State Patter Sample Codes:

public abstract class MonkeyStatePattern {  
  
 public MonkeyStatePattern keyUp(Monkey monkey) {  
 return this.keyUp(monkey);  
 }  
  
 public MonkeyStatePattern keyRight(Monkey monkey) {  
 return this.keyRight(monkey);  
 }  
  
 public MonkeyStatePattern keyDown(Monkey monkey) {  
 return this.keyDown(monkey);  
 }  
  
 public MonkeyStatePattern keyLeft(Monkey monkey) {  
 return this.keyLeft(monkey);  
 }  
  
 public MonkeyStatePattern keyReleased(Monkey monkey) {  
 return this.keyReleased(monkey);  
 }  
}

MonkeyStatePattern is an abstract class here. This class defines all the possible state that can be achieved by key action. There are 5 key actions that can be done.

KeyUp when UP Arrow is pressed.

KeyDown when DOWN Arrow is pressed.

KeyLeft when LEFT Arrow is pressed.

KeyRight when RIGHT Arrow is pressed.

KeyReleased when NO Keys are pressed.

*/\*\*  
 \*   
 \*/*package com.adipandey.monkeybanana.states;  
  
import com.adipandey.monkeybanana.MonkeyStatePattern;  
import com.adipandey.monkeybanana.Constants;  
import com.adipandey.monkeybanana.Monkey;  
  
public class MoveStatePattern extends MonkeyStatePattern {  
  
 public MonkeyStatePattern keyUp(Monkey monkey) {  
 monkey.setY(monkey.getY() - Constants.*BOX*);  
 monkey.setMonkeyStatePattern(this);  
 return this;  
 }  
  
 public MonkeyStatePattern keyDown(Monkey monkey) {  
 monkey.setY(monkey.getY() + Constants.*BOX*);  
 monkey.setMonkeyStatePattern(this);  
 return this;  
 }  
  
 public MonkeyStatePattern keyLeft(Monkey monkey) {  
 monkey.setX(monkey.getX() - Constants.*BOX*);  
 monkey.setMonkeyStatePattern(this);  
 return this;  
 }  
  
 public MonkeyStatePattern keyRight(Monkey monkey) {  
 monkey.setX(monkey.getX() + Constants.*BOX*);  
 monkey.setMonkeyStatePattern(this);  
 return this;  
 }  
  
 public MonkeyStatePattern keyReleased(Monkey monkey) {  
 monkey.setMonkeyStatePattern(new RestStatePattern());  
 return new RestStatePattern();  
 }  
}

MoveStatePattern extends MonkeyStatePattern. All possible combination are checked while moving. And respective actions are performed.

Suppose X Co-ordinate of a Monkey is 420. If UP key is pressed the X coordinate does not change but the Y coordinate changes.

public MonkeyStatePattern keyUp(Monkey monkey) {  
 return monkeyStatePattern.keyUp(this);  
 }  
  
 public MonkeyStatePattern keyDown(Monkey monkey) {  
 return monkeyStatePattern.keyDown(this);  
 }  
  
 public MonkeyStatePattern keyLeft(Monkey monkey) {  
 return monkeyStatePattern.keyLeft(this);  
 }  
  
 public MonkeyStatePattern keyRight(Monkey monkey) {  
 return monkeyStatePattern.keyRight(this);  
 }  
  
 public MonkeyStatePattern keyReleased(Monkey monkey) {  
 return monkeyStatePattern.keyReleased(this);  
 }  
  
 public int getX() {  
 return x;  
 }  
  
 public int getY() {  
 return y;  
 }  
  
 public void setX(int x) {  
 this.x = x;  
 }  
  
 public void setY(int y) {  
 this.y = y;  
 }  
}

public MonkeyStatePattern keyUp(Monkey monkey) {  
 return monkeyStatePattern.keyUp(this);  
 }  
  
 public MonkeyStatePattern keyDown(Monkey monkey) {  
 return monkeyStatePattern.keyDown(this);  
 }  
  
 public MonkeyStatePattern keyLeft(Monkey monkey) {  
 return monkeyStatePattern.keyLeft(this);  
 }  
  
 public MonkeyStatePattern keyRight(Monkey monkey) {  
 return monkeyStatePattern.keyRight(this);  
 }  
  
 public MonkeyStatePattern keyReleased(Monkey monkey) {  
 return monkeyStatePattern.keyReleased(this);  
 }  
  
 public int getX() {  
 return x;  
 }  
  
 public int getY() {  
 return y;  
 }  
  
 public void setX(int x) {  
 this.x = x;  
 }  
  
 public void setY(int y) {  
 this.y = y;  
 }  
}

setX and setY are the method which sets the X and Y coordinate.

@Override  
public void keyPressed(KeyEvent e) {  
  
 int code = e.getKeyCode();  
  
 if (code == KeyEvent.*VK\_LEFT*) {  
 if (monkey.getX() < 20) {  
 monkey.setX(20);  
 }  
 monkey.keyLeft(monkey);  
 }  
  
 if (code == KeyEvent.*VK\_RIGHT*) {  
 if (monkey.getX() > 540) {  
 monkey.setX(540);  
 }  
 monkey.keyRight(monkey);  
 }  
  
 if (code == KeyEvent.*VK\_UP*) {  
  
 if (monkey.getY() < 20 ){  
 monkey.setY(20);  
 }  
 monkey.keyUp(monkey);  
 }  
   
 if (code == KeyEvent.*VK\_DOWN*) {  
 if (monkey.getY() > 500) {  
 monkey.setY(500);  
 }  
 monkey.keyDown(monkey);  
 }

@Override  
public void start(Stage primaryStage) throws Exception {  
 *play*();  
}  
  
static void play() {  
 try {  
 String bip = "resources/media/cheer.mp3";  
 Media hit = new Media(new File(bip).toURI().toString());  
 MediaPlayer mediaPlayer = new MediaPlayer(hit);  
 mediaPlayer.play();  
 new Timer().schedule(new TimerTask() {  
 @Override  
 public void run() {  
 mediaPlayer.stop();  
 }  
 }, 750);  
 } catch (IllegalStateException e) {  
 Application.*launch*(BGPlayer.class);  
 *play*();  
 }  
  
}  
  
static void playMusic() {  
 Application.*launch*(BGPlayer.class);  
}  
  
  
public BGPlayer() {  
}

**Design :**

There are two functions; key press and key release using the keyboard cursor Keys. User(Monkey) has two states, move and idle. The Monkey class contains 2 methods keyPressed and keyReleased which represents the 2 states that the monkey can be in. The Monkey needs to eat banana withing a specified time to win. A total of xyz banana has to be eaten in a given time to win the game.

**Implementation:**

The MonkeyBanana Game used Java Swings to build. There are 2 Separate classes for Monkey and Banana. ActionListners and Event Listeners are used to get input from the User. We have a JPanel which has all the other components. JPanel has a JFrame in it. JFrame has the gameGird. User is the Monkey here and has got two States. Both states overrides the keyPressed and KeyReleased inbuild java methods which allow the monkey to change states and move.

Goal, In 40 seconds, eat 10 bananas. Each round is of 10 seconds. Your goal is to eat 10 in 40.

**Source Code:**

Source code above uses various state and object patterns. KeyListeners and EventListeners are used here. These drive the user to various positions.