

A

PROJECT REPORT

ON

SNOWMAN

UNDER

NON SYLLABUS PROJECT



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
POORNIMA INSTITUTE OF ENGINEERING & TECHNOLOGY,
JAIPUR

(Academic Year 2022-23)(Odd)

Submitted To:

Mr. Atrakesh Pandey

Submitted by:

Vaibhav Gupta (PIET20CS186)

DECLARATION

I hereby declare that the report entitled “**SNOWMAN**” was carried out and written by me under the guidance of ”Mr. Atrakesh Pandey”, Professor, DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING, Poornima Institute of Engineering & Technology, Jaipur. This work has not been previously formed the basis for the award of any degree or diploma or certificate nor has been submitted elsewhere for the award of any degree or diploma.

Place: - PIET

Student Name: - Vaibhav Gupta

Date: - 04.1.2023

Registration Number: - PIET20CS186

TABLE OF CONTENTS

S. No.	Description	Page No.
1	Title Page	i
2	Declaration	ii
3	Table of Contents	iii - iv
4	List of Tables	v
5	List of Figures	vi
6	Abbreviation	vii
7	Chapter 1 - Introduction	1 - 2
8	• Objective of Project	1
9	• Constraints and Dependency	1
10	• Project Life Cycle	1 - 2
11	Chapter 2 - Requirement Analysis	3 - 5
12	• Functional Requirements	3
13	• Non-Functional Requirements	3
14	• Technology Used	3 – 6
16	Chapter 3 - Design	7– 9
17	• DFD	7
18	• Use Case Diagram	8
19	• Class Diagram	9
20	Chapter 4 - Conclusion	10
21	Chapter 5 - References	11
22	Chapter 6 – Snapshots of Project	12 - 17

25	Chapter 7 – Code of Project	18 - 19
----	-----------------------------	---------

List of Tables

S. No.	Table	Page No.
2.1	System requirements for IntelliJ IDEA	6

List of Figures

S. No.	Figure	Page No.
1.1	Spiral Model	2
3.1	DFD	7
3.2	Use Case Diagram	8
3.3	Class Diagram	9
5.1	Original position	12
5.2	Move Left	13
5.3	Move Right	14
5.4	Move Up	15
5.5	Move Down	16
5.6	Reset	17
6.1	Code Screenshot	18
6.2	Code Screenshot	18
6.3	Code Screenshot	19
6.4	Code Screenshot	19

Abbreviations

- SQL: - Structured Query Language
- APL: - APPLET
- OOL: - Object Oriented Language
- DFD: - Data Flow Diagram
- UI : - User Interface
- RAM: - Random Access Memory
- JDK: - Java Development Kit
- JRE: - Java Runtime Environment

Chapter 1

Introduction to Project

1.1. Objective of Project

As an NSP project, SNOWMAN is created using Java. The button move left, move right, move up, and move down linked to Snowman may be handled by this application or project.

This software shows movement of snowman in left right up and down using applet.

1.2. Constraints and Dependencies

This project's dependencies are as follows:

- A user with basic knowledge of system
- User should have knowledge about run command of applet.
- User should have basic knowledge of java

1.3. Project Life Cycle

The software development model used in making of this project is a spiral model which can be explained as: -

A Spiral model is an evolutionary software process paradigm that blends the iterative nature of prototyping with the controlled and systematic components of the linear sequential model. It has the ability to quickly create new software versions. The spiral process is used to create the software in a succession of incremental releases. During the early stages, the extra release may be a paper model or prototype. Later cycles yield progressively complete versions of the desired system.

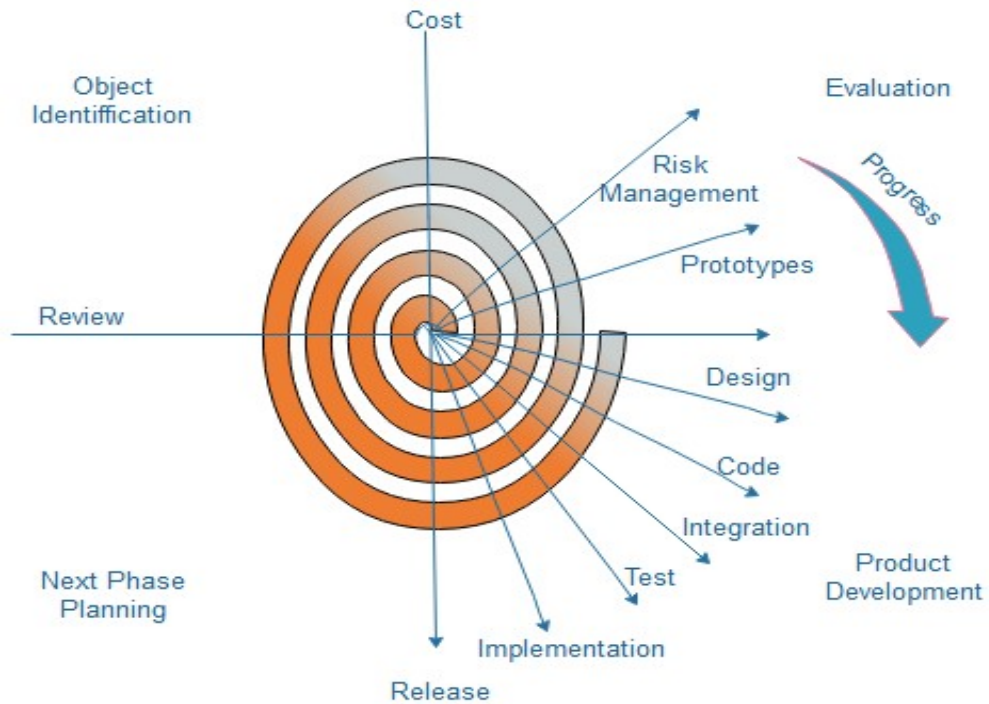


Fig 1.1: Spiral Model

1.3.1. Advantages

The following are the benefits of employing this model:

1. A significant amount of risk analysis
2. This is useful for large, mission-critical initiatives.

1.3.2. Disadvantages

The disadvantages of employing this model are as follows:

1. It might be an expensive model to utilize.
2. Risk analysis need extremely specialized knowledge.
3. It is ineffective for smaller tasks.

Chapter 2

Requirement Analysis

2.1. Functional requirement

The functional requirements of the project are: -

1. Whenever a user run program a snowman is draw in applet.
2. He/she can move the snowman by using button.
3. By button reset he/she draw original(re-position) snowman.

2.2. Non-Functional requirements

The non-functional requirements of the project are: -

1. The processing of each request should be done within 10 seconds.
2. Data fetching and manipulation should be fast.

2.3. Technology Used

The technology used in making of this project are: -

1. JAVA: -

Java is a high-level, class-based, object-oriented programming language with a low number of implementation dependencies. It is a general-purpose programming language designed to allow programmers to write once and run anywhere (WORA), which means that generated Java code may run on any platform that supports Java without the need for recompilation. Java programs are often compiled to bytecode that can run on any Java virtual machine (JVM), independent of computer architecture. Java was originally developed by James Gosling at Sun Microsystems. It was released in May 1995 as a core component of Sun Microsystems' Java platform. The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses.

2. JAVA Applet: -

Java applets were small applications written in the Java programming language, or another programming language that compiles to Java bytecode, and delivered to users in the form of Java bytecode. The user launched the Java applet from a web page, and the applet was then executed within a Java virtual machine (JVM) in a process separate from the web browser itself. A Java applet could appear in a frame of the web page, a new application window, Sun's AppletViewer, or a stand-alone tool for testing applets. Java applets run at very fast speeds and until 2011, they were many times faster than JavaScript. Unlike JavaScript, Java applets had access to 3D hardware acceleration, making them well-suited for non-trivial, computation-intensive visualizations. As browsers have gained support for hardware-accelerated graphics thanks to the canvas technology (or specifically WebGL in the case of 3D graphics), as well as just-in-time compiled JavaScript, the speed difference has become less noticeable.

The applets are used to provide interactive features to web applications that cannot be provided by HTML alone. They can capture mouse input and also have controls like buttons or check boxes. In response to user actions, an applet can change the provided graphic content. This makes applets well-suited for demonstration, visualization, and teaching. There are online applet collections for studying various subjects, from physics to heart physiology.

An applet can also be a text area only; providing, for instance, a cross-platform command-line interface to some remote system. If needed, an applet can leave the dedicated area and run as a separate window. However, applets have very little control over web page content outside the applet's dedicated area, so they are less useful for improving the site appearance in general, unlike other types of browser extensions (while applets like news tickers or WYSIWYG editors are also known). Applets can also play media in formats that are not natively supported by the browser.

Pages coded in HTML may embed parameters within them that are passed to the

applet. Because of this, the same applet may have a different appearance depending on the parameters that were passed.

As applets were available before HTML5, modern CSS and JavaScript interface DOM were standard, they were also widely used for trivial effects such as mouse over and navigation buttons. This approach, which posed major problems for accessibility and misused system resources, is no longer in use and was strongly discouraged even at the time.

3. JAVA Swing: -

Swing is a Java GUI widget toolkit. It's a component of Oracle's Java Foundation Classes (JFC), which provides an API for creating graphical user interfaces (GUIs) for Java programs.

Swing was created to give a more comprehensive collection of graphical user interface components than the previous Abstract Window Toolkit (AWT). Swing offers a pluggable look and feel that allows applications to have a look and feel that is unconnected to the underlying platform, as well as a look and feel that emulates the look and feel of numerous platforms. It includes more powerful and adaptable components than AWT. Swing features various complex components, such as tabbed panels, scroll panes, trees, tables, and lists, in addition to traditional components such as buttons, check boxes, and labels.

4. IntelliJ IDEA: -

IntelliJ IDEA is an integrated development environment (IDE) written in Java for developing computer software written in Java, Kotlin, Groovy, and other JVM-based languages. It is developed by JetBrains (formerly known as IntelliJ) and is available as an Apache 2 Licensed community edition and in a proprietary commercial edition. Both can be used for commercial development.



System Requirements: -

Table 2.1: - System requirements for IntelliJ IDEA

	Windows	macOS	Linux
OS Version	64 Bit Microsoft Windows 8 or later	macOS 10.13 or later	Any Linux distribution that supports Gnome, KDE, or Unity DE
RAM	2 GB RAM minimum, 8 GB RAM recommended		
Disk space	2.5 GB and another 1 GB for caches minimum, solid-state drive with at least 5 GB of free space recommended		
JDK Version	Add support for Java 16		
JRE Version	JRE 11 is bundled.		
Screen resolution	1024×768 minimum screen resolution. 1920×1080 is a recommended screen resolution.		

Chapter 3

Design

2.1. DFD (DATA FLOW DIAGRAM)

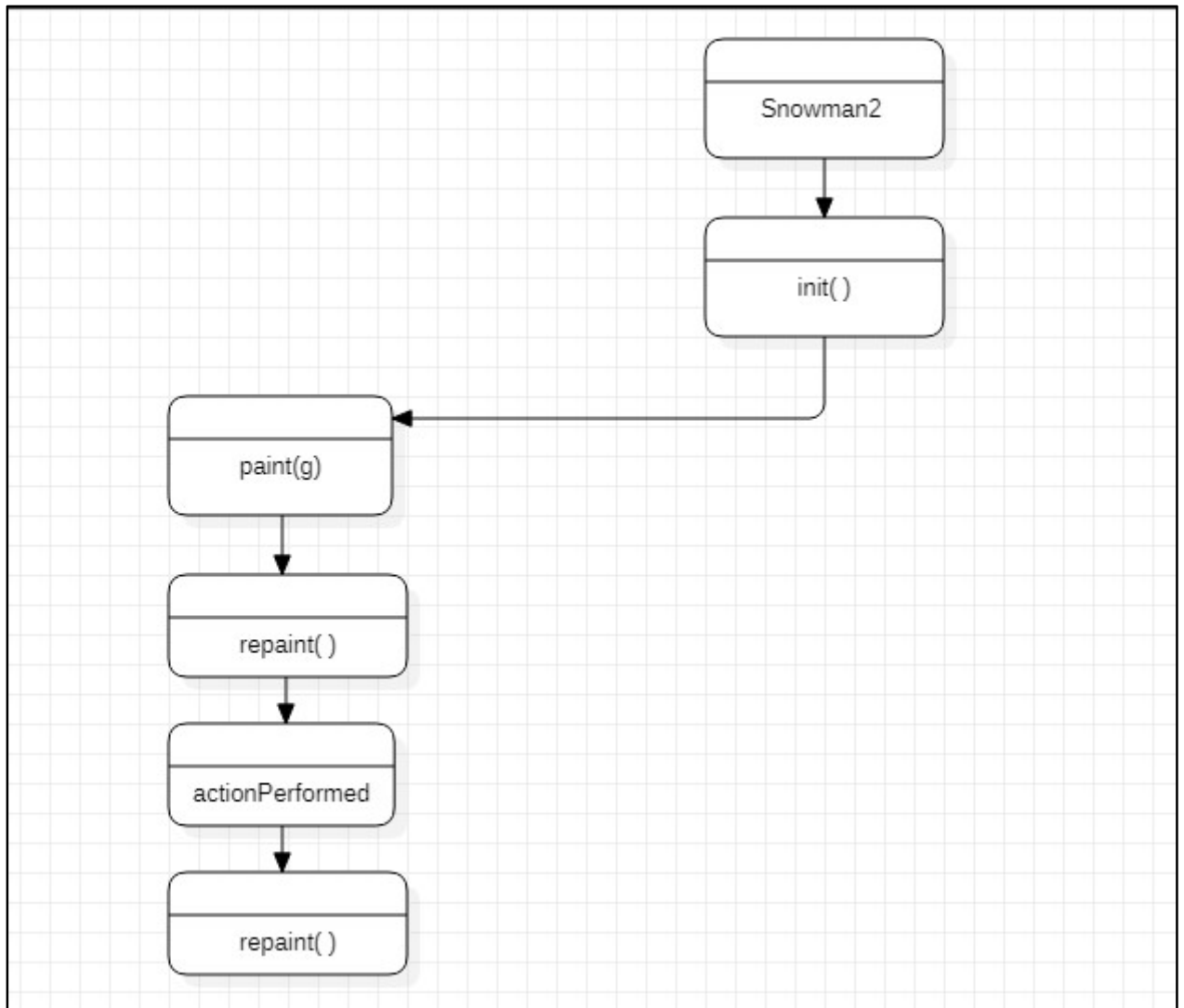


Fig 3.1. DFD

2.2. Use Case Diagram

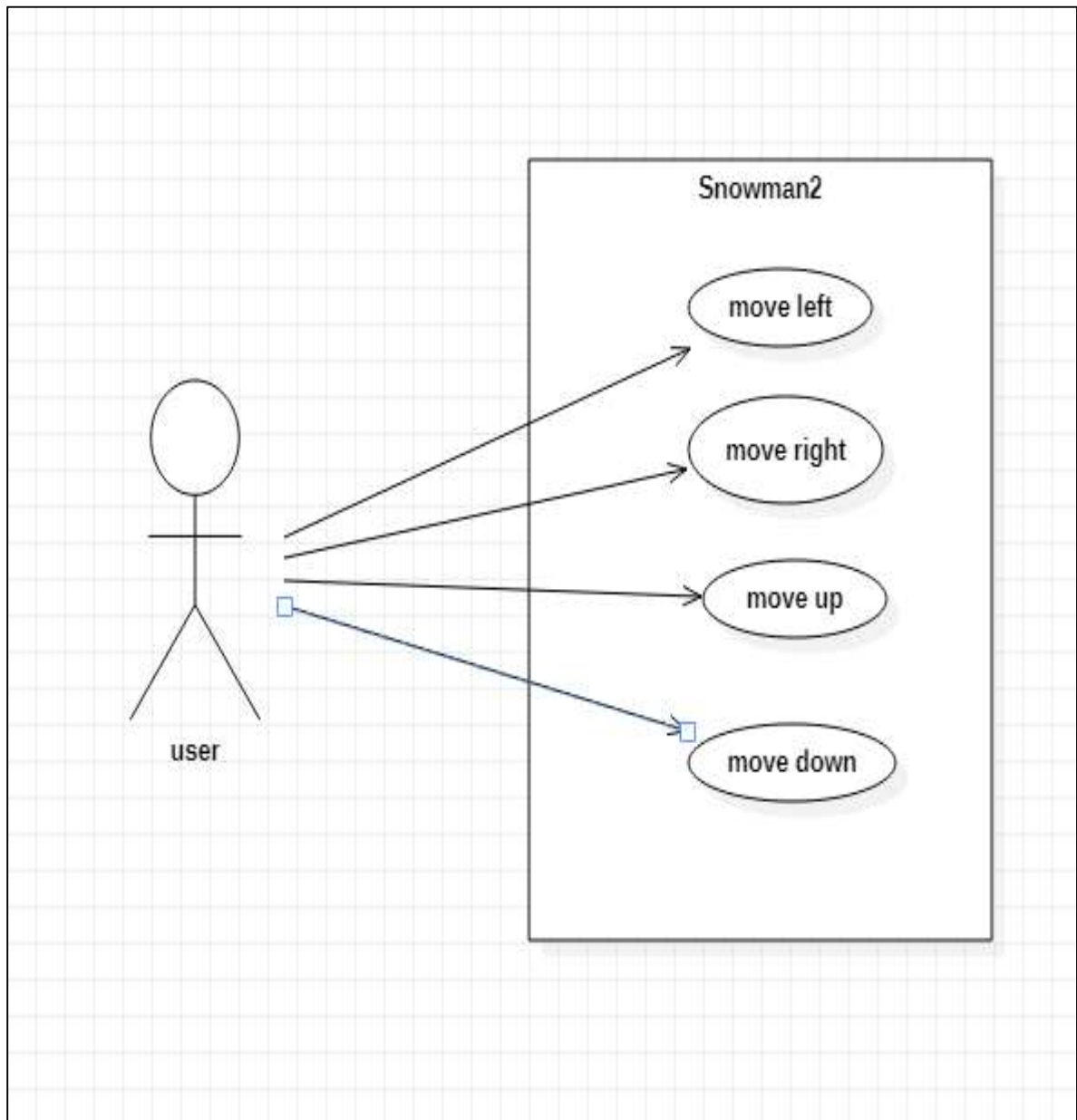


Fig 3.2. USE CASE DIAGRAM

2.3. Class Diagram

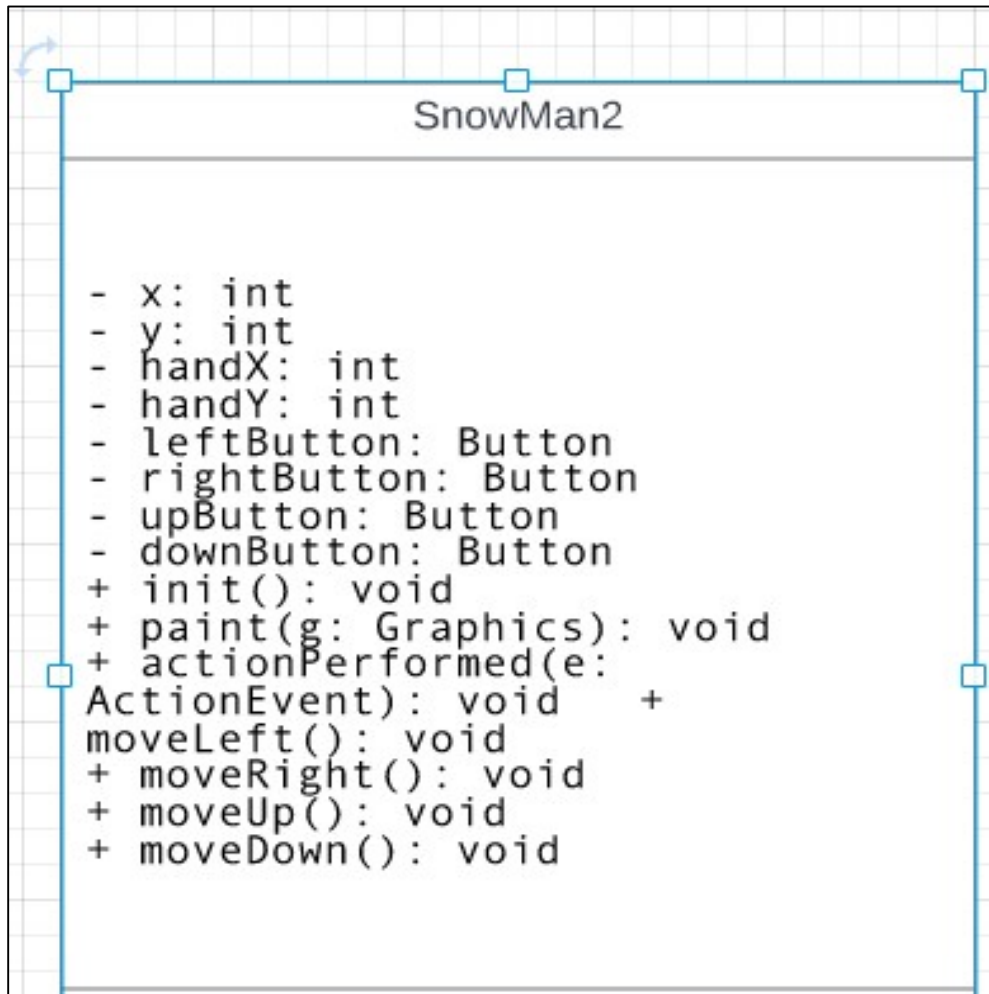


Fig 3.3. CLASS DIAGRAM

Chapter 4

Conclusion

To summarize this report on the project entitled **SNOWMAN**, it is capable of draw and handling position of snowman. Snowman is design with the help of applet in java, both java code and applet code are in one file. There are 5 buttons in it which change the position of snowman acc. To which button pressed. The button are move left, move right ,move up ,move down and reset. As move left button press the snow man move left side if right then it move right side to current Position as same up an down button work ,there is another button reset which set the snowman at its original position. To run the project, the person should know the command to run applet. As the UI is clean and easy making it good application for non IT background users to use it.

Chapter 5

References

- <https://www.wikipedia.org/>
- <https://www.google.co.in/>
- <https://www.javatpoint.com/>
- <https://www.geeksforgeeks.org/>
- <https://stackoverflow.com/>

Chapter 6

Snapshots of Project

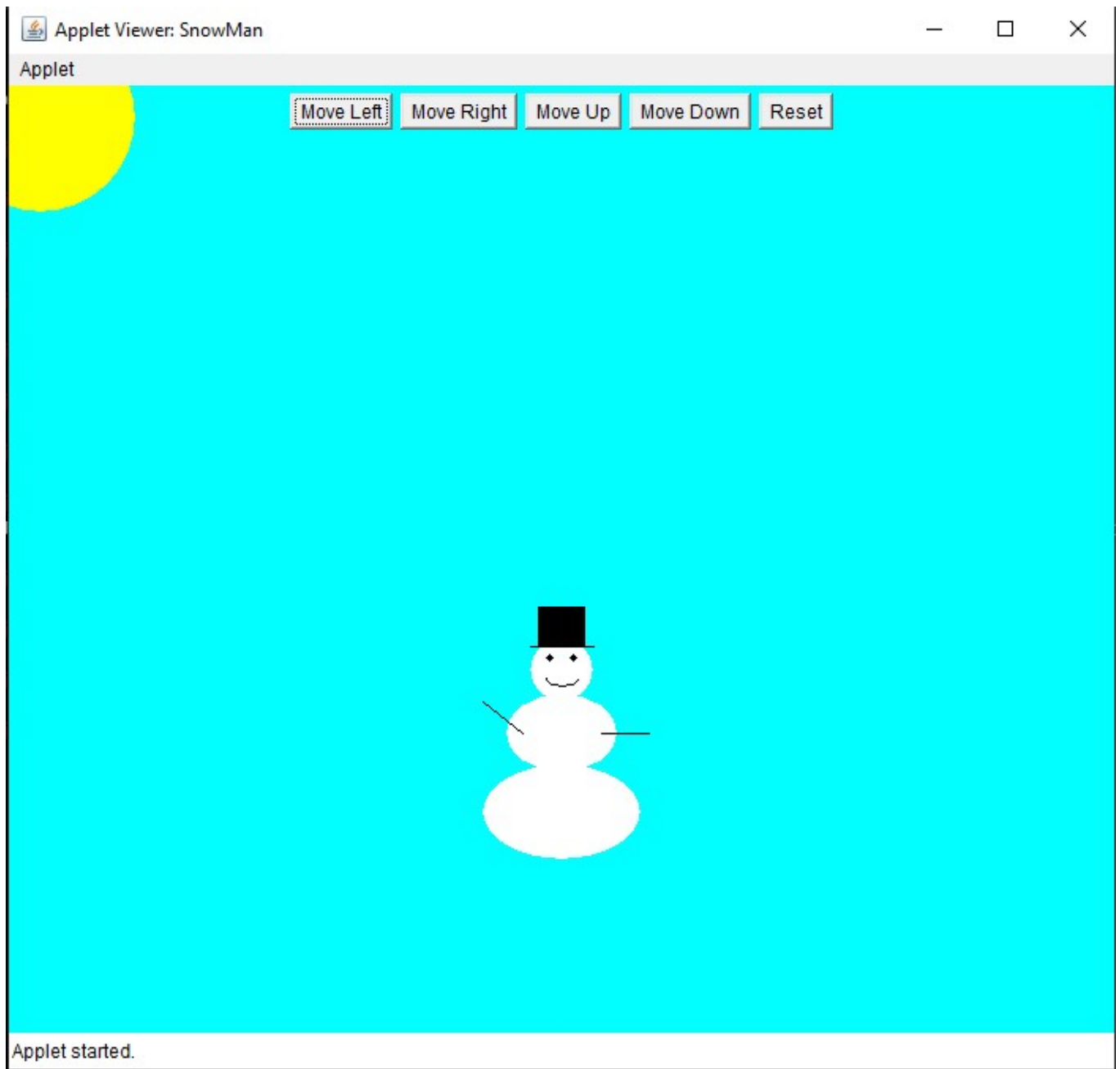


Fig 5.1. Original position

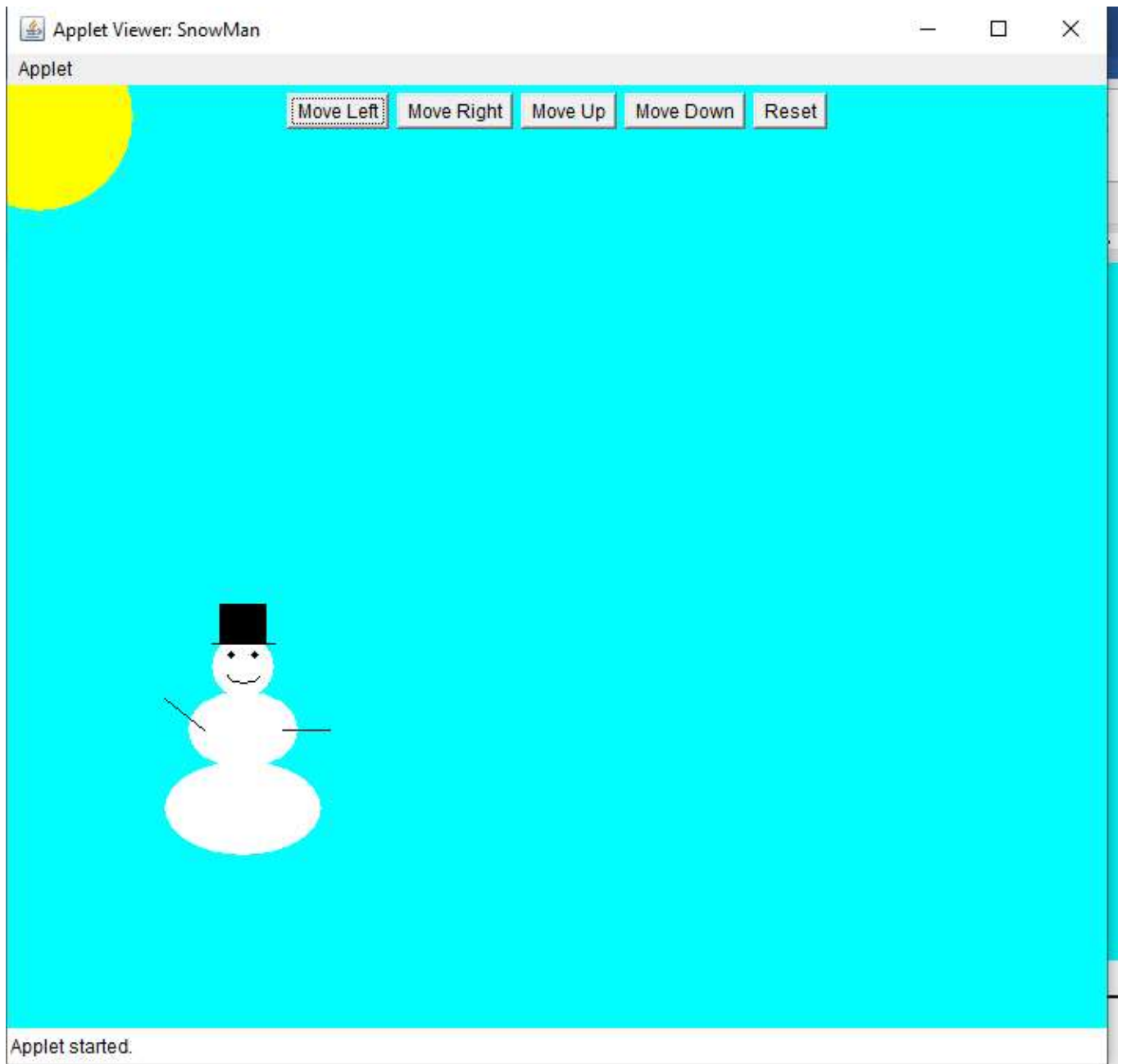


Fig 5.2. Move Left

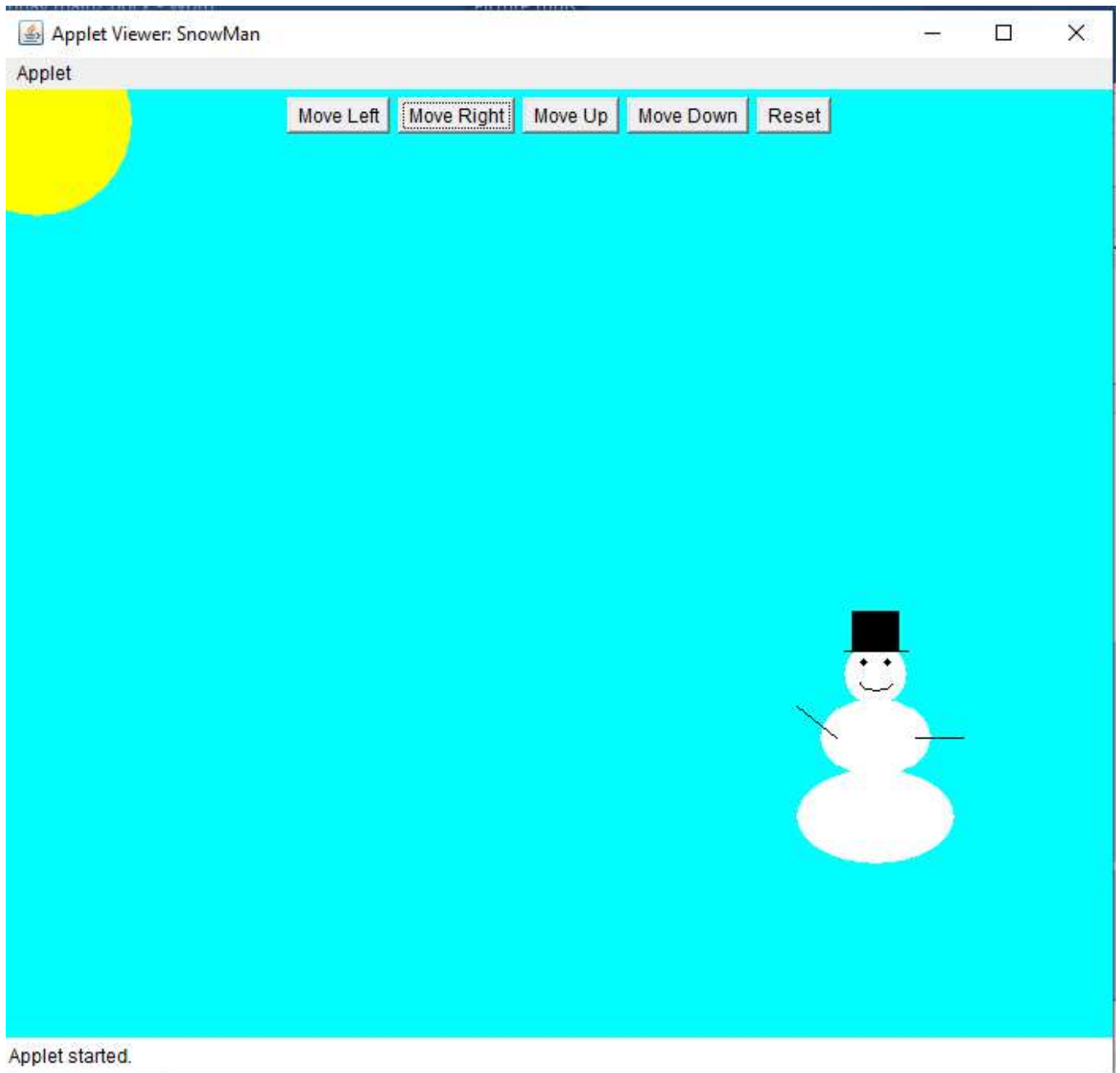


Fig 5.3. Move Right

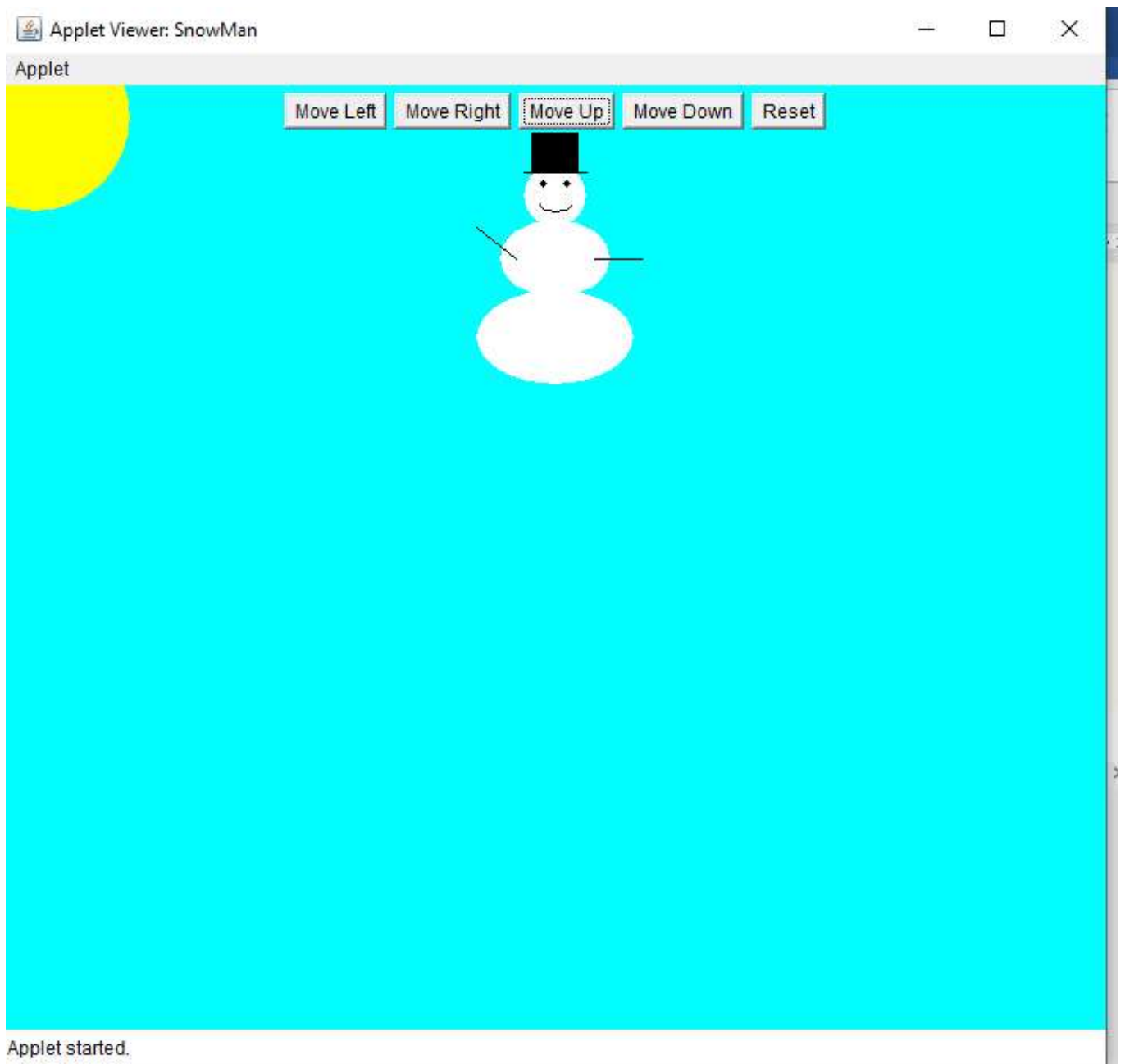


Fig 5.4. Move Up

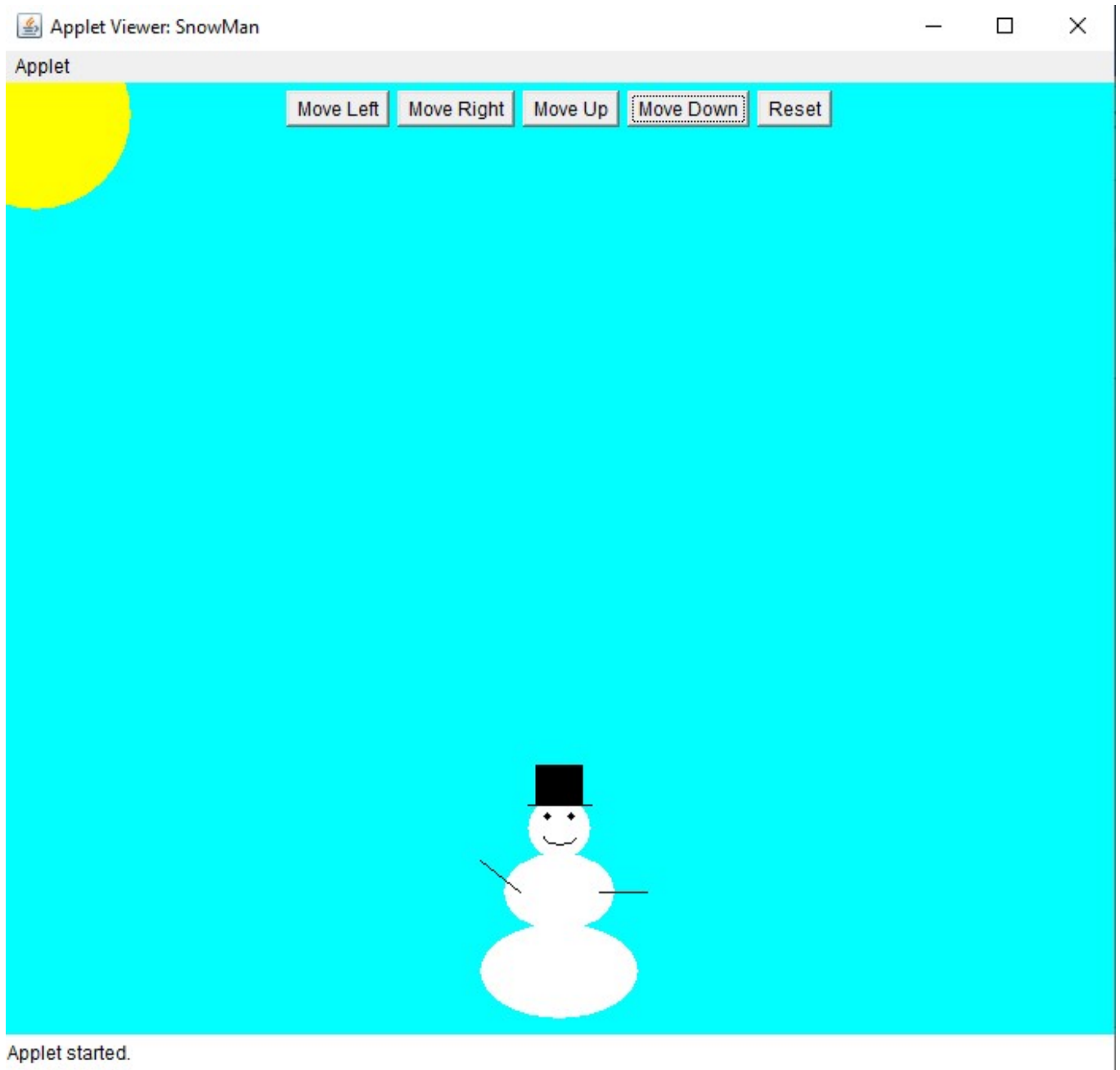


Fig 5.5. Move Down

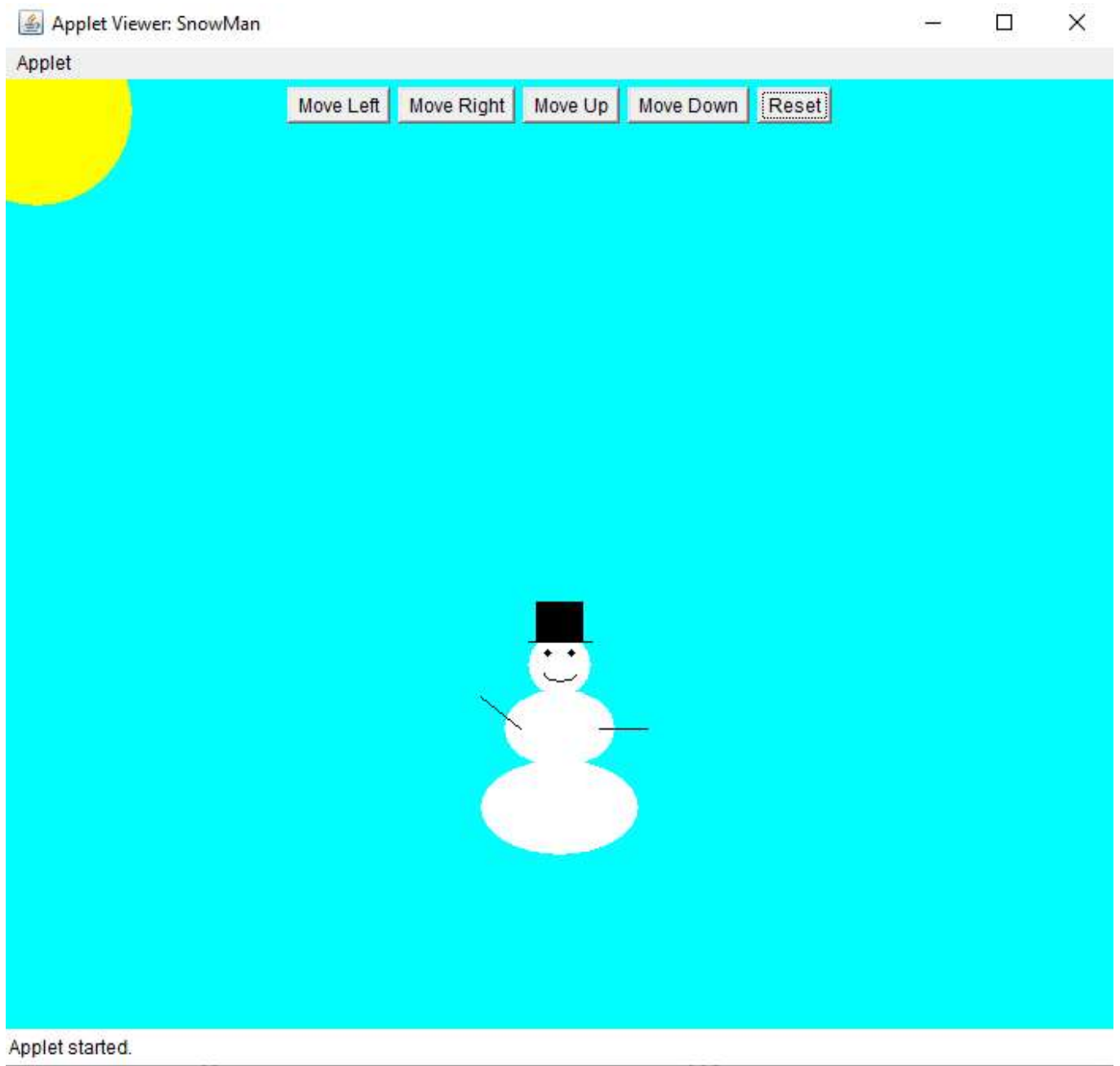
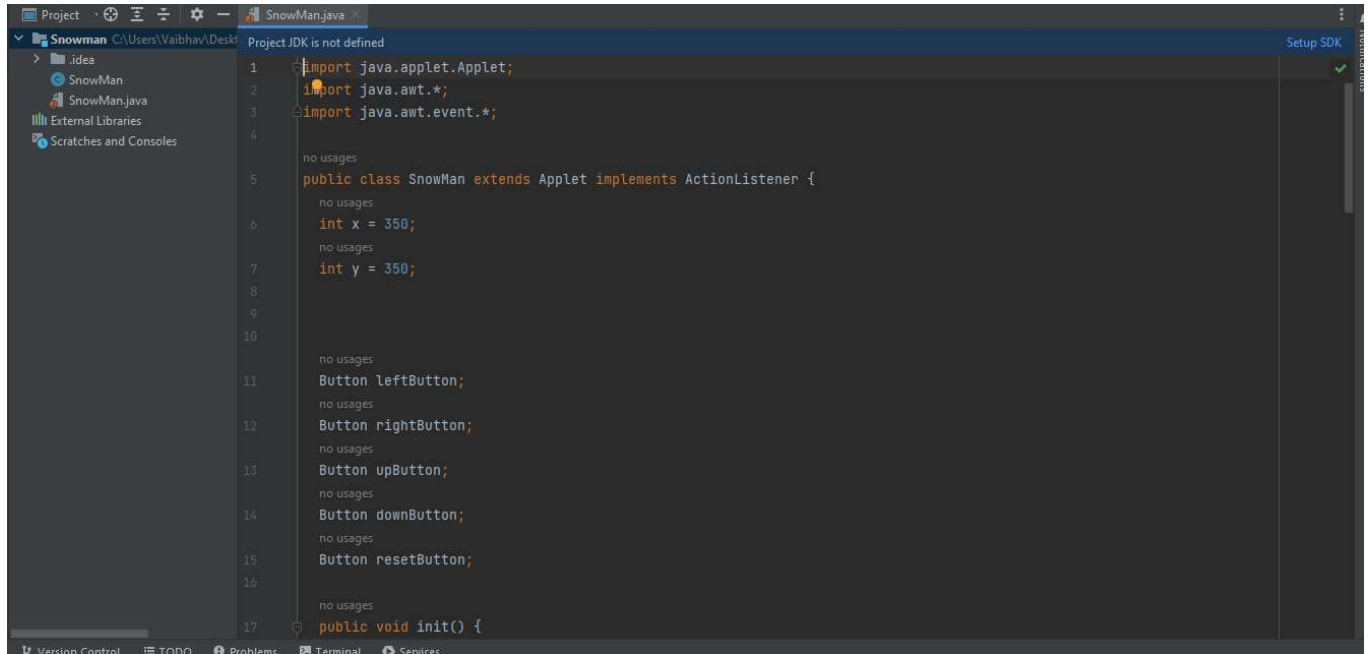


Fig 5.6. Reset

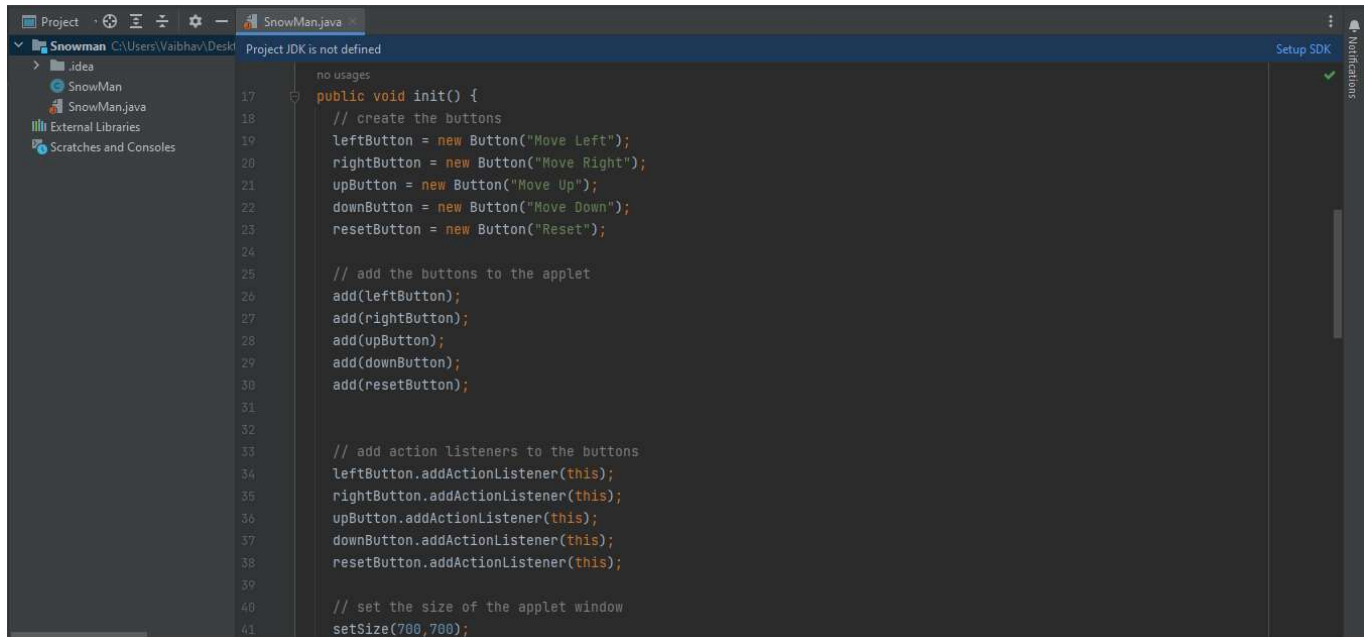
Chapter 7

Code of Project



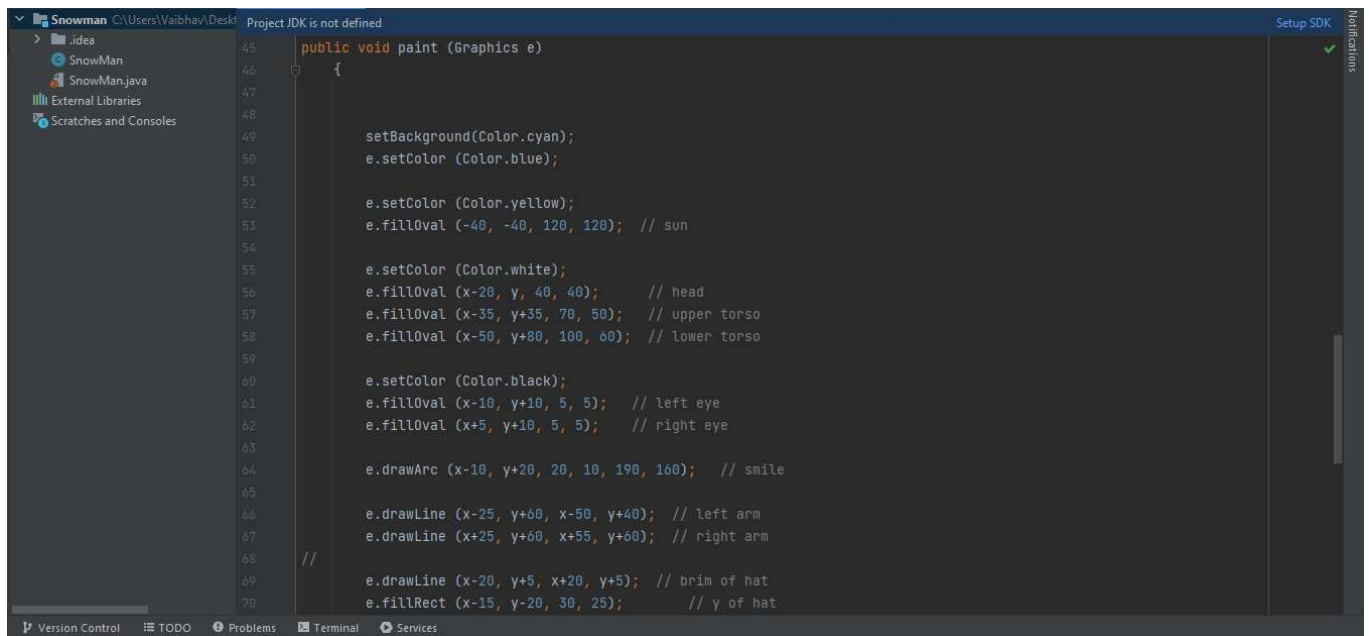
```
1 import java.applet.Applet;
2 import java.awt.*;
3 import java.awt.event.*;
4
5 no usages
6 public class SnowMan extends Applet implements ActionListener {
7     no usages
8     int x = 350;
9     no usages
10    int y = 350;
11
12    no usages
13    Button leftButton;
14    no usages
15    Button rightButton;
16    no usages
17    Button upButton;
18    no usages
19    Button downButton;
20    no usages
21    Button resetButton;
22
23    no usages
24    public void init() {
```

Fig 6.1. CODE SCREENSHOT



```
17 public void init() {
18     // create the buttons
19     leftButton = new Button("Move Left");
20     rightButton = new Button("Move Right");
21     upButton = new Button("Move Up");
22     downButton = new Button("Move Down");
23     resetButton = new Button("Reset");
24
25     // add the buttons to the applet
26     add(leftButton);
27     add(rightButton);
28     add(upButton);
29     add(downButton);
30     add(resetButton);
31
32
33     // add action listeners to the buttons
34     leftButton.addActionListener(this);
35     rightButton.addActionListener(this);
36     upButton.addActionListener(this);
37     downButton.addActionListener(this);
38     resetButton.addActionListener(this);
39
40     // set the size of the applet window
41     setSize(700,700);
```

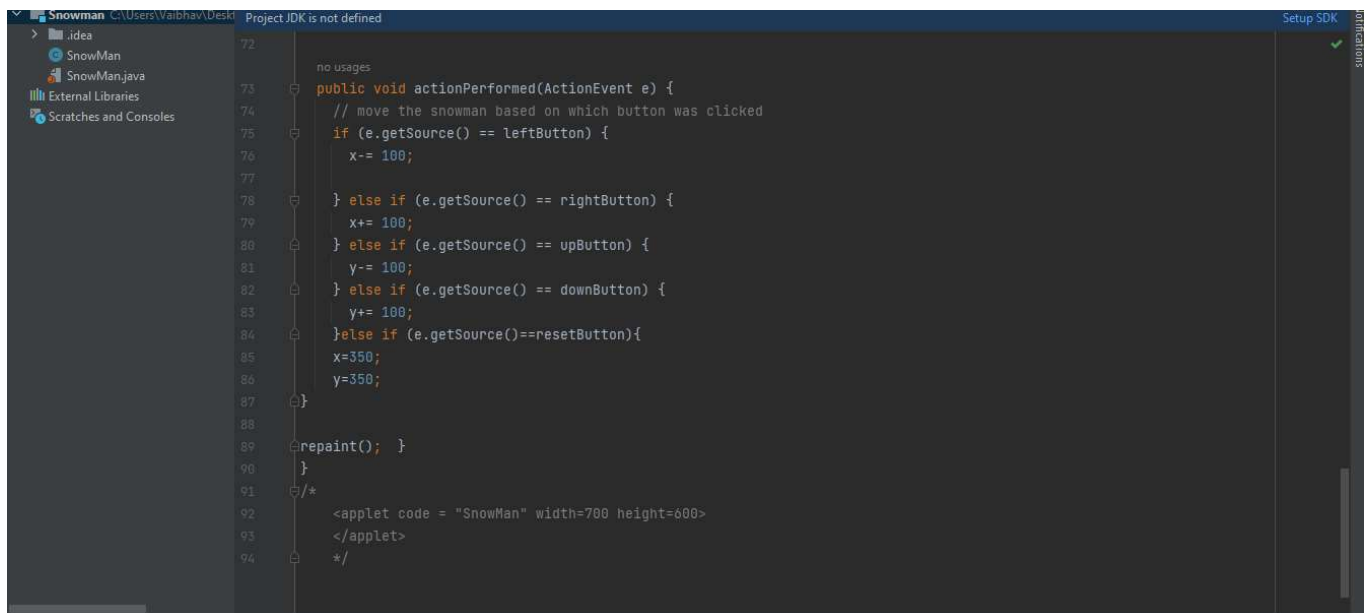
Fig 6.2. CODE SCREENSHOT



The screenshot shows the IntelliJ IDEA IDE with a project named 'Snowman'. The 'SnowMan.java' file is open, displaying the 'paint' method. The code is as follows:

```
45 public void paint (Graphics e)
46 {
47
48
49     setBackground(Color.cyan);
50     e.setColor (Color.blue);
51
52     e.setColor (Color.yellow);
53     e.fillOval (-40, -40, 120, 120); // sun
54
55     e.setColor (Color.white);
56     e.fillOval (x-20, y, 40, 40); // head
57     e.fillOval (x-35, y+35, 70, 50); // upper torso
58     e.fillOval (x-50, y+80, 100, 60); // lower torso
59
60     e.setColor (Color.black);
61     e.fillOval (x-10, y+10, 5, 5); // left eye
62     e.fillOval (x+5, y+10, 5, 5); // right eye
63
64     e.drawArc (x-10, y+20, 20, 10, 190, 160); // smile
65
66     e.drawLine (x-25, y+60, x-50, y+40); // left arm
67     e.drawLine (x+25, y+60, x+55, y+60); // right arm
68
69     //
70     e.drawLine (x-20, y+5, x+20, y+5); // brim of hat
71     e.fillRect (x-15, y-20, 30, 25); // y of hat
```

Fig 6.3. CODE SCREENSHOT



The screenshot shows the IntelliJ IDEA IDE with the same 'Snowman' project. The 'SnowMan.java' file is open, displaying the 'actionPerformed' method. The code is as follows:

```
72 no usages
73 public void actionPerformed(ActionEvent e) {
74     // move the snowman based on which button was clicked
75     if (e.getSource() == leftButton) {
76         x-= 100;
77
78     } else if (e.getSource() == rightButton) {
79         x+= 100;
80     } else if (e.getSource() == upButton) {
81         y-= 100;
82     } else if (e.getSource() == downButton) {
83         y+= 100;
84     } else if (e.getSource() == resetButton) {
85         x=350;
86         y=350;
87     }
88
89     repaint(); }
90 }
91 /*
92 <applet code = "SnowMan" width=700 height=600>
93 </applet>
94 */
```

Fig 6.4. CODE SCREENSHOT