3-D TIC TAC TOE

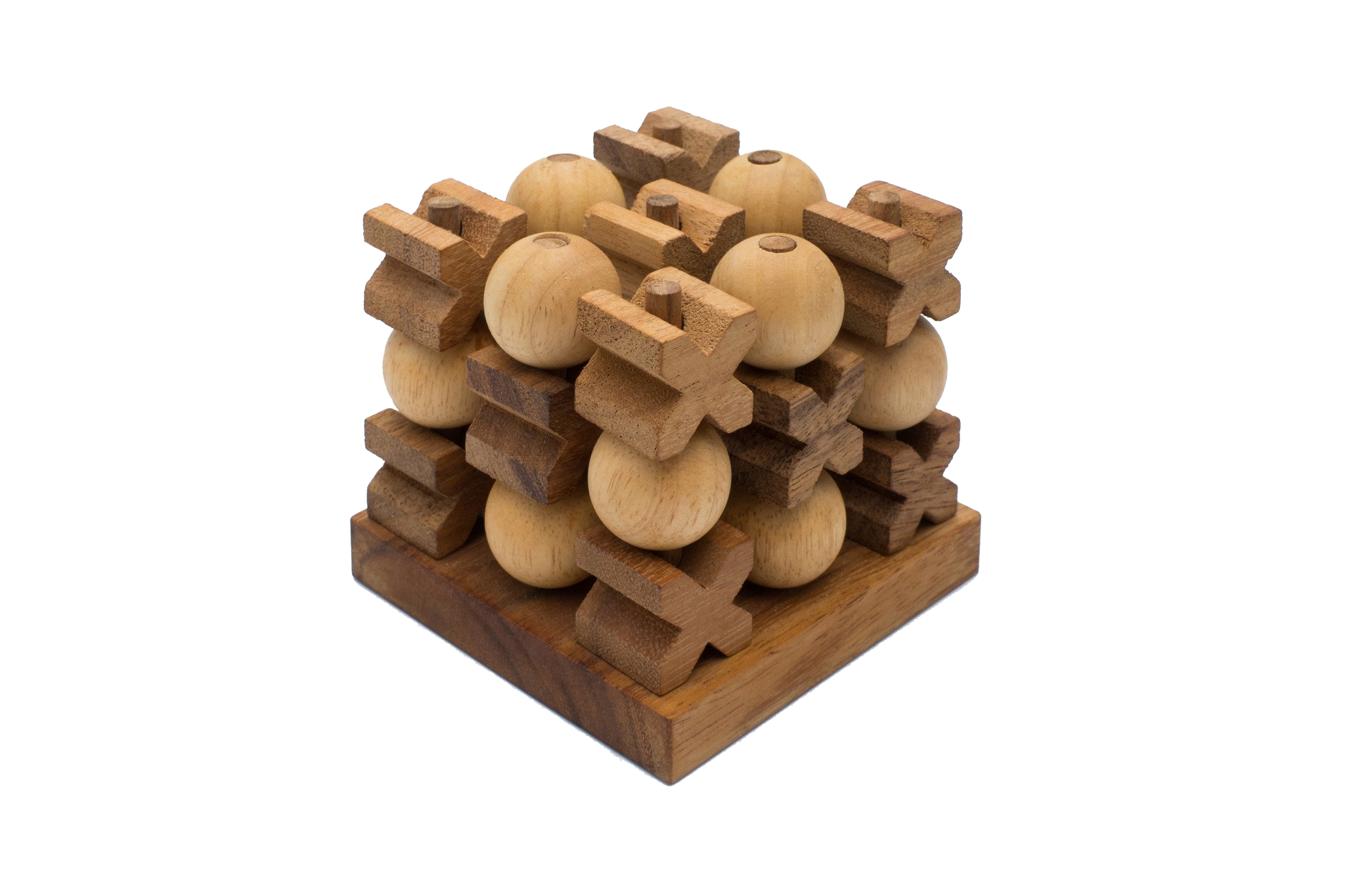
3-D TIC TAC TOE

PROGRAM DEVELOPEMENT PROJECT

2017

S.Y. B.Tech. I.T.

11/14/2017



1. JAYANT RANE (161080023)
2. SIDDHESH PATEL (161080018)
3. NIKHIL DHARASKAR (161080022)
4. YUVRAJ PAYIL (161050002)

**Abstract:**

To create a two player 3-D TIC TAC TOE game in Java.

**Softwares used:**

• Eclipse

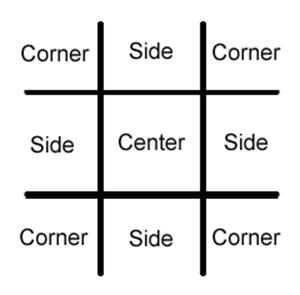
• NetBeans IDE: 8.2

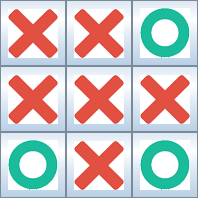
**How to play ?**

The game is basically the same as regular tic-tac-toe: you try to win by getting three X’s or O’s in a row. The trick is, instead of one grid of nine squares, you have three grids of nine squares, all stacked on top of each other. A diagonal, horizontal, or vertical set of three lets you win, but since the game is 3-D, there are many more winning possibilities. After completing cube of 3\*3\*3, four new surfaces are formed from which winner of each face can be decided. Player winning maximum faces out of 7 wins the game.

**Basic structre:**

The element of each layer contributes to front faces as follows.





Three layers like above are arranged one above the other to realise 3d XO game.

**Concepts Of Java Used :**

* **java.awt.event :**

## What is an Event?

Change in the state of an object is known as event i.e. event describes the change in state of source. Events are generated as result of user interaction with the graphical user interface components. For example, clicking on a button, moving the mouse, entering a character through keyboard,selecting an item from list, scrolling the page are the activities that causes an event to happen.

**Types of Event :**

The events can be broadly classified into two categories:

* **Foreground Events** - Those events which require the direct interaction of user.They are generated as consequences of a person interacting with the graphical components in Graphical User Interface. For example, clicking on a button, moving the mouse, entering a character through keyboard,selecting an item from list, scrolling the page etc.
* **Background Events** - Those events that require the interaction of end user are known as background events. Operating system interrupts, hardware or software failure, timer expires, an operation completion are the example of background events.

**What is Event Handling?**

Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism have the code which is known as event handler that is executed when an event occurs. Java Uses the Delegation Event Model to handle the events. This model defines the standard mechanism to generate and handle the events. Let's have a brief introduction to this model.

The Delegation Event Model has the following key participants namely:

* **Source** - The source is an object on which event occurs. Source is responsible for providing information of the occurred event to it's handler. Java provide as with classes for source object.
* **Listener** - It is also known as event handler. Listener is responsible for generating response to an event. From java implementation point of view the listener is also an object. Listener waits until it receives an event. Once the event is received , the listener process the event an then returns.

The benefit of this approach is that the user interface logic is completely separated from the logic that generates the event. The user interface element is able to delegate the processing of an event to the separate piece of code. In this model ,Listener needs to be registered with the source object so that the listener can receive the event notification. This is an efficient way of handling the event because the event notifications are sent only to those listener that want to receive them.

**Steps involved in event handling:**

* The User clicks the button and the event is generated.
* Now the object of concerned event class is created automatically and information about the source and the event get populated with in same object.
* Event object is forwarded to the method of registered listener class.
* the method is now get executed and returns.
* **javax.swing :**

**Java Swing**  is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

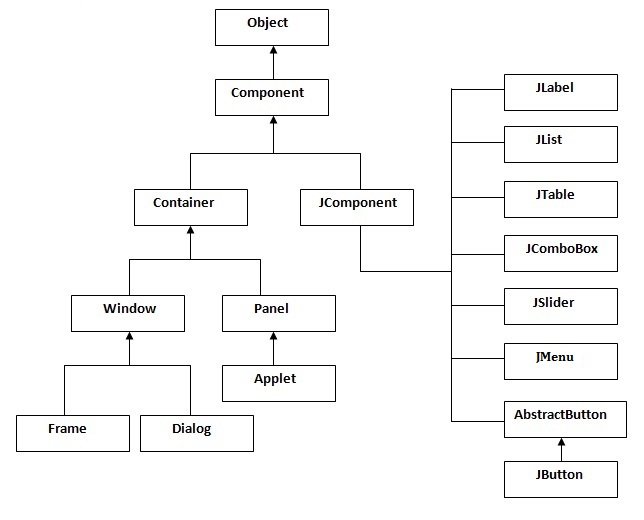
Unlike AWT, Java Swing provides platform-independent and lightweight components.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

**Difference between AWT and Swing:**

|  |  |  |
| --- | --- | --- |
| No. | Java AWT | Java Swing |
| 1) | AWT components are **platform-dependent**. | Java swing components are **platform-independent**. |
| 2) | AWT components are **heavyweight**. | Swing components are **lightweight**. |
| 3) | AWT **doesn't support pluggable look and feel**. | Swing **supports pluggable look and feel**. |
| 4) | AWT provides **less components** than Swing. | Swing provides **more powerful components**such as tables, lists, scrollpanes, colorchooser, tabbedpane etc. |
| 5) | AWT **doesn't follows MVC**(Model View Controller) where model represents data, view represents presentation and controller acts as an interface between model and view. | Swing **follows MVC**. |

### Hierarchy of Java Swing classes:



**Commonly used Methods of Component classd:**

The methods of Component class are widely used in java swing that are given below.

|  |  |
| --- | --- |
| Method | Description |
| public void add(Component c) | add a component on another component. |
| public void setSize(int width,int height) | sets size of the component. |
| public void setLayout(LayoutManager m) | sets the layout manager for the component. |
| public void setVisible(boolean b) | sets the visibility of the component. It is by default false. |

There are two ways to create a frame:

* By creating the object of Frame class (association)
* By extending Frame class (inheritance)

**MouseListener :**

The Java MouseListener is notified whenever you change the state of mouse. It is notified against MouseEvent. The MouseListener interface is found in java.awt.event package. It has five methods.

1. public abstract void mouseClicked(MouseEvent e);
2. public abstract void mouseEntered(MouseEvent e);
3. public abstract void mouseExited(MouseEvent e);
4. public abstract void mousePressed(MouseEvent e);
5. public abstract void mouseReleased(MouseEvent e);

**Functions used :**

public abstract void mouseClicked(MouseEvent e);

**JFrame :**

The javax.swing.JFrame class is a type of container which inherits the java.awt.Frame class. JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI.

Unlike Frame, JFrame has the option to hide or close the window with the help of setDefaultCloseOperation(int) method.

**Functions used :**

**JPanel :**

The JPanel is a simplest container class. It provides space in which an application can attach any other component. It inherits the JComponents class.

It doesn't have title bar.

**Functions used :**

jPanel.add(component);

jPanel.setLayout();

**JButton :**

The JButton class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed. It inherits AbstractButton class.

### Commonly used Methods of AbstractButton class:

|  |  |
| --- | --- |
| Methods | Description |
| void setText(String s) | It is used to set specified text on button |
| String getText() | It is used to return the text of the button. |
| void setEnabled(boolean b) | It is used to enable or disable the button. |
| void setIcon(Icon b) | It is used to set the specified Icon on the button. |
| Icon getIcon() | It is used to get the Icon of the button. |
| void setMnemonic(int a) | It is used to set the mnemonic on the button. |
| void addActionListener(ActionListener a) | It is used to add the action listener to this object. |

**Functions used :**

void setIcon(Icon b)

Icon getIcon()

# Java JLabel :

The object of JLabel class is a component for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly. It inherits JComponent class.

### Commonly used Methods:

|  |  |
| --- | --- |
| Methods | Description |
| String getText() | t returns the text string that a label displays. |
| void setText(String text) | It defines the single line of text this component will display. |
| void setHorizontalAlignment(int alignment) | It sets the alignment of the label's contents along the X axis. |
| Icon getIcon() | It returns the graphic image that the label displays. |
| int getHorizontalAlignment() | It returns the alignment of the label's contents along the X axis. |

**Functions used :**

# void setText(String text)

# Java GridLayout :

The GridLayout is used to arrange the components in rectangular grid. One component is displayed in each rectangle.

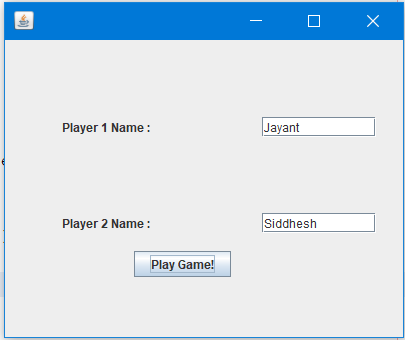
### Constructors of GridLayout class

1. **GridLayout():** creates a grid layout with one column per component in a row.
2. **jPanel1.setLayout(jPanel1Layout):** creates a grid layout with the given rows and columns but no gaps between the components.
3. **GridLayout(int rows, int columns, int hgap, int vgap):** creates a grid layout with the given rows and columns alongwith given horizontal and vertical gaps.

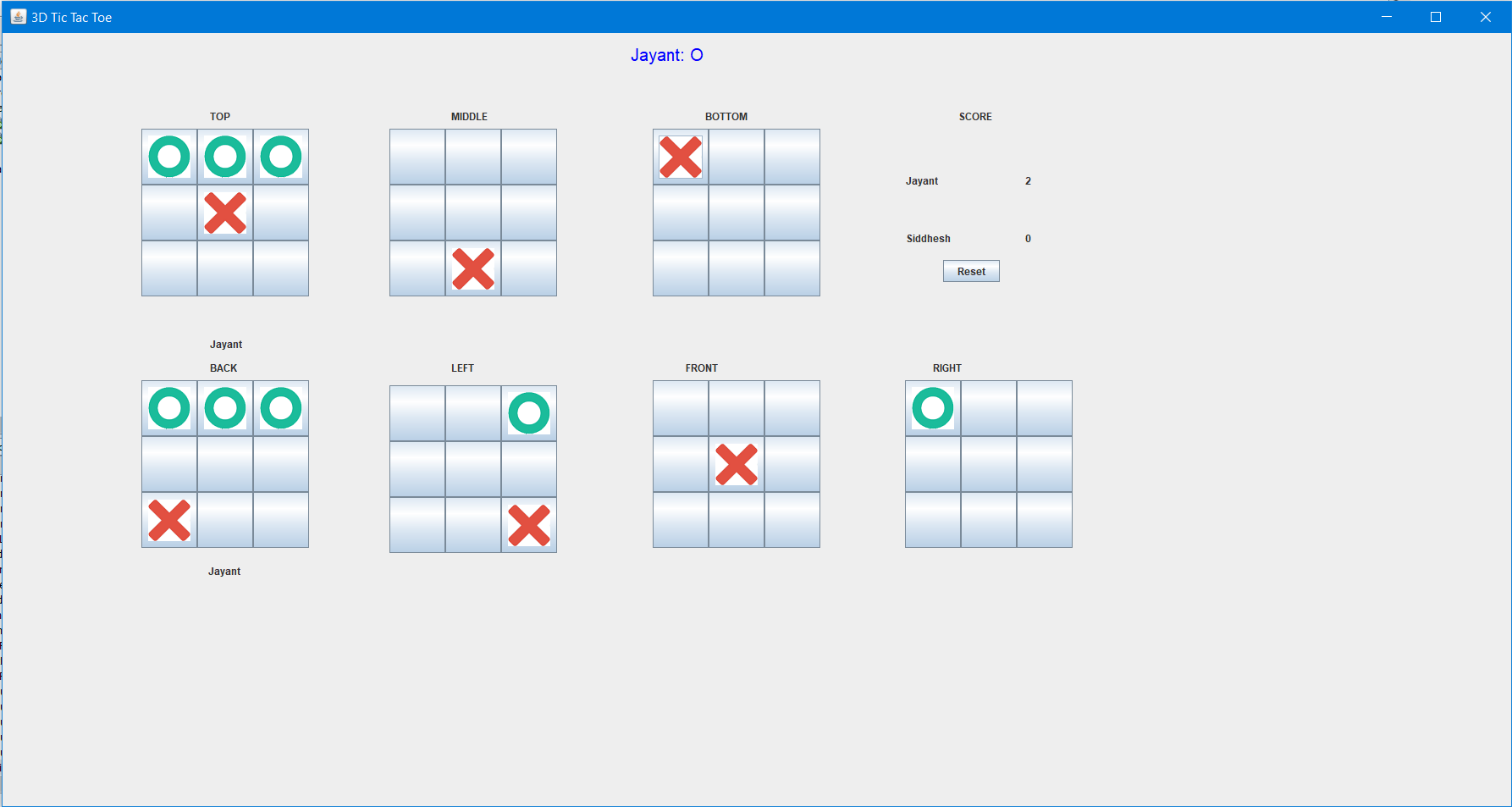
**Functions used :**

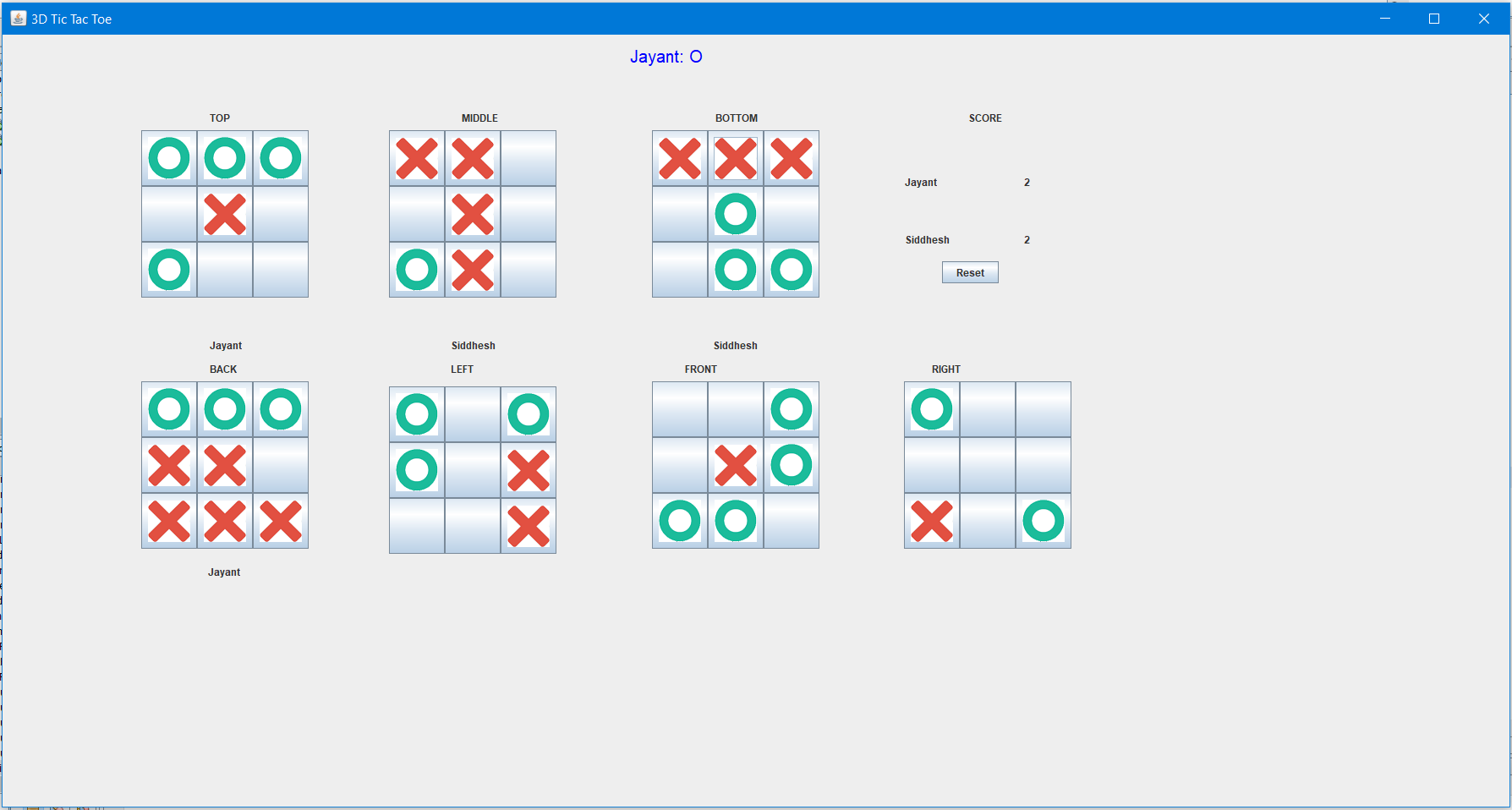
**jPanel1.setLayout(jPanel1Layout)**

**Screenshots:**

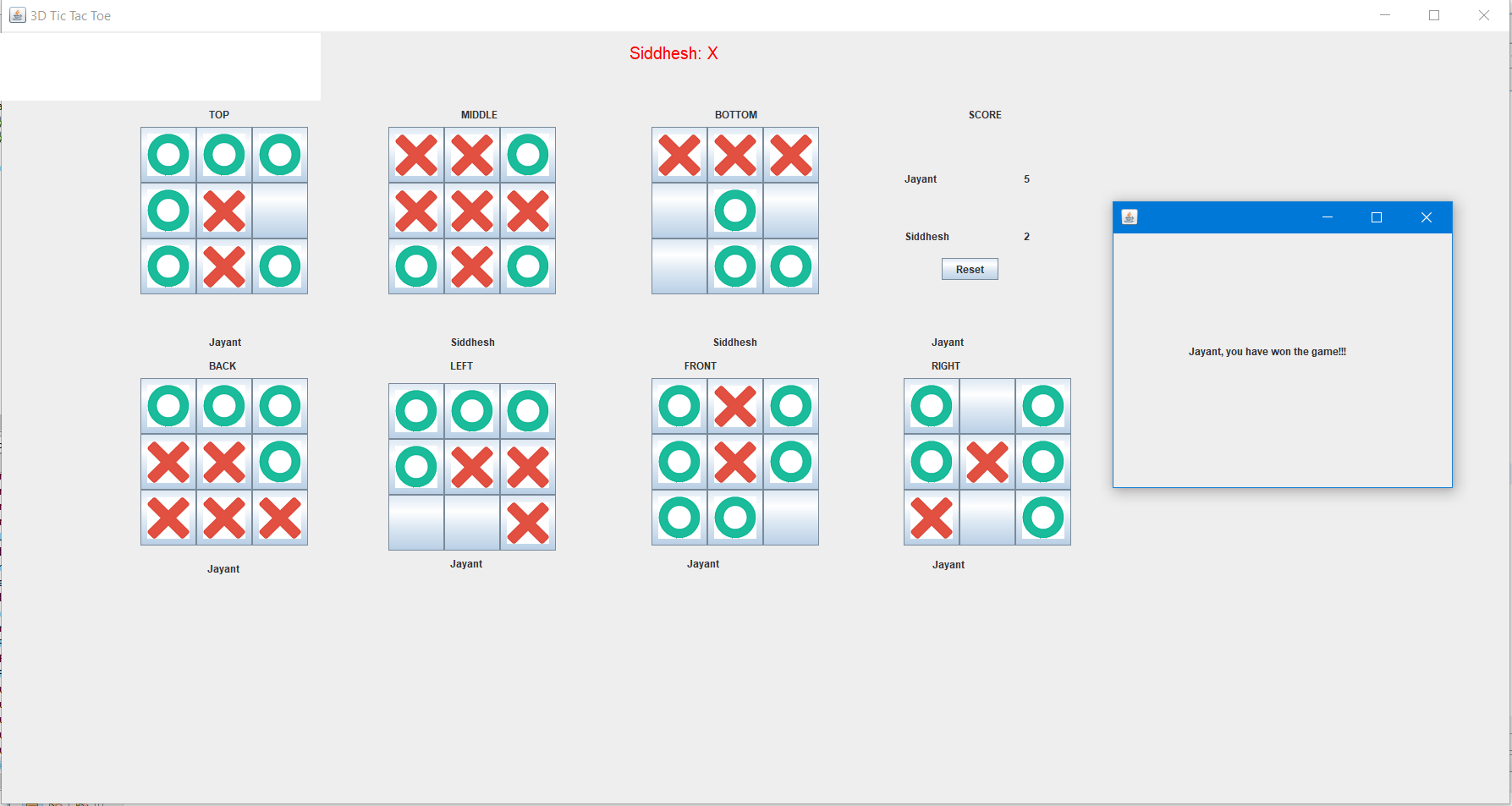


**Start Screen: Add player names**





**GamePlay: Two players playing according to their respective turns**



**Player winining maximum faces wins the game**

**Conclusion : Two player 3-D TIC TAC TOE game is successfully created using Java**

**s**