

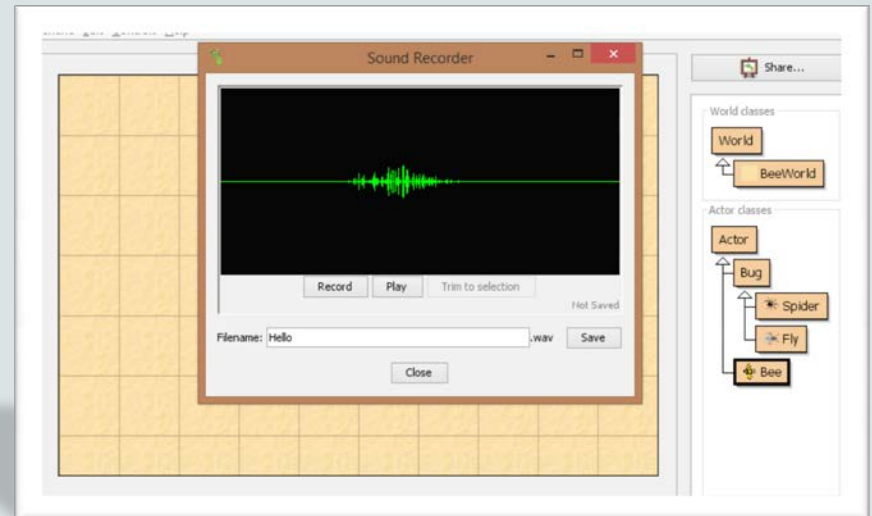


Java Fundamentals

3-7

Sound and Keyboard Control

```
/**
 * Act - Bee constantly fly with the user
 * the 'Act' or 'Run' button gets pressed
 */
public void act()
{
    move(3);
    if (Greenfoot.isKeyDown("left")) {
        turn(-2);
    }
    if (Greenfoot.isKeyDown("right")) {
        turn(2);
    }
}
```



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.

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:

```
public static boolean isKeyDown(String key)
```

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.

```
public class Bee extends Actor
{
    /**
     * Act - Bee constantly fly with the user a
     * the 'Act' or 'Run' button gets pressed i
     */
    public void act()
    {
        move(3);
        if (Greenfoot.isKeyDown("left")) {
            turn(-2);
        }
        if (Greenfoot.isKeyDown("right")) {
            turn(2);
        }
    }
}
```

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.

Sound Example

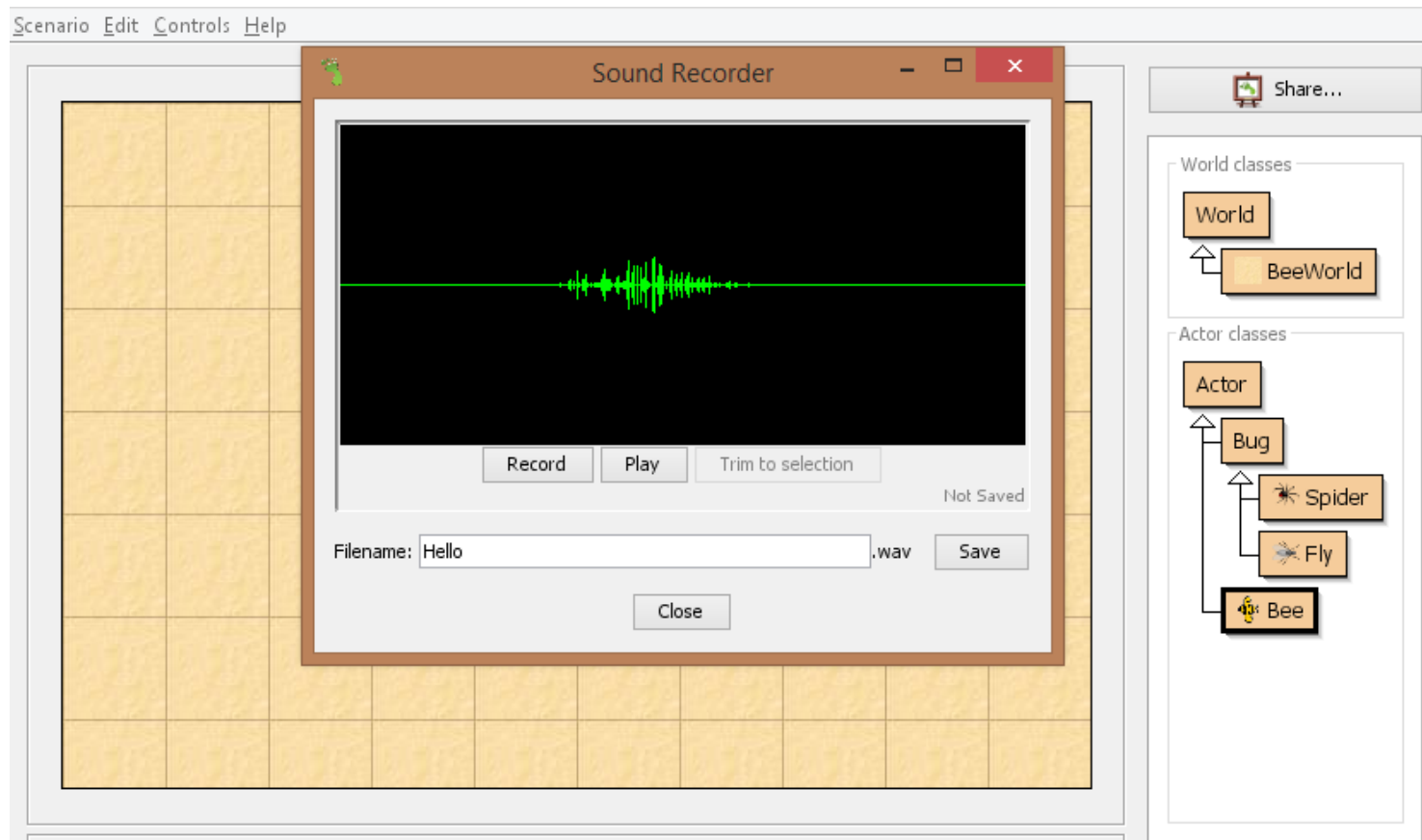
- The playSound method is called using dot notation in the body of the catchFly method.
- Whenever the Bee object catches a fly, it makes a sound.

```
/**  
 * catchFly. If the bee touches a fly remove it and play a sound.  
 */  
public void catchFly() {  
    if (isTouching(Fly.class)) {  
        removeTouching(Fly.class);  
        Greenfoot.playSound("slurp.wav");  
    }  
}
```

Steps to Record Original Sounds

- In the Controls 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 your scenario.
- The file is now ready to reference in your code.

Greenfoot Sound Recorder Display



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()`, `mouseDraged`, `mousePressed()`

Using the Mouse Example

- The scenario we are building does not use mouse controls, but lets 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:

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

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.

```
MouseInfo mouse = Greenfoot.getMouseInfo();  
if(mouse!=null){  
    if (mouse.getButton() == 1) {  
        setLocation(mouse.getX(),mouse.getY());  
    }  
}
```

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)
```

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.

```
String name = Greenfoot.ask("Please input your name: ");
```

- Would produce



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.

