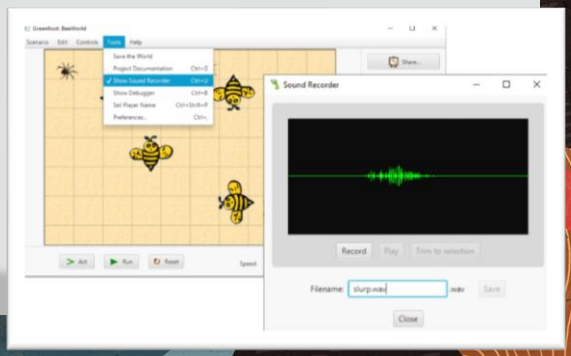
ORACLE

Academy

# Java Fundamentals

## 3-7

## Sound and Keyboard Control




ORACLE
Academy

# Objectives

- This lesson covers the following objectives:
  - Write programming statements to include sound in a program
  - Write programming statements to include keyboard movements in a program
  - Write programming statements to include mouse interaction in a program
  - Write programming statements to retrieve information from the user

# Keyboard Controls

- Games are controlled by a human or computer player using a remote control or keyboard controls
- To make a scenario behave like a true game, program statements that include keyboard controls so the player can control one or more objects in the game

ORACLE Academy





Sound and keyboard interactions can help bring a game to life.

We have used the isKeyDown method previously in the course to capture keyboard interaction with the user.

# The isKeyDown() Method

- The isKeyDown() method checks if a key on the keyboard has been pressed
  - Located in the Greenfoot class
  - Is a static method (associated with a class)
  - Returns true or false value
  - Expects a String argument in the parameter list
  - Can be used as a condition in an IF statement
- Method signature:

```
isKeyDown(String key)
```





We can use any key on the keyboard for our controls. When using the arrow keys we use the string values "left", "right", "up" or "down".

# String Parameter in isKeyDown() Method

- A String is a piece of text (word or sentence) written in double quotes
- For example:
  - "This is a String"
  - "A"
  - "name"


- The String parameter in the isKeyDown() method expects the name of the key to press on the keyboard
- Find a key's name by looking at your keyboard
  - Sometimes the name isn't evident (right cursor key is called "right")

# Using the isKeyDown() Method Example

- This code in the act() method uses the left and right keys on the keyboard to allow the player to control the Bee object's direction as it moves

```
/**
 * Act - do whatever the Bee wants to do. This method is called whenever
 * the 'Act' or 'Run' button gets pressed in the environment.
 */
public void act()
{
    move(3);
    if(Greenfoot.isKeyDown("left")){
        turn(-2);
    }else if(Greenfoot.isKeyDown("right")){
        turn(2);
    }//endif
}//end method act
```

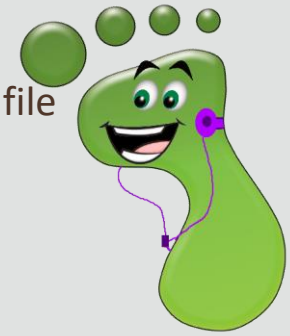




Remember that a positive turn integer rotates the actor clockwise and a negative value counter-clockwise.

# Include Sound in Your Game

- Sounds can enhance your game
  - Give feedback sounds to the player when they win, lose, or achieve minor victories throughout the game
  - Include background sounds in a game
- The playSound() method is used to play sounds in a game
  - Method is located in the Greenfoot class
  - Parameter list expects the name of a sound file (as String) as an argument
  - The method does not return data







Sounds can increase the excitement in a game. Perhaps a beep that gets quicker as you get closer to an item, or a Hooray when an achievement is reached. Adding your own sound effects can really personalize a game.

# Sound Example

- The playSound() method is called using dot notation in the body of the catchfly() method
- Whenever the Bee object catches a fly, it makes a sound

```
/**
 * catchFly - if the Bee touches a fly the fly is removed
 * A sound is played
 */
private void catchFly(){
        if(isTouching(Fly.class)){
            removeTouching(Fly.class);
            Greenfoot.playSound("slurp.wav");
         }//endif
}
```





Sound files in Greenfoot can be either wav or mp3 files and are stored in the sounds sub folder within the project.

# Steps to Record Original Sounds

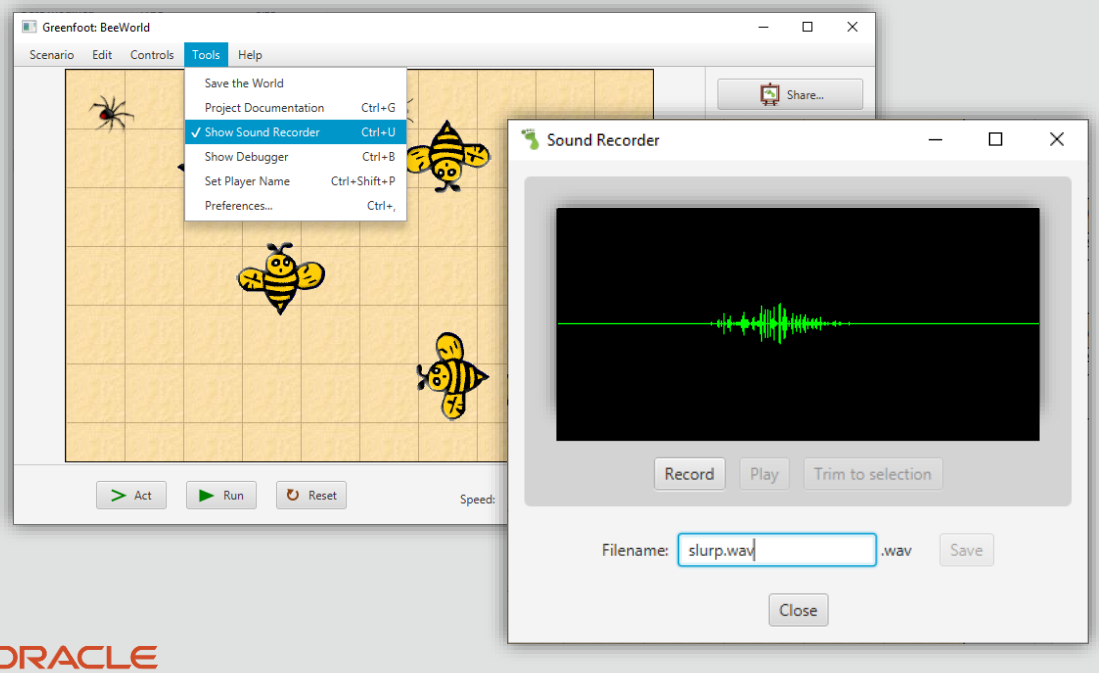
- In the Tools menu in the environment, select Show Sound Recorder
- Press Record, then talk into your computer's microphone to record sound
- Press Stop Recording when finished
- Press Play to play back the sound
- Re-record if necessary
- Enter a file name, then click Save to save the file to the sounds directory of your scenario
- The file is now ready to reference in your code





If you don't have a microphone you can download sound samples from the internet. Always remember about copyright laws.

# Greenfoot Sound Recorder Display



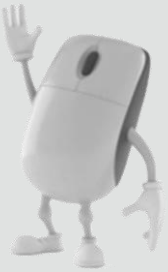





Select the Controls menu then Sound Recorder to open the Sound Recorder window

# Using the Mouse

- Greenfoot allows multiple input methods rather than just using the keyboard
- There is also the ability to use controllers, mice and other input devices
- You may wish to use a mouse within the scenario you are building, rather than the keyboard
- The Greenfoot class has a number of methods that allow you to get information on the mouse actions
- These include:
  - `getMouseInfo(), mouseClicked(), mouseDraggedEnded(), mouseDragged(), mousePressed()`







Other input devices such as the Xbox Kinnect can be used in the Greenfoot environment.
http://www.greenfoot.org/doc/kinect

# Using the Mouse Example

- The scenario we are building does not use mouse controls, but let's show an example
- If we had an actor called Spider and we wished to detect when the mouse was clicked on an instance of it we would do the following:

```
public void act()
{
    if (Greenfoot.mouseClicked(this)) {
        //do something
    }
}
```





"this" refers to the current class. In this example we are referring to the Spider class so it will only detect when an instance of a Spider is clicked.

# Using the Mouse Example

- If we wanted to detect if the mouse was clicked elsewhere we would use the MouseInfo class
- Below we see code that would move the current instance to the location where the mouse was clicked

```
public void act()
{
    MouseInfo mouse = Greenfoot.getMouseInfo();
    if(mouse!=null){
        if (mouse.getButton() == 1)  {
            setLocation(mouse.getX(),mouse.getY());
        }//endif
    }//endif
}//end method act
```





We could have used the actor method of Spider called turnTowards(x,y). This would then turn the spider to move in the direction of where we wanted an actor to move to. i.e. where we clicked.
We compare mouse.getButton() to 1 which is the left mouse button.
If you have a three button mouse then 1 is normally the left, 2 is the middle and 3 is the right.

# Obtaining Keyboard Input From The User

- There may be a point in your program that you wish to gain input from the user
- i.e.
  - Asking for their name
  - Asking for a starting speed etc
- From version 2.4.1 of Greenfoot this is now possible through the Greenfoot method called ask()

```
String Greenfoot.ask(String message)
```

ORACLE Academy





We could create a high score table that would prompt for the name of the user and display their position in the best scores.

# Obtaining Keyboard Input From The User

- The ask() method will display the message as a prompt and obtain the result as a string
- i.e., Let us ask the user their name and store it in the variable name
- While Greenfoot is waiting for your response it will pause the world and its actors

```
public void act()
{
     String name = Greenfoot.ask("Please input your name: ");
}//end method act
```

- Would produce


ORACLE Academy





If you were wanting a number to come back like your age, you would have to store this in a string and then convert it with a Java method like Integer.parseInt() to convert it from a string to a number.

# Terminology

- Key terms used in this lesson included:
  - Keyboard control
  - Play Sounds
  - Mouse Interaction
  - Ask

# Summary

- In this lesson, you should have learned how to:
  - Write programming statements to include sound in a program
  - Write programming statements to include keyboard movements in a program
  - Write programming statements to receive the mouse state
  - Write programming statements to retrieve a response from the user


ORACLE
Academy