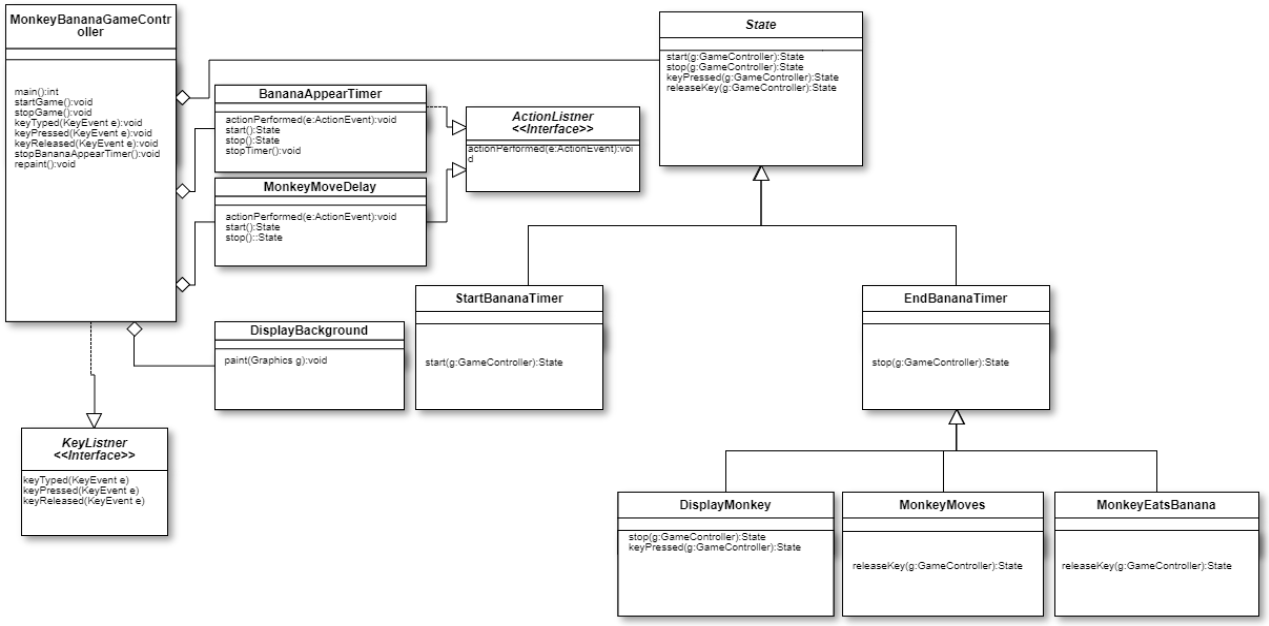
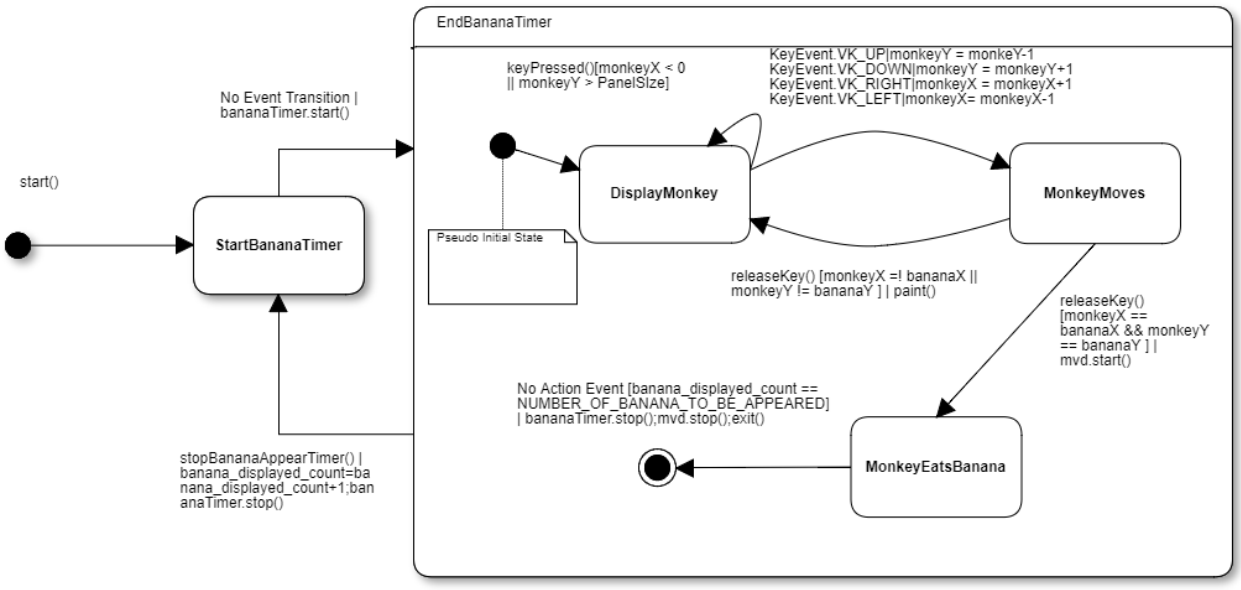
Monkey Banana Game – SDP Home Work – 1

DCD

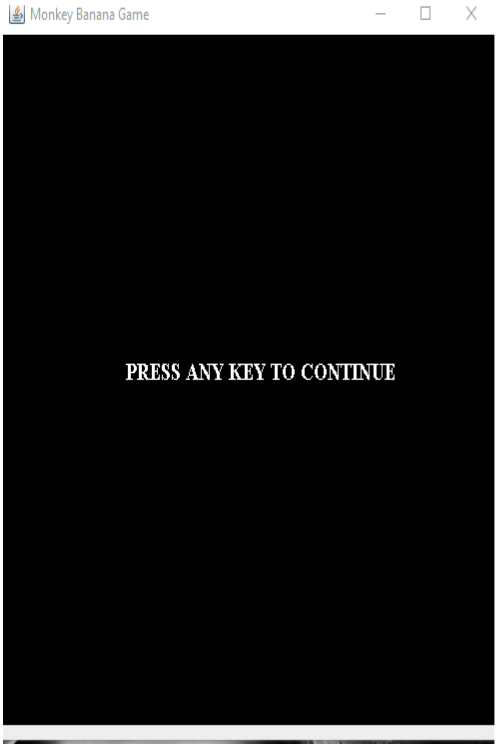


State Diagram – Formal

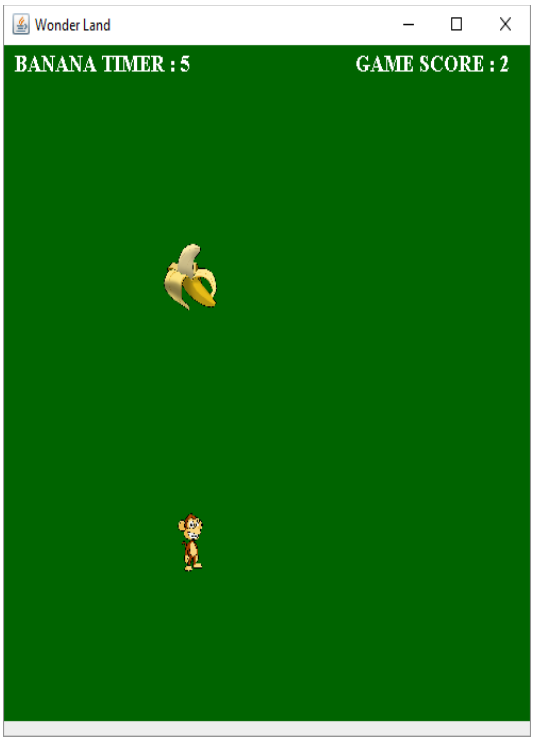


Screen Shots:

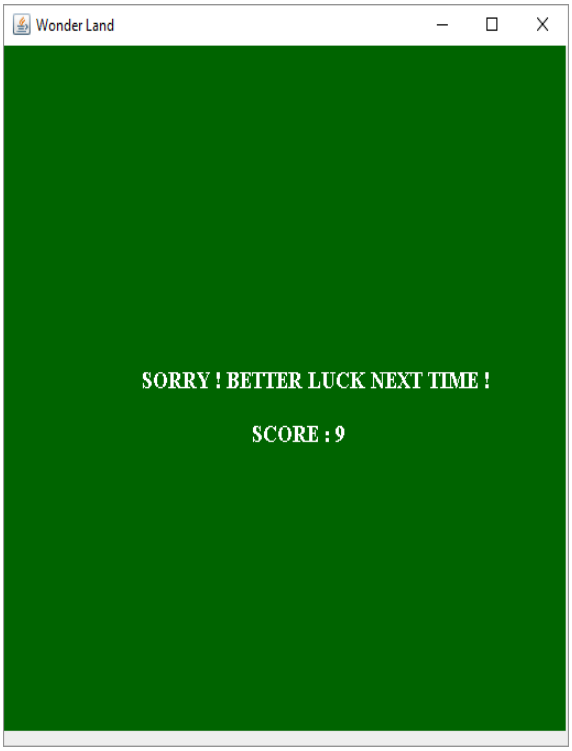
1. Initial State



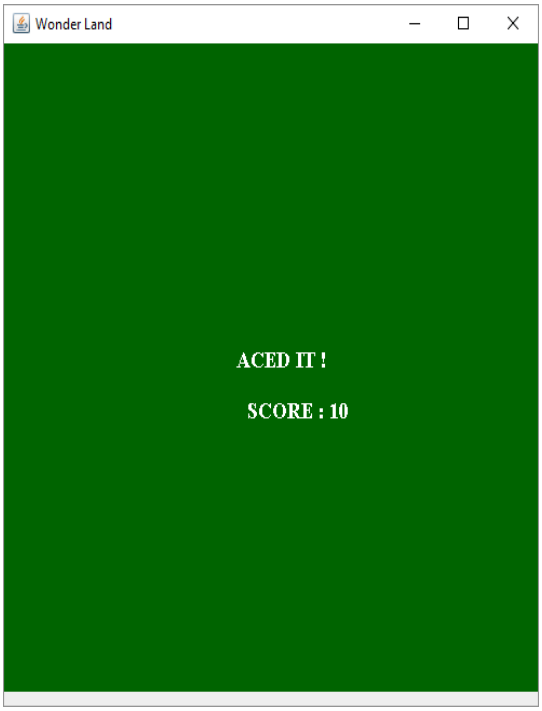
2. Monkey Move State



3. Lost



4. Won Game



Code Snippets Illustrating the usage of State Pattern

1. Controller state transformations (Subject class). ‘KeyPressed’ function is the one that gets the new state.

```
@Override
public void keyReleased(KeyEvent e) {
    // TODO Auto-generated method stub
    state = state.releaseKey(this);
}

@Override
public void keyPressed(KeyEvent e) {
    // TODO Auto-generated method stub
    if(START_GAME){
        score = 0;

        START_GAME = false;

        banana_displayed_count = 0;

        this.startGame();
        return;
    }

    state = state.keyPressed(this, e.getKeyCode());


}
```

2. Illustration of few of the state change in ‘MonkeyMove’ state based on key press

```
public class DisplayMonkey extends EndBananaTimer {

    public State keyPressed(MonkeyBananaGameController g,int keyCode){
        switch(keyCode){
            case KeyEvent.VK_LEFT:
                if(g.mnky.monkeyX>0){
                    g.mnky.monkeyX--;
                    return new MonkeyMoves();
                }
                break;

            case KeyEvent.VK_RIGHT:
                if(g.mnky.monkeyX<MonkeyBananaGameController.ROW_COL_COUNT-1){
                    g.mnky.monkeyX++;
                    return new MonkeyMoves();
                }
                break;

            case KeyEvent.VK_UP:
                if(g.mnky.monkeyY>1){
                    g.mnky.monkeyY--;;
                    return new MonkeyMoves();
                }
                break;
```

3. Code snippet showing that the StartBnananTimer extending State abstract class.

```
public class StartBananaTimer extends State {

    public State start(MonkeyBananaGameController g){
```

Write-up about the design

The Monkey Banana/Wonder Land game has launcher class named **MonkeyBananaGameController.** key press, key release and type events from keyboard are listened by it. The game has the following states : **DisplayMonkey, MonkeyMoves** and **MonkeyEats.**

The game starts after pressing any key. Monkey and Banana are displayed at random positions. As soon as the banana appears, its timer (**BananaAppearTimer**) starts ticking.

Initially, monkey position is randomly generated.

Note: Following are functions in State abstract class. Start(), stop(), keyPressed().

Step 1: When the user presses any key, start() function on the StartBananaTimer is called. This function starts the ticker for banana timer and randomly places banana. It is made sure that banana position does not collide with monkey position.

Once the above tasks are the done, state is transformed to DisplayMonkey.

Step 2: Whenever the user clicks one of the arrow keys (left, right, up, down), keyPressed function is called. These functions increases the monkey position by 1 box, either on X or Y axis (provided, monkey does not go beyond the wall)

If the monkey does not hit the wall it will move to MonkeyMoves state. Else it will remain in same state.

Step 3: When key is released, update position of the monkey by calling repaint. If when key is released, monkey and banana are in same position, then game score is incremented.

Step 4: Post eating banana, recalculate the random banana position by returning DisplayMonkey state and resetting BananaAppearTimer count to 1. Also increments the banana_displayed_ count.

Step 5: The game ends if banana_displayed_ count = total banana to be appeared.

Step 6: If gamer's score equals the total banana to be appeared then user wins. Else he loses.