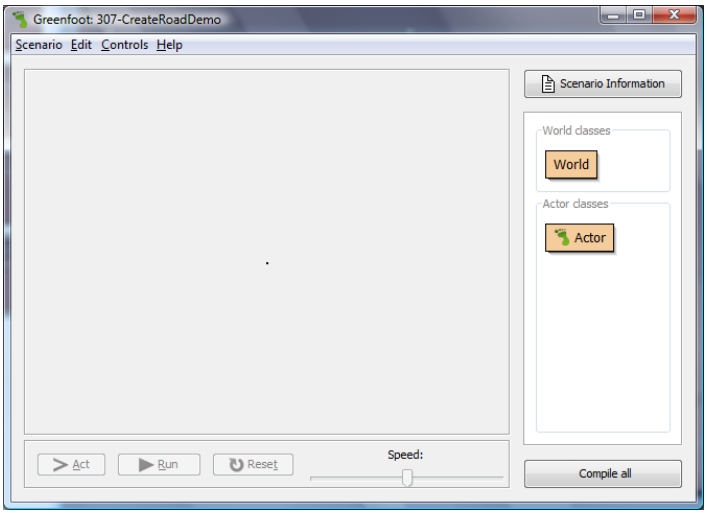
Greenfoot

Greenfoot and Inheritance

Find a "road" image to use as the background of your Greenfoot scenario. Save it as a JPEG. Or create your own image using Photoshop or GIMP.

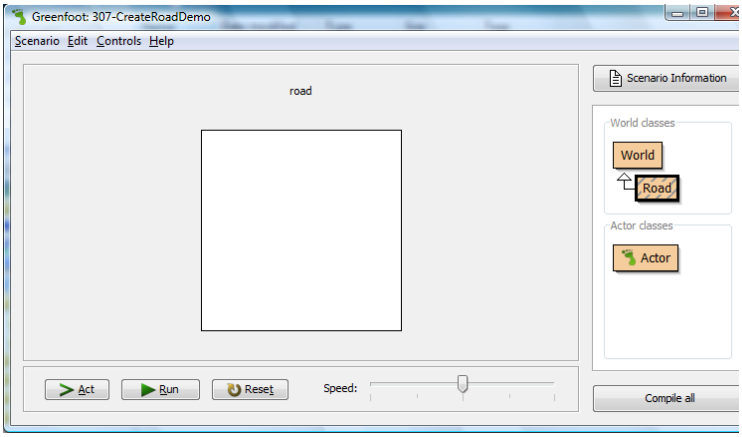
Greenfoot (Background)

1. Create a New Scenario in Greenfoot.



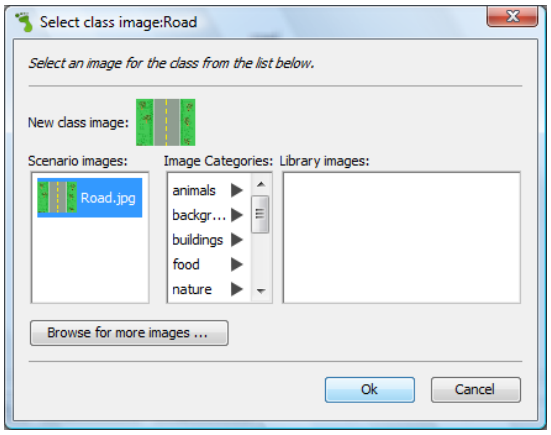
2. Create Your World Class

Right-Click on World
Select "New Subclass"
Give your subclass a name.
Click "OK"



3. Make the background image file available.

Open the folder for you Greenfoot scenario.
Find the "images" folder.
Copy your JPEG image into that "images" folder
This folder is where you will save all of your images

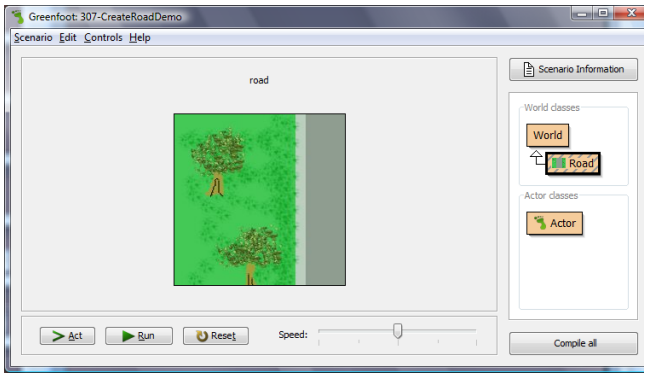


4. Set the image in Greenfoot:

Go back into Greenfoot
Right-click on your World sub-class that you created.
Select "Set image"
Select your image file and click "Okay"

5. Your image should appear as the background for the World!

But notice how the size is not correct. You have to fix that!



6. Double-click the World sub-class that you created to edit it.

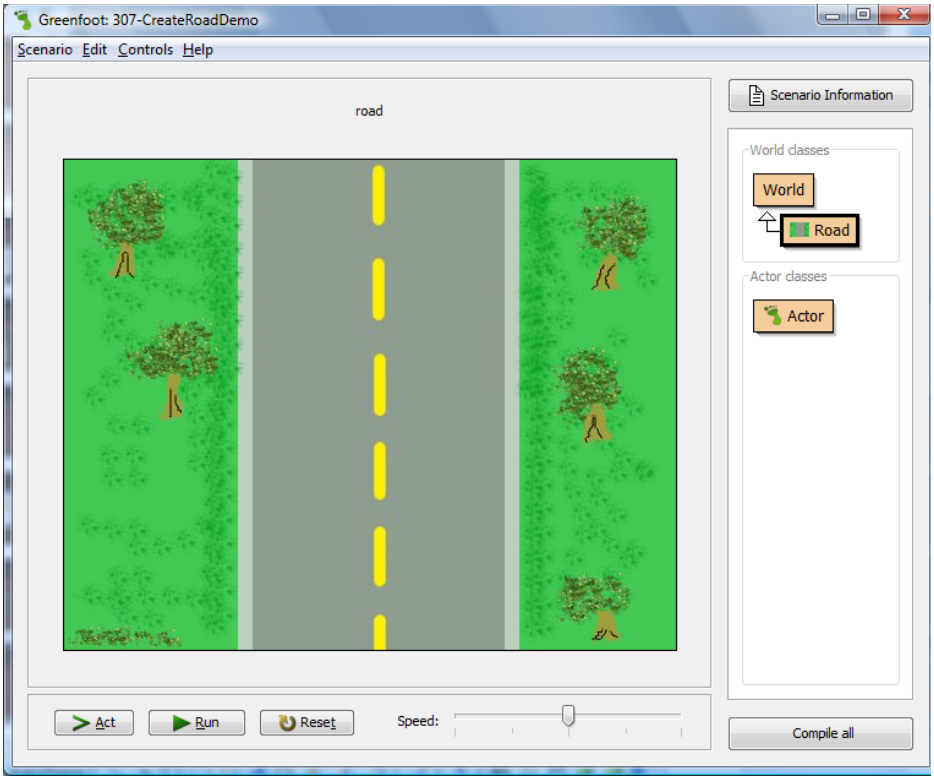
The default World size in the constructor is:

super(20, 20, 10);

Change the settings so that they match the width and height of your image (in pixels). For example, if your world is 500 pixels wide by 400 pixels high, you would set the size to:

super(500, 400, 1);// 1 is for the height and width of one cell in Greenfoot.

7. You are finished setting the background!



Greenfoot and Inheritance

- Inheritance:
  - Allows a class to extend an existing class.
  - This means that a class can use the members (variables and methods) of another class.

- **Inheritance involves the following:**
  - Superclass:
    - The general class.
    - Ex: Mover class in Greenfoot.
  - Subclass
    - The specialized class that uses the general class.
    - Ex: Lizard class in our Greenfoot.

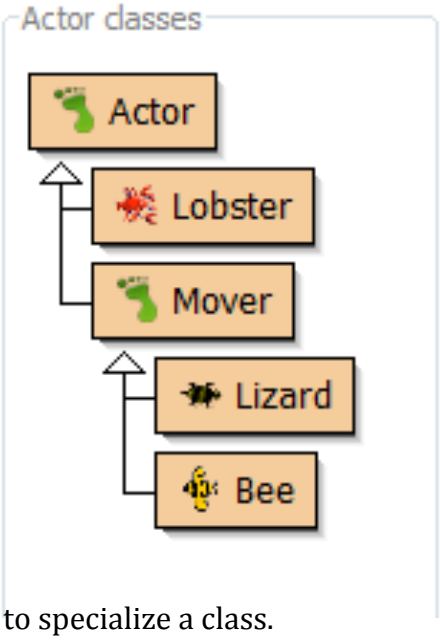
Super and Subclasses

The Actor class is the superclass of the Mover class and Lobster class

The Mover class is the superclass of the Lizard class and the Bee class.

The Mover class is a subclass of the Actor class.
It inherits all of the variables and methods from the Actor class**.**

The Lizard and Bee classes are "subclasses" of the Mover class.
The inherit the Mover class' move( ) and turn( ) methods!



Advantages of using inheritance:

- You don't have to create an entirely new class if you need to specialize a class.
- Inheritance gives you access to the superclass' methods and variables.
- Work smart, not hard!

Inheritance does not work in reverse.

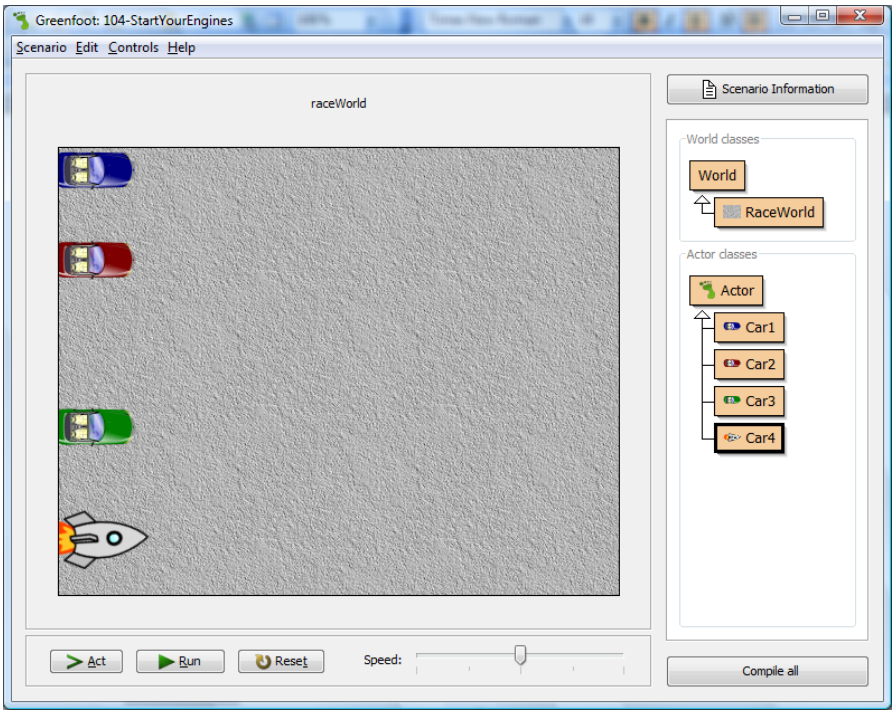
The Mover class will not have access to the Lizard class' methods and variables.

- How do we create inheritance in the code for a subclass?
  - Use the "extends" declaration.
  - For example, if you want the Lizard class to inherit the Mover class you modify the Lizard's class declaration:

```
public class Lizard extends Mover
{
}
```

- **Terminology:**
  - Superclasses → Also known as "base" classes
  - Subclasses -> Also known as "derived" classes

Place Your Bets! (Greenfoot)



Start Your Engines….

Create a Greenfoot Scenario and name it the following:

nastasi_Bets__(LastNameFirstName)

For example, if your name is John Smith from AP Java:

nastasi_Bets_SmithJ

Introduction:

Yeah!!! It's time to create a gambling program!! This program will feature four racing vehicles that start from the same place - but only one will reach the finish line first. How is this possible? We will be using the random class!

Instructions:

1. Create a World subclass and name it RaceWorld. Use the background as shown in the picture above.

2. Edit your RaceWorld class so that the animation will not be choppy. The size and dimension of the RaceWorld should be similar to the picture above.

3. Create 4 Car Classes using the pictures that you see in the above Greenfoot World.

4. Edit the RaceWorld class so that the Car objects will be created on the left side of the world (just like the picture) when it is initialized.

5.Edit the code for the Car classes so that the cars will move forward when the Run button is clicked. In your code you will use the Random Class to create a Random object that makes the forward distance of

each car random (each time the Act ( ) method is called). The distance moved forward should be between 1 and 5.

Here is a refresher:

```
import java.util.Random;// This is how you import the Random class

//Create object to generate random numbers
Random randomNumber = new Random();

randomNumber.nextInt(100) //This will return a number between 1 and 100.
```

Program Requirements:

-There are 4 different Car classes represented by different icons.
-Car objects are created when the RaceWorld is initialized.
-Each Car class uses the Random class to vary the distance at which the car moves forward.
-There are comments that describe what the code is doing.

Saving Your File:

-Save this project to your Flash Drive.

Rubric:

5 points:
-There are 4 different Car classes represented by different icons.
-Car objects are created when the RaceWorld is initialized.
-Each Car class uses the Random class to vary the distance at which the car moves forward.
-There are comments that describe what the code is doing.

4 points:
-Scenario works correctly.
-1 or more mistakes.

3 points:
-The scenario is not functional, but there are parts that are correct.

1 point:
-Mrs. Nastasi is unable to view files.

Greenfoot(Adding Sound, Keeping Score, and Displaying Text)

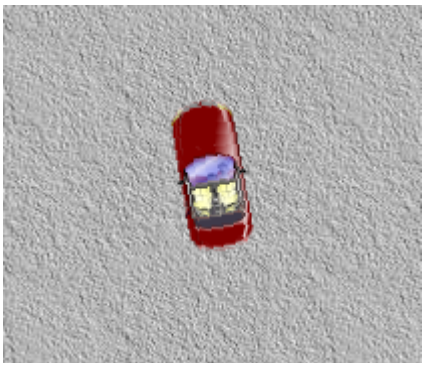
Students will modify a Greenfoot scenario so that a user can control a Car object using the arrow keys

Quick Review:
The Animal class and mouse instructions

- Copy and paste Animal class into the Scenario
- The move( ) method in the code from the Animal class:
- Remember when the Crab class used the Animal class' move( ) method to move in the world.

```
move( ); //This moves the object forward.
turn(-12); //This turns the object -12 degrees
if(Greenfoot.isKeyDown("left"))
{
   turn(12);
} //This code turns the object whenever the left key is pressed down.
```

Crazy Car (Greenfoot)



It's a car… and it's CRAZY!

Create a CrazyCar scenario using Greenfoot.

Introduction:

Using what you learned in the notes, you will interactively move the car with your keypad.

Instructions:

Create code in the CrazyCar class that allows a user to drive the car using the up, left, and right arrow keys.

Then, for kicks and giggles, see if you can create or modify code that will allow the user to drive in reverse using the down arrow key.

Program Requirements:

-When the CrazyCar Scenario is running, a user can move the Crazy Car forward, left, and right with the arrow keys.
-There are comments describing what the code does.

Saving Your File:

-When you are finished go to:
-Scenario → "Save a Copy As" → Save the project with the following name:

nastasi_CrazyCar_(LastNameFirstInitial)

For example, if your name is John Smith in 3rd period:

nastasi_CrazyCar_SmithJ

-Save this project to your Flash Drive. The entire FOLDER will go in the CLASS folder on the due date.

Rubric:

5 points:
-The scenario was saved correctly.
- When the CrazyCar Scenario is running, a user can move the Crazy Car forward, left, and right with the arrow keys.
-There are comments describing what the code does

4 points:
-Program works correctly.
-1 or more mistakes.

3 points:
-The program is functional, but it has errors.

1 point:
-Mrs. Nastasi is unable to view files.

A. Adding Sound

- To add sound to your scenario:
  - Create a "sounds" folder in the same directory as the "images" folder.
  - Place your sound files in the "images" folder. Greenfoot supports:
    - AIFF files
    - AU files
    - WAV files
  - Use the following code where you want the sound to execute:

Greenfoot.playSound("FILENAME");

  - Example:

Greenfoot.playSound("crash.wav");

B. Keeping Score

- Keeping a Score in your Greenfoot Scenario involves:
  - 1. Creating a variable the belongs to the main Actor class.
  - 2. Setting the beginning score in the Constructor.
  - 3. Decreasing or increasing the score depending upon an event.

- 1. Creating a variable the belongs to the main Actor class:
  - This variable will be a static variable since it belongs to the Actor class and NOT to one of the Actor objects.
  - Ex:

```
public static int intScore;
```

- 2. Set the beginning score in the Constructor.
- Ex:

```
public Car( )
{
  super("car", ".png", 3);//calling the superclass constructor
  intScore = 100; //This sets the default score
}
```

- 3. Decrease or increase the score depending upon an event.
- Ex:

```
if (collided != null)
{

    //This will remove the object
   getWorld().removeObject(collided);

   //This creates sound
   Greenfoot.playSound("crash.wav");

    //This decreases the score
    intScore = intScore - 1;
}
```

Displaying Text

- To display the score we need to:
  - Create a “Score” class that will extend the Actor class.
    - This “Score” class will have a constructor and an act( ) method that:
      - Create a Greenfoot image
      - Create text in the Greenfoot image
      - Grab the score from the Car class (Car.intScore)
      - Place the image in the world
  - Create a Score object upon initialization of the world.

- Creating a “Score” class:
- Ex of constructor:

```
public Score(int intPScore)//defining the Constructor
{
      //This creates a transparent image
      GreenfootImage img = new GreenfootImage (100, 30);

      //2 is 2 pixels down from the upper left of the image, 20 down
      //The drawString( ) method creates
      //We are grabbing the score from the Car class
      img.drawString("Score: " + Car.intScore, 2, 20);

      //Using the setImage method to set the image of the Score
      setImage(img);
}
```

- Ex of the act( ) method:

```
{
      //This recreates the image every time the act( )method is run
      GreenfootImage img = new GreenfootImage (100, 30);
      img.drawString("Score: " + Car.intPScore, 2, 20);
      setImage(img);
}
```

- Create a “Score” object upon the initialization in the world:

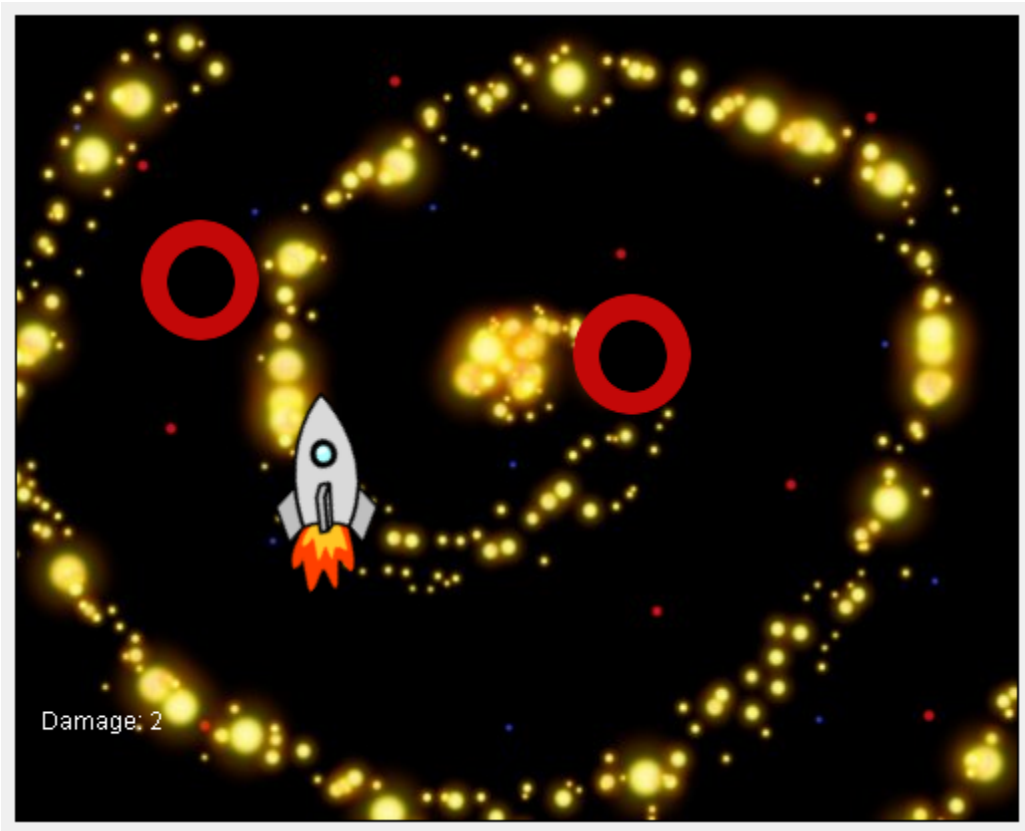
```
public Road()
{
     // Create a new world with 20x20 cells with a cell size of 10x10
      //pixels.
      super(500, 400, 1);

     //Let's add an object to the world
     addObject(new Car(), 250, 300);
      addObject(new Score(), 60, 30);
}
```

- Sometimes your background will be the same color as the text you want to display.

To solve this, set the color of the text using the GreenfootImage's objects setColor( ) method.
Ex:

```
public void act( )
{
    GreenfootImage img = new GreenfootImage (100, 30);
     img.setColor(Color.WHITE);
     img.drawString("Damage: " + Rocket.intScore, 2, 20);
     setImage(img);
}
```

Score and Sound (Greenfoot)



(Notice that we are keeping track of the damage now!)

Create a folder and name it: nastasi_SpaceGame_LastNameFirstInitial)
For example, if your name is John Smith, you will create a folder called:
nastasi_SpaceGame_SmithJ

Create a SpaceGame Greenfoot Scenario.

Introduction:

You will create sound for the SpaceGame whenever an energy bullet hits the rocket. Also, display a score in the RoadRage demo. You will keep track of how many energy bullets hit the rocket in this scenario.

Instructions:

1. Give your SpaceGame sound whenever an energy bullet hits the spaceship.

2. Create a variable for the Rocket class that keeps track of the rocket's damage! The damage should be initialized to 0 upon creation of the world and the rocket. Every time an energy bullet hits the rocket the damage should be increased by 1.

3. Display the amount of damage done to the rocket. By the way, here is a method to change the color of the image you will create to white. (If it is black it will blend in with the background of space.)

```
img.setColor(Color.WHITE);
```

Program Requirements:

-The SpaceGame scenario creates a sound whenever the rocket is hit by a bullet.
-The damage done to the rocket is visible and displayed in the scenario.

Saving Your File:

-Remember, make a copy of your Greenfoot scenario in the folder your created.
-

Rubric:

5 points:
-The SpaceGame scenario creates a sound whenever the rocket is hit by a bullet.
-The damage done to the rocket is visible and displayed in the scenario.

4 points:
-Scenario works correctly.
-1 or more mistakes.

3 points:
-The scenario is not functional, but there are parts that are correct.

1 point:
-Mrs. Nastasi is unable to view files.