# CSE 113 A

September 7 – 11, 2009

# Announcements

- No classes Monday 9/7
- First labs meet this week
- Lab 1 posted on course website – due 10/2
- If you are having trouble logging into the computers in the lab or the Web-CAT website that we will be using for submission, please email me.
- Turn in signed last page of syllabus by 9/14.

# Greenfoot Environment

- Main parts
  - World
    - The world that the actors will interact within
  - Execution Controls
    - Controls the execution start/stop and speed of the simulation
  - Class Diagram
    - Shows us the component parts of the scenario

## Class Diagram Panel

- Each box represents a class.
- These classes make up the building blocks of what is going on in our simulations.
- Classes are definitions of the things that will be in our scenario when it is running.

## Objects

- If we want something interesting to happen when we run our scenario, we need to make actual objects (instances of our classes).

- How do you add an object to a scenario in Greenfoot?

## Interacting with Objects

- Once the object is in the scenario, if we right click, we get a listing of all of the actions it can perform.
- These actions are formally called methods inside our programs – they are specified in every detail inside the class definitions (Java source code).
- If we click on one of the menu items, the associated action will happen inside the world. This process is called "calling" or "invoking" a method.

## Method listing

- Each method that is listed has three parts ("words" if you'd like)
- The first word represents the method's return type – the type of information that will be returned after the method is executed
- The second word is the name of the method
- The third word the () is called the parameter list and tells us what type of information we need to pass in to have the method perform its action

7

## Return Type

- The type of information that is returned when the method is finished executing.
- If we see the word void, that means that nothing is being returned from that method.
- If we see the word int, that means an int is being returned. An int is a whole number.
- If we see the word boolean, that means a boolean is being returned. A boolean is a true/false value.

8

## Method names

- Method names are chosen by the programmer when he/she is defining the class in their source code.
- It is a good idea to give methods names that describe their functionality.

9

## Parameter List

- Enclosed in () always.
- If () are empty, then there are no parameters (no inputs) to the method.
- If there is something in the (), then it is telling us what type of information we need to pass into the method in order for the method to function properly. We could pass in an int (whole number), boolean (true/false value) and other types of things that we will learn about this semester.
- Notice that void is never a parameter type – it is only used as a return type.

10

## Calling a method with parameters

- When we call a method that needs parameters (input), we need to provide the actual value of the input in the method call. We would put that value in the dialog box that Greenfoot gives us when we select to invoke that method.

11

## Inheritance

- Looking back at the class diagram panel, we notice that in between some of the class boxes, there are arrows.
- These arrows indicate a relationship between the classes.
- A special relationship called inheritance.
- All classes in our Greenfoot scenarios use inheritance.
- Note that many of the classes have arrows back to Actor, and at least one class has an arrow back to World

12

## Inheritance

- When we see these arrows, the class at the top of the arrow is called the superclass, and the clas on the bottom is called the subclass.

- Inheritance is a very powerful feature of Java and several other languages, but for now, we need to focus on the following facts:

  - All classes in our scenarios will inherit from either Actor (most of our classes) or World (maximum 1 class)

  - When we use inheritance, the subclass inherits and gets to use all of the methods from the superclass

13