Creating Java Programs with Greenfoot

Understanding Abstraction

Overview

This lesson covers the following topics:

- Define abstraction and provide an example of when it is used

Abstraction

- You can program a new instance to perform a single, specific task, such as play a sound when a specific keyboard key is pressed, or display a set of questions and answers every time a game is started.
- To create programs on a larger scale, for example one that creates 10 objects that each perform different actions, you need to write programming statements that let you repeatedly create objects that perform different tasks, by just providing the specifics for the differences.

Abstraction Example

- For example, if you are going to create 10 objects programmatically, all placed in different locations, it is inefficient to write 10 lines of code for each object.
- Instead, you abstract the code and write more generic statements to handle the creation and positioning of the objects.

Abstraction Principle

- Abstraction aims to reduce duplication of information in a program by making use of abstractions.
- The abstraction principle can be a general thought such as “don’t repeat yourself.”
- For example, you want to create a game board that has blocks, trees, sticks, and widgits.
  - You do not need to write repetitive programming statements to add each of these items.
  - Instead, you can abstract the procedure to simply add objects to a game board in a specific location.

Abstraction Pseudocode Example

- For example, you will display a Duke image and when called, it is either going to point its hand up to the sky, out to the side, or down to the ground.
- Your code will display Duke and specify the direction to point. Here is the pseudocode:
  - Create new Duke (point left, position x16, y20, z0)
  - Create new Duke (point up, position x34, y52, z0)
  - Create new Duke (point down, position x58, y71, z0)

Abstraction Pseudocode Example

- Imagine the code needed for 300 Duke images.
  - To implement abstraction, create a procedure that creates a new object that is positioned where needed and displays the appropriate image.
    - Call Procedure newObject (image, position)

Abstraction Techniques

- Abstraction occurs many ways in programming.
  - One technique is to abstract programming code using variables and parameters to pass different types of information to a statement.
  - Another technique is to identify similar programming statements in different parts of a program that can be implemented in just one place by abstracting out the varying parts.

Abstraction Techniques Example

- For example, in a game tracking points, you may have an action decrease points from a running total and in another section of a game you may have elapsed time values decrease points from a running total.
- You could use abstraction to have an event decrease points by specifying the type of event and the amount to decrease from the running total.

Constructor Using Variables

In this example, the Duke constructor has variables defined to store the key and sound values.

```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class Duke here.
 *
 * @author Oracle Academy
 * @version JF_S03_L05
 */
public class Duke extends Animal
{
    private GreenfootImage image1;
    private GreenfootImage image2;
    private boolean isKeyDown;
    private String key;
    private String sound;
    /**
     * Create a Duke and initialize his two images. Link Duke to a specific keyboard
     * key and sound.
     */
    public Duke(String keyName, String soundFile)
    {
        key = keyName;
        sound = soundFile;
        image1 = new GreenfootImage("Duke.PNG");
        image2 = new GreenfootImage("duke2.png");
        setImage(image1);
    }
```

Programming to Place Instances

- After sound and key variables are defined in a constructor, write programming code to automatically add instances of the class to the world.
- The following programming statement added to the Duke class:
  - Creates a new instance of Duke each time DukeWorld is re-initialized, with a specific key and sound file.
  - Places the instance in DukeWorld at the specific x and y coordinates.

```
addObject (new Duke ( “k” , “test.wav” ), 150, 150);
```

Constructor Example

Examine the addObject statement in the DukeWorld constructor.

```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class DukeWorld here.
 *
 * @author Oracle Academy
 * @version JF_S03_L05
 */
public class DukeWorld extends World
{
    /**
     * This constructor creates a new world and the objects that start the
     * game.
     */
    public DukeWorld()
    {
        super(600, 400, 1);
        addObject (new Duke ("k", "test.wav"), 150, 100);
    }
}
```

Abstract Code to a Method

- You can anticipate abstraction during the design phase of a project, or you can examine programming code to identify statements that would benefit from abstraction.
- Often times you will recognize an opportunity to abstract programming statements when writing lines of code that appear repetitive.

Abstract Code to a Method Example

Examine the code below and on the following slide.

```
public class ZombieWorld extends world
{
    /**Constructor creates new world and initial game objects
    */
    public ZombieWorld()
    {
          super(600, 400, 1);
          addObject (new Zombie ( “1” ,  “1.wav” ), 1, 10);
          addObject (new Zombie ( “2” ,  “2.wav” ), 2, 10);
          addObject (new Zombie ( “3” ,  “3.wav” ), 3, 10);
          addObject (new Zombie ( “4” ,  “4.wav” ), 4, 10);
          addObject (new Zombie ( “5” ,  “5.wav” ), 5, 10);
          addObject (new Zombie ( “6” ,  “6.wav” ), 6, 10);
          addObject (new Zombie ( “7” ,  “7.wav” ), 7, 10);
          addObject (new Zombie ( “8” ,  “8.wav” ), 8, 10);
          addObject (new Zombie ( “9” ,  “9.wav” ), 9, 10);
    }
}
```

Abstract Code to a Method Example

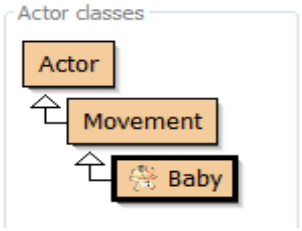
```
public class ZombieWorld extends world
{
    /**Declare constructor variables
    */
    public string varKey;
    public string varSound;
    public int varX;
    int varY=10;

    /**Constructor creates new world and initial game objects
    */

    public ZombieWorld()
    {
          super(600, 400, 1);
          for (int i=1; i<10; i++) {
          addObject (new Zombie (varKey=i, varSound=i+" .wav" ) varX,   varY);)
    }
}
```

Simple Abstraction of Code

- What if you had little knowledge of Greenfoot or programming, and wanted to create a game to move a baby around the screen?
  - You could simplify how to move an Actor object around the screen by creating a simple set of movement methods: moveRight, moveLeft, moveUp and moveDown.
  - This provides a simpler abstraction than the standard Greenfoot API with its built-in setLocation and getX/getY methods.



Create Baby Subclass

- Create a subclass of the Actor class called Movement that would allow the player to tell the Baby actor to move in a desired direction.
- Add the following code to Movement which will set the amount of movement each time a move is required.

```
import greenfoot.*;
//  An actor superclass that provides movement in four directions.

public class Movements extends Actor {

private static final int speed = 4;

}
```

Create Move Methods for Baby Subclass

Then, add the following move methods to make the movement. These methods simplify and Abstract the Greenfoot API of getX/getY.

```
public void moveRight()
{
      setLocation ( getX() + speed, getY() );
}
public void moveLeft()
{
      setLocation ( getX() - speed, getY() );
}

public void moveUp()
{
      setLocation ( getX(), getY() - speed );
}
public void moveDown()
{
      setLocation ( getX(), getY() + speed );
}
```

Code the act Method

- Code the act method of the Baby Actor class so its instances move when the arrow keys are pressed.
- This abstraction hides and automates the more complex code, only showing moveLeft, moveRight, etc.

```
public void act()
{
     if (Greenfoot.isKeyDown("left") )
     {
          moveLeft();
     }

     if (Greenfoot.isKeyDown("right") )
     {
          moveRight();
     }
}
```

Terminology

Key terms used in this lesson included:

- Abstraction

Summary

In this lesson, you should have learned how to:

- Define abstraction and provide an example of when it is used

Practice

The exercises for this lesson cover the following topics:

- Actor abstraction
- Journaling abstraction