HW#3

Computer Science: Program Your Own RPG

Instructions: Complete as much as you can in one hour and then stop. (Do your best.) Do this one on your own.

Tomorrow we will begin implementing graphics in our games with the help of some libraries. Read the example code below, then do the exercises that follow. Try to understand what the code does, but do not worry if you don't understand everything.

```
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.io.File;
import sprite.Img;
import sprite.ImgUpload;
import world.SimpleObject;

public class Player extends SimpleObject implements KeyListener {
	private Img RobN = ImgUpload.getInstance(new File("sprites")).getImg("NRobWalk.png");
	private Img RobS = ImgUpload.getInstance(new File("sprites")).getImg("SRobWalk.png");
	private Img RobE = ImgUpload.getInstance(new File("sprites")).getImg("ERobWalk.png");
	private int speed = 10;
	private int x_dir = 0;
	private int y_dir = 0;
	private boolean move = false;

	public Player() {
		super("sprites/SRobWalk.png");
	}

	@Override
	public char id() {
		return 'P';
	}

	@Override
	public void collision(SimpleObject s) {
	}

	@Override
	public void update() {
		if (move) {
			this.moveCell(x_dir, y_dir, speed, true);
		} else {
			this.getImage().setSlide(0);
		}
	}
```

```
@Override
public void keyTyped(KeyEvent e) {}

@Override
public void keyPressed(KeyEvent e) {
	switch (e.getKeyCode()) {
	case KeyEvent.VK_ESCAPE:
		System.exit(0);
		break;
	case KeyEvent.VK_LEFT:
		this.setDrawMode(SimpleObject.FLIP_X, 0);
		this.setImage(RobE);
		x_dir = -1;
		move = true;
		break;
	case KeyEvent.VK_RIGHT:
		this.setDrawMode(SimpleObject.NONE, 0);
		this.setImage(RobE);
		x_dir = 1;
		move = true;
		break;
	case KeyEvent.VK_UP:
		this.setDrawMode(SimpleObject.NONE, 0);
		this.setImage(RobN);
		y_dir = -1;
		move = true;
		break;
	case KeyEvent.VK_DOWN:
		this.setDrawMode(SimpleObject.NONE, 0);
		this.setImage(RobS);
		y_dir = 1;
		move = true;
		break;
	case KeyEvent.VK_SPACE:
		this.playSound("sounds/evillaugh.wav");
		break;
	}

	if (x_dir != 0 && y_dir != 0) {
		speed = 7;
	}
}

@Override
public void keyReleased(KeyEvent e) {
	switch (e.getKeyCode()) {
	case KeyEvent.VK_LEFT:
		x_dir = 0;
		break;
```

```
                case KeyEvent.VK_RIGHT:
                        x_dir = 0;
                        break;
                case KeyEvent.VK_UP:
                        y_dir = 0;
                        break;
                case KeyEvent.VK_DOWN:
                        y_dir = 0;
                        break;
                case KeyEvent.VK_SPACE:
                        break;
                }

                if ((x_dir & y_dir) == 0) {
                        if (y_dir + x_dir == 0) {
                                move = false;
                        }
                        speed = 5;
                }
        }
}
```

Explanation:

public class Player extends SimpleObject implements KeyListener

The class *Player* uses methods that are contained inside the *SimpleObject* class. In order to use the these methods we must extend *SimpleObject*. The Player also uses keys typed by a user, and it implements *KeyListener* so that other classes know that *Player* has the methods:

public void keyTyped(KeyEvent e)
public void keyPressed(KeyEvent e)
public void keyReleased(KeyEvent e)

You may notice that each of these methods have *@Override* above them. This command is optional. It tells the compiler that each of these methods are inside *KeyListener.*

public char id()
public void collision(SimpleObject s)
public void update()

These methods come from the *SimpleObject* class, and similarly have *@Override* above them.

this.moveCell(x_dir, y_dir, speed, true);

The *this* refers to *Player* and lets us look at all the methods that are inside *Player* including all of the ones that *SimpleObject* have. There are several more methods that can be accessed through *this*. The *moveCell* method lets you move from one x, y coordinate to another. The *speed*

is a number representing how many frames it should take to move. The boolean *true* tells *SimpleObject* that we are moving relatively from where we are instead of to an absolute coordinate.

private Img RobN = ImgUpload.getInstance(new File("sprites")).getImg("NRobWalk.png");

This gets an image (*Img*) named "*NRobWalk.png*" loaded from a file called "*sprites*", and loads it into the variable *RobN*.

1. The following simple player has an id, '*S*', and moves right until it collides with an object, and then it moves left. Complete the code for the simple player below:

```
import world._____;

public class SimplePlayer extends SimpleObject{
	private boolean _____ = false;

	public SimplePlayer() {
		//A red 20x20 square.
		this.setImage(0xFFFF0000, 20, 20);
	}

	@Override
	public char id() {
		return '_____';
	}

	@Override
	public void collision(SimpleObject s) {
		_____;
	}

	@Override
	public void update() {
		if (_____) {
			this.moveCell(1, 0, 10 ,true);
		} else {
			_____;
		}
	}
}
```

2. For the *Player* above the game development team needs to change the buttons for moving. Instead of the arrow keys they want *Player* to use W, A, S and D to move, and ENTER to laugh. How would you change the code above to accommodate the team's need? (Give your best guess.)

_____

_____

_____