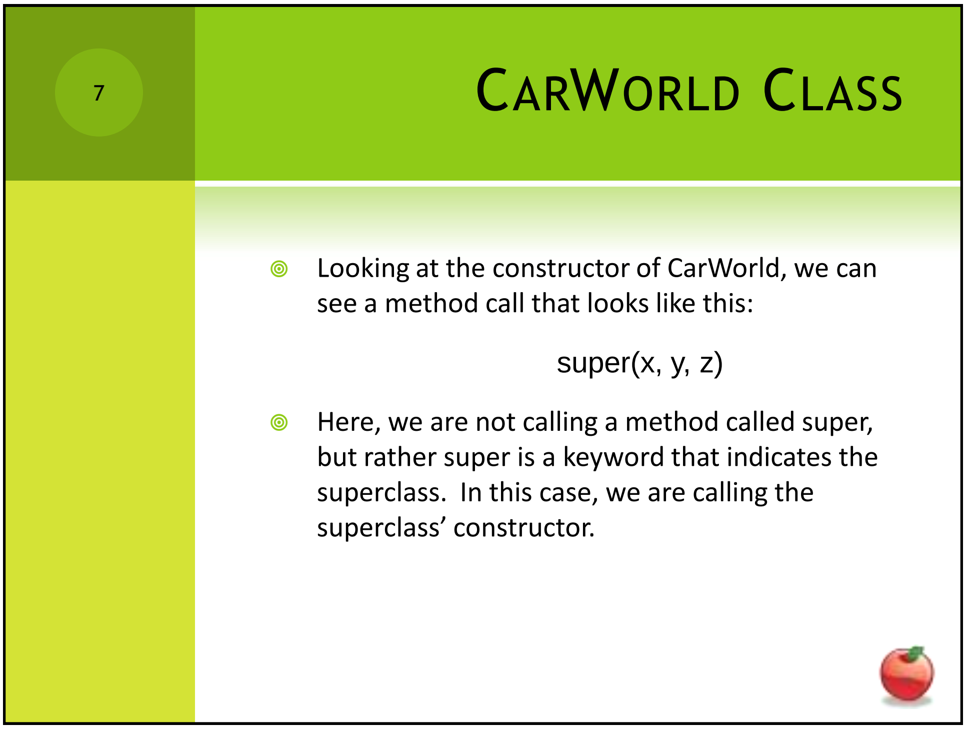
```
public NameOfClass()
{
}
```



6

Constructors

- A constructor has the same name as the name of the class.
- It does not have a return type.
- If there is no explicit constructor in the source code for a class, Java provides an implicit one that you do not see in the source code, but is inserted at compile time.



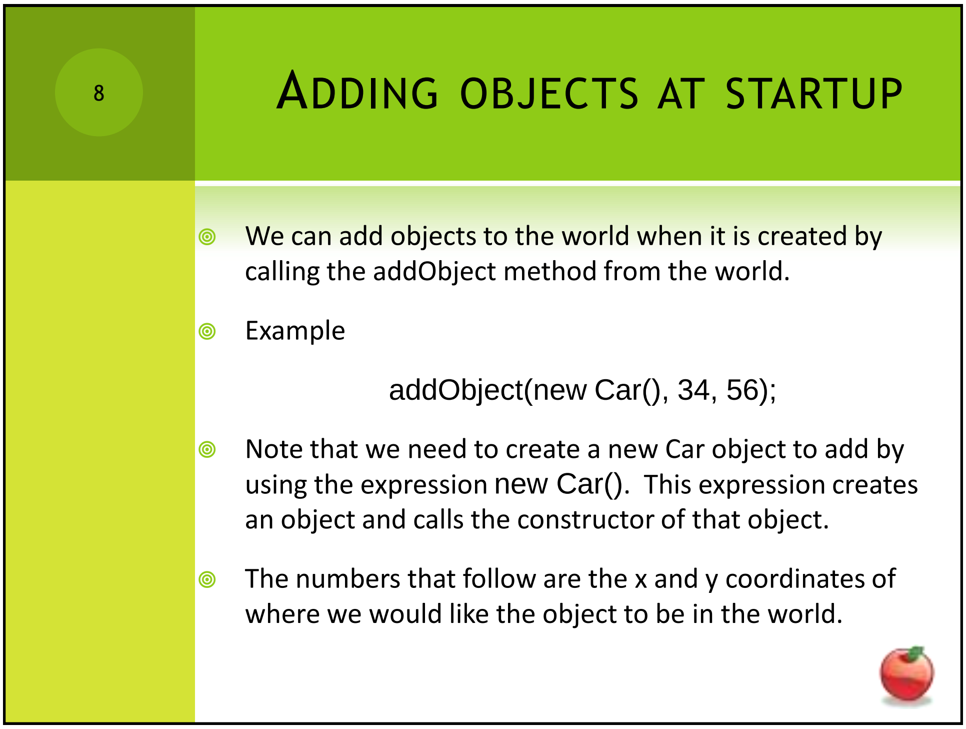
7

CarWorld Class

- Looking at the constructor of CarWorld, we can see a method call that looks like this:

```
super(x, y, z)
```

- Here, we are not calling a method called super, but rather super is a keyword that indicates the superclass. In this case, we are calling the superclass' constructor.



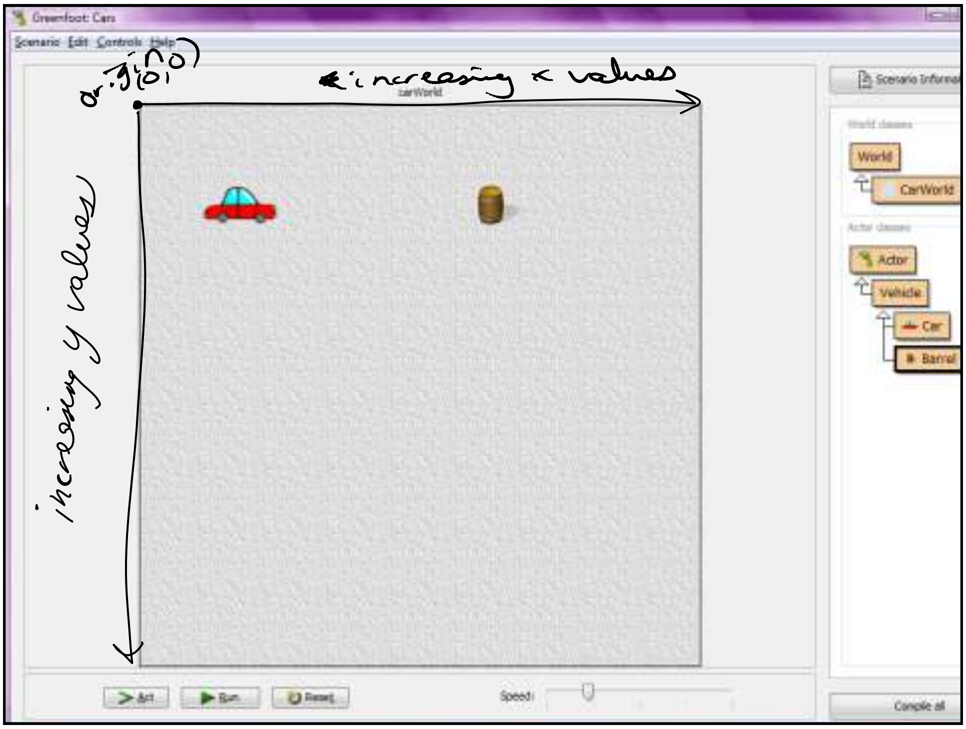
8

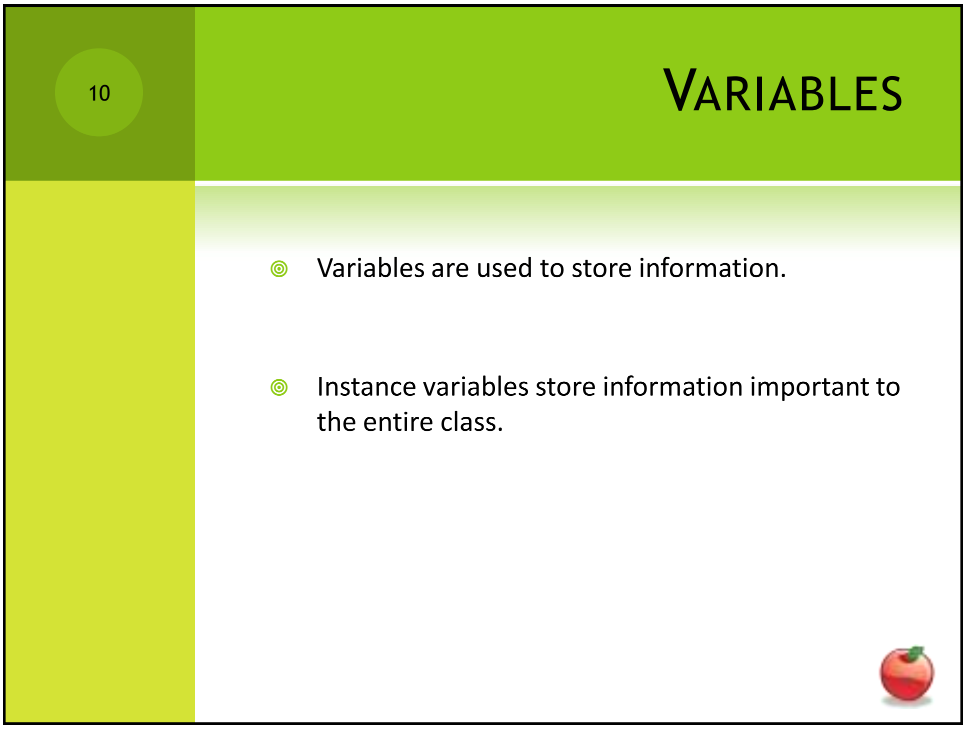
Adding objects at startup

- We can add objects to the world when it is created by calling the addObject method from the world.
- Example

```
addObject(new Car(), 34, 56);
```

- Note that we need to create a new Car object to add by using the expression new Car(). This expression creates an object and calls the constructor of that object.
- The numbers that follow are the x and y coordinates of where we would like the object to be in the world.

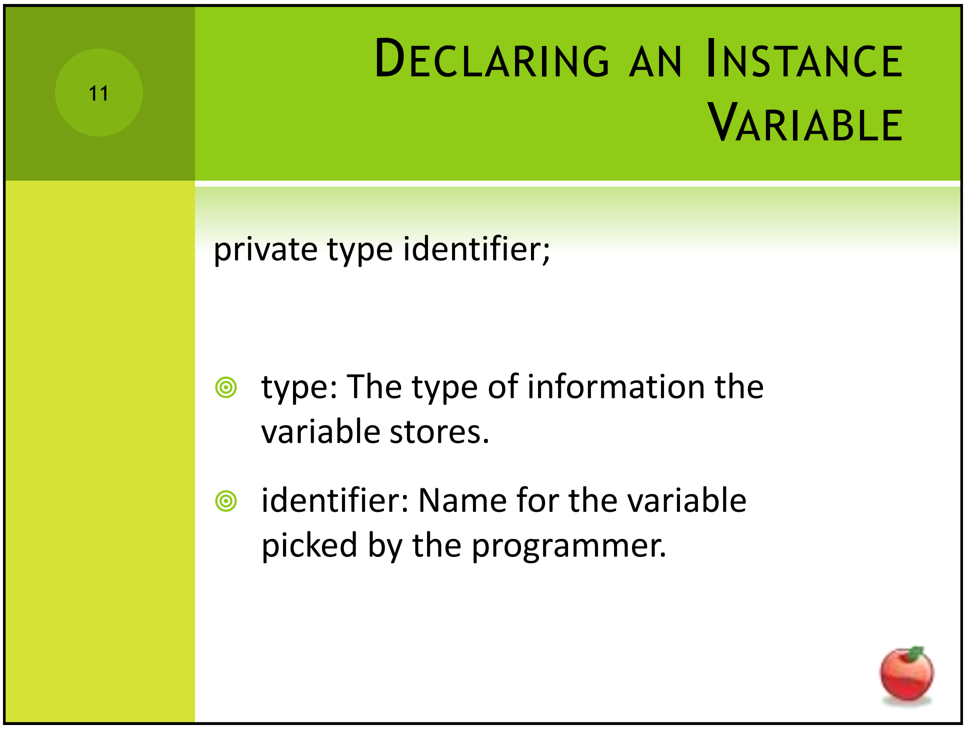




10

VARIABLES

- Variables are used to store information.
- Instance variables store information important to the entire class.

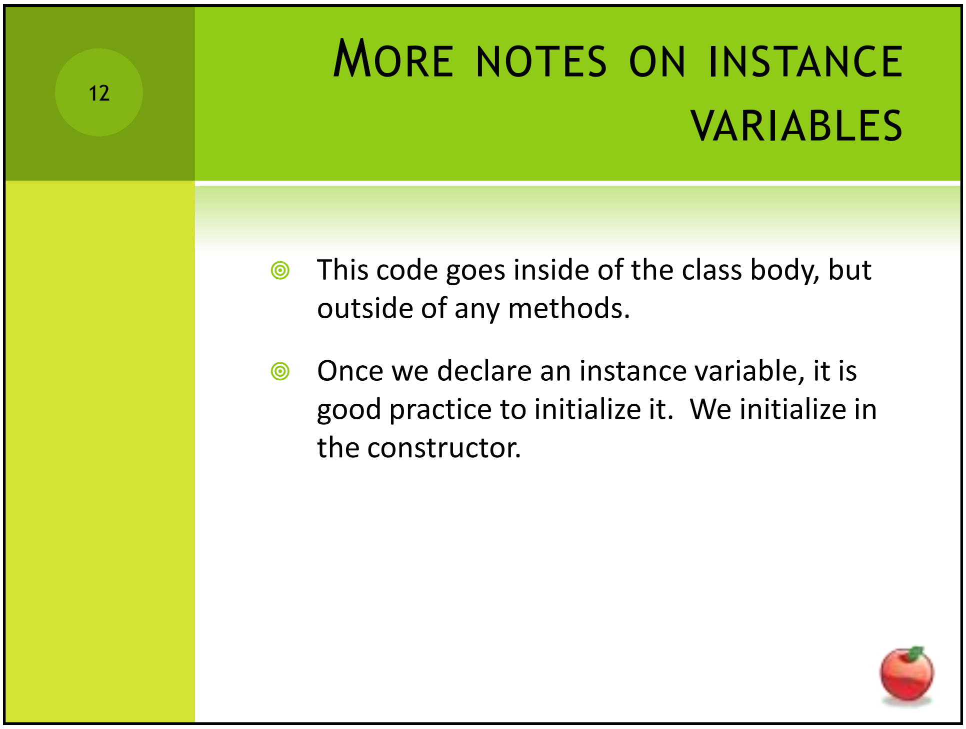


11

Declaring an Instance Variable

```
private type identifier;
```

- type: The type of information the variable stores.
- identifier: Name for the variable picked by the programmer.



12

More notes on instance variables

- This code goes inside of the class body, but outside of any methods.
- Once we declare an instance variable, it is good practice to initialize it. We initialize in the constructor.

13

Assignment

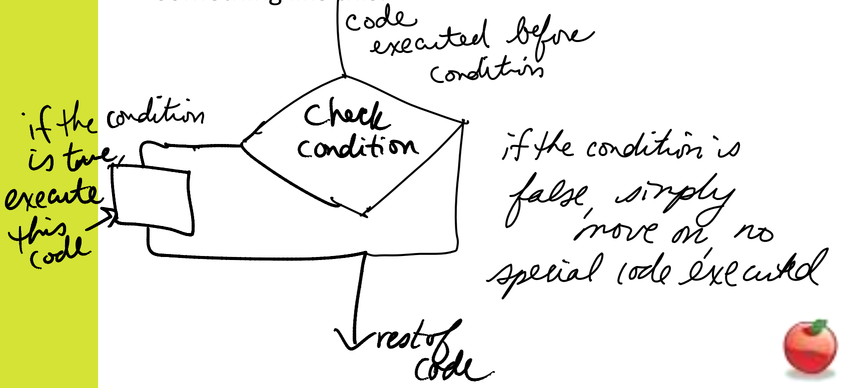
variableName = expression;

- The expression on the right is evaluated first and then the result is stored in the variable named on the left.

14

If-statements

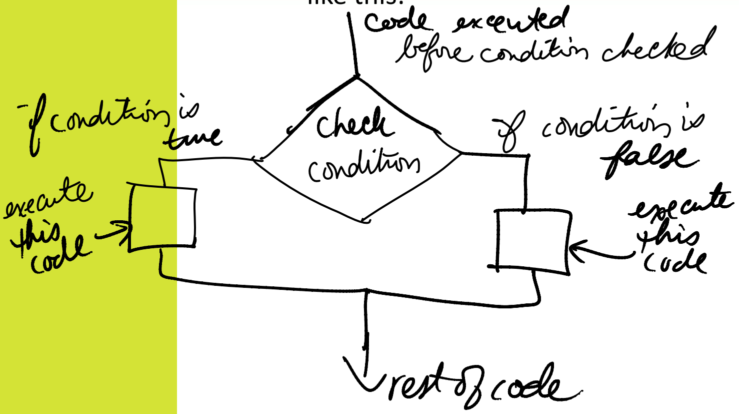
- We have been working a lot with if-statements to determine choices in our programs. If we look at our program execution with if-statements, it would look something like this:


code executed before condition
Check condition
if the condition is true, execute this code
if the condition is false, simply move on, no special code executed
rest of code

15

More ways to choose

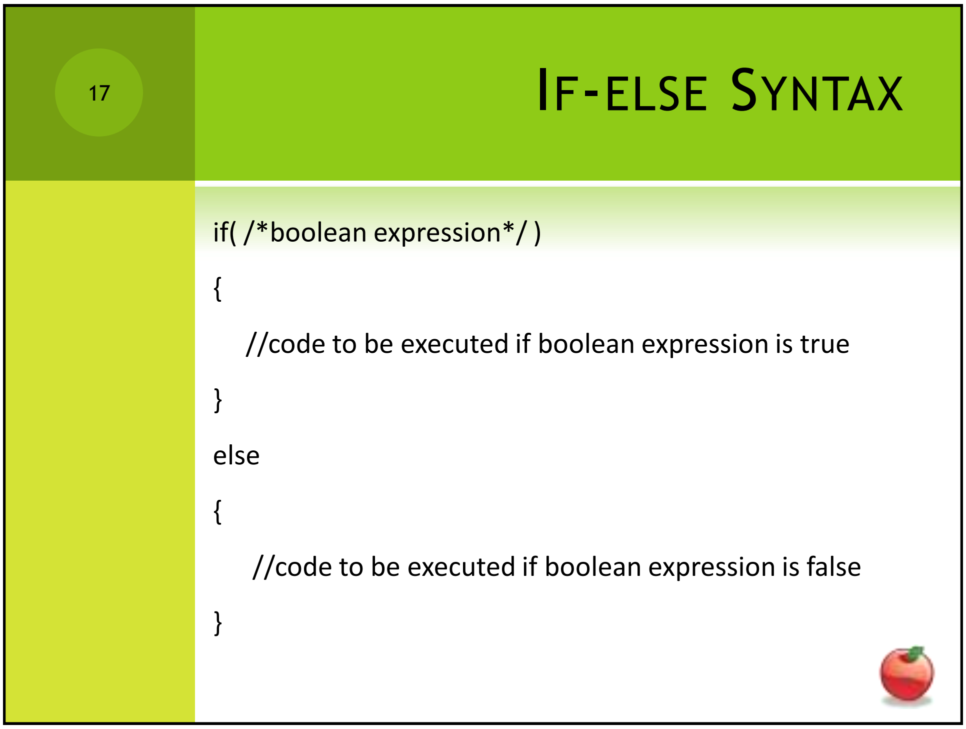
- We could create choice in programs that looks like this:



16

More ways to choose

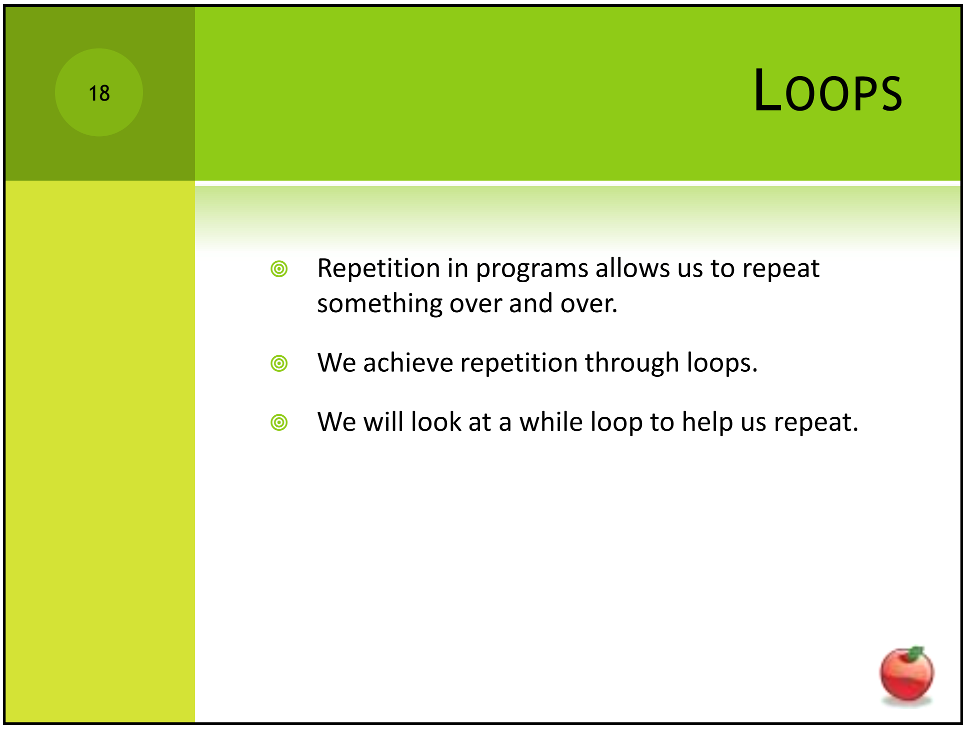
- That would be the notion of a choice when there is a definitive path when a condition is true and another path when the condition is false.
- In order to do this type of choice in code, we would need to use if-else statements instead of just if-statements.



17

If-else Syntax

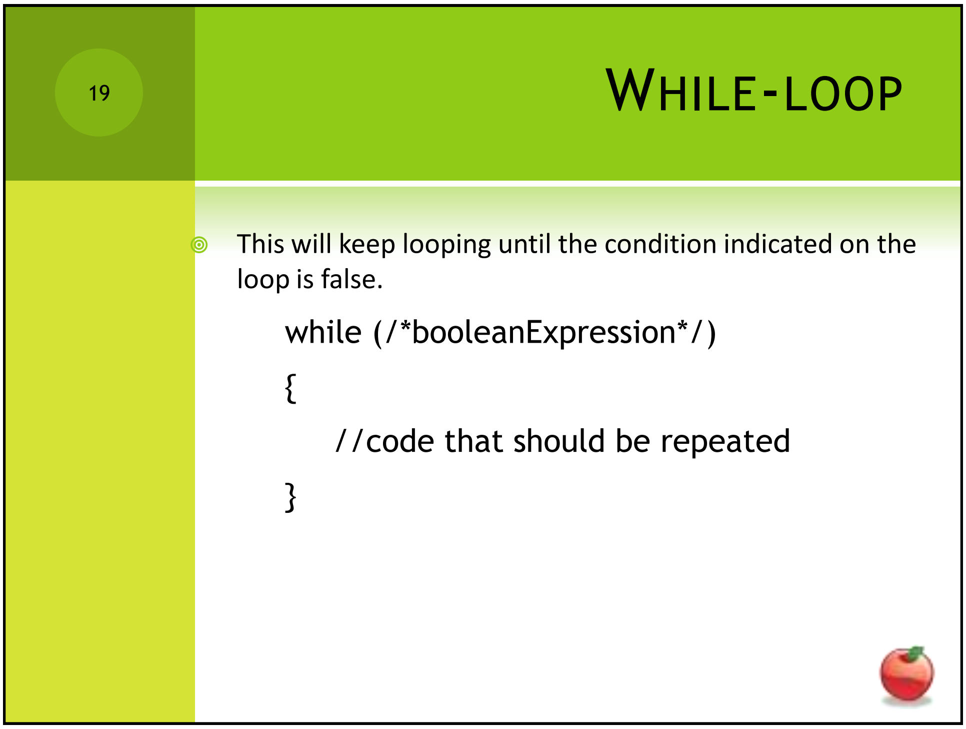
```
if( /*boolean expression*/ )
{
    //code to be executed if boolean expression is true
}
else
{
     //code to be executed if boolean expression is false
}
```



18

Loops

- Repetition in programs allows us to repeat something over and over.
- We achieve repetition through loops.
- We will look at a while loop to help us repeat.

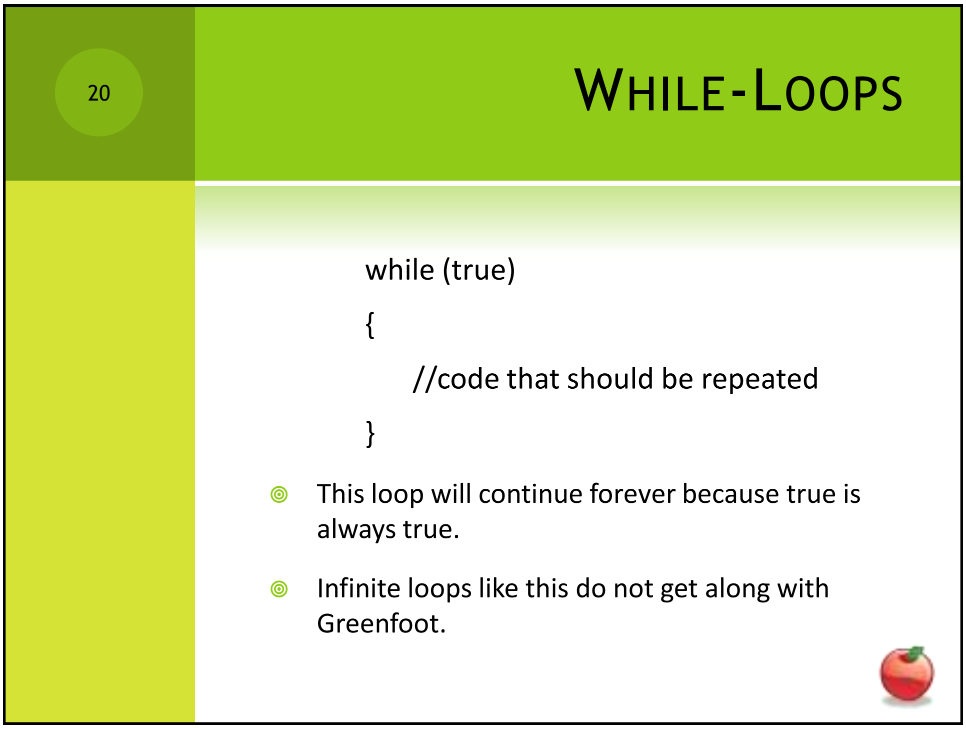


19

While-loop

- This will keep looping until the condition indicated on the loop is false.

```
while (/*booleanExpression*/)
{
     //code that should be repeated
}
```

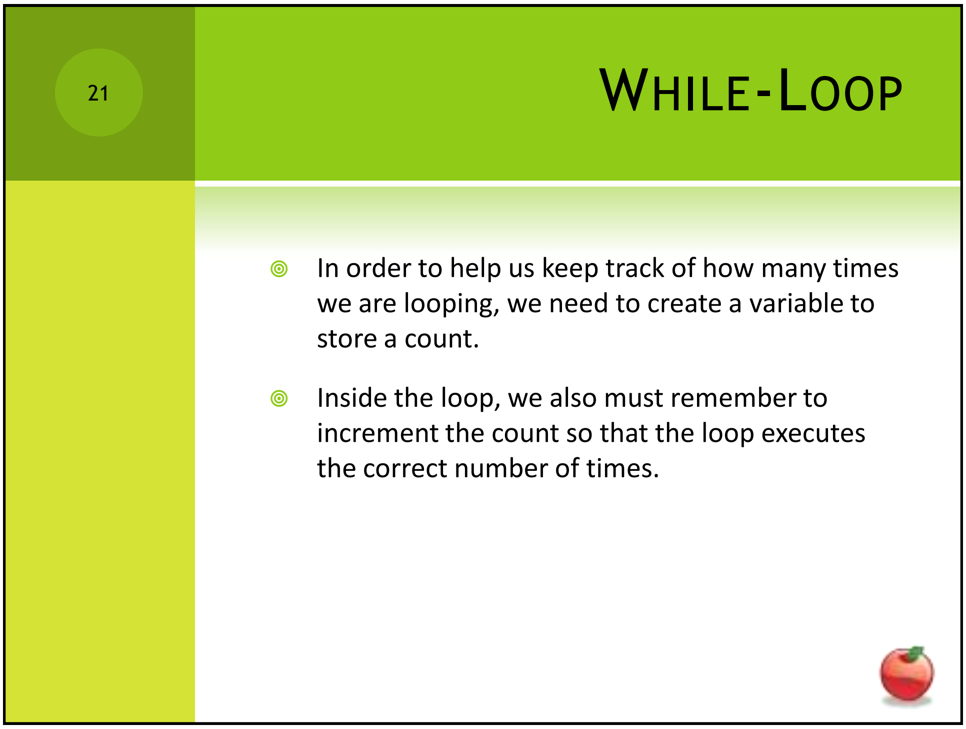


20

While-Loops

```
while (true)
{
     //code that should be repeated
}
```

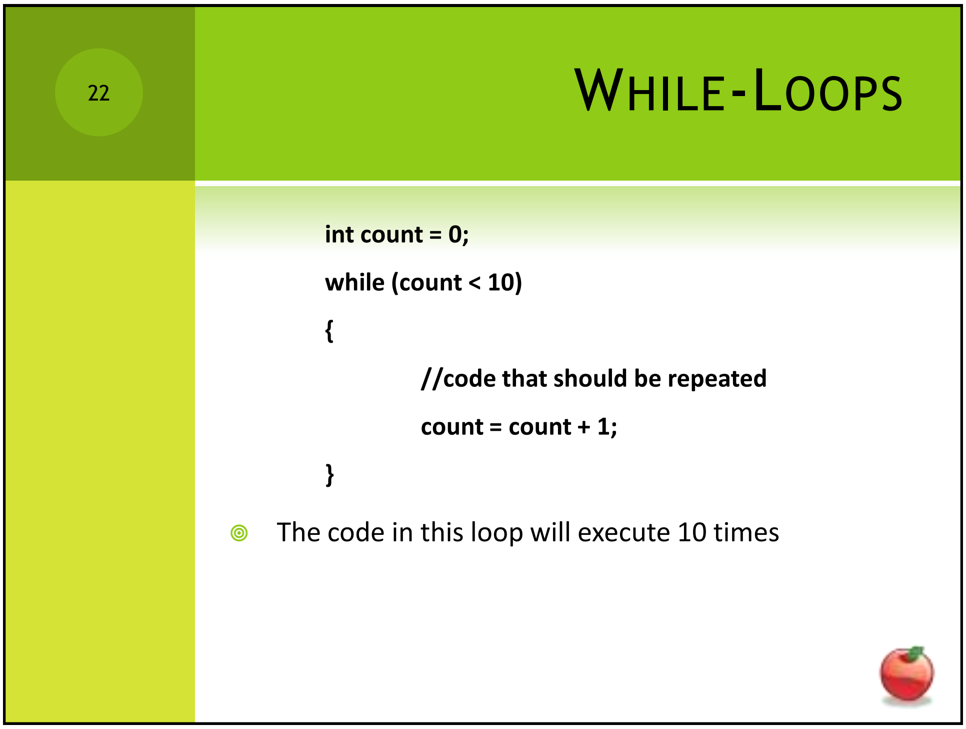
- This loop will continue forever because true is always true.
- Infinite loops like this do not get along with Greenfoot.



21

WHILE-LOOP

- In order to help us keep track of how many times we are looping, we need to create a variable to store a count.
- Inside the loop, we also must remember to increment the count so that the loop executes the correct number of times.



22

WHILE-LOOPS

```
int count = 0;
while (count < 10)
{
        //code that should be repeated
        count = count + 1;
}
```

- The code in this loop will execute 10 times