Write the NAME of one of the controller classes (or class that contains a controller). Copy and paste a code segment of the controller that calls the mutator of the model.

Controller class: Board.java

```
public void mouseClicked(MouseEvent e) {
                // TODO Auto-generated method stub
                //Keeps track of the current player
                try {
                int x0 = e.getX();
                int y0 = e.getY();
                int x = -1;
                int y = -1;

                //Getting the position of the player on the x - axis
                if (x0 <= 100) {
                    x = 0;
                } else if (x0 <= 200) {
                    x = 1;
                } else if (x0 <= 300) {
                    x = 2;
                }

                //Getting the position of the player on the y - axis
                if (y0 <= 100) {
                    y = 0;
                } else if (y0 <= 200) {
                    y = 1;
                } else if (y0 <= 300) {
                    y = 2;
                }

                if (moves[x][y] != 0)
                {
                    counter--;
                }
                else
                {
                    model.update(x,y, currentPlayer);
                }

            } catch (Exception e1) {
                // TODO Auto-generated catch block
```

```
                counter--;
            }
```

Write the NAME of the model class. Copy and paste a code segment of a mutator of the model that modifies data and also notifies view(s). Give me the name of mutator as well.

Model class: Model.java

Mutator: update(int x, int y, int player)

```
public void undo() {
            moves[currentMove[0]][currentMove[1]] = 0;

            // Set current move to equal previous move
            currentMove[0] = previousMove[0];
            currentMove[1] = previousMove[1];

            setChanged();
            notifyObservers();
      }

      public void update(int x, int y, int player) {
            previousMove[0] = currentMove[0];
            previousMove[1] = currentMove[1];
            currentMove[0] = x;
            currentMove[1] = y;

            moves[currentMove[0]][currentMove[1]] = player;

            // Check columns
            for (int i = 0; i < 3; i++) {
                  if (moves[x][i] != player) {
                        break;
                  }
                  if (i == 2) {
                        winner = player;
                  }
            }
```

```
        // Check rows
        for (int i = 0; i < 3; i++) {
              if (moves[i][y] != player)
                    break;
              if (i == 2) {
                    winner = player;
              }
        }

        // Check top right diagonal
        if (x == y) {
              for (int i = 0; i < 3; i++) {
                    if (moves[i][i] != player)
                          break;
                    if (i == 2) {
                          winner = player;
                    }
              }
        }
        if(x + y == 2){
         for(int i = 0; i < 3; i++){
              if(moves[i][2-i] != player)
                  break;
              if(i == 2){
               winner = player;
              }
         }
      }
        //Check top left diagonal

        setChanged();
        notifyObservers();
    }
```

Write the NAME of the view class. Copy and paste a code the notification method of the view and show me how the notification method paints the view using the data from the model.

View class: Board.java

```
public void paintComponent(Graphics g) {
            ...
            //Draw x and o//
            Font biggerFont = new Font("Arial", Font.PLAIN, 90);
            g2.setFont(biggerFont);
            for (int i = 0; i < 3; i++)
            {
                  for (int j = 0; j < 3; j++)
                  {
                        System.out.println(i + " " +  j + " " +
moves[i][j]);
                        if(moves[i][j] == PLAYER_ONE)
                        {

g2.setColor(this.formatter.formatXColor());
                              g2.drawString("X", i*100+10, j*100+90);
                        }
                        else if (moves[i][j] == PLAYER_TWO)
                        {

g2.setColor(this.formatter.formatOColor());
                              g2.drawString("O", i*100+10, j*100+90);
                        }
                  }
            }

      }


@Override
      public void update(Observable o, Object arg) {
            // TODO Auto-generated method stub
          moves = model.getMoves();
```

```
                repaint();
        }
```

Write the NAME of a strategy and copy the code.

Strategy: BoardFormatter.java

Write the name of two concrete strategies. (Just names required).

2 strategies: LightBoard.java, DarkBoard.java

Copy and paste the code segment where you create a concrete strategy and plug-in into the context program.

```
public class DarkBoard implements BoardFormatter{

    @Override
    public Color formatLineColor() {
        // TODO Auto-generated method stub
        return Color.WHITE;
    }

    @Override
    public Color formatBoardColor() {
        // TODO Auto-generated method stub
        return Color.BLACK;
    }

    @Override
    public Color formatXColor() {
        // TODO Auto-generated method stub
        return Color.CYAN;
    }

    @Override
    public Color formatOColor() {
        // TODO Auto-generated method stub
        return Color.PINK;
    }

}
```

```
public class LightBoard implements BoardFormatter {

    @Override
    public Color formatLineColor() {
        // TODO Auto-generated method stub
        return Color.BLACK;
    }

    @Override
    public Color formatBoardColor() {
        // TODO Auto-generated method stub
        return Color.WHITE;
    }

    @Override
    public Color formatXColor() {
        // TODO Auto-generated method stub
        return Color.BLUE;
    }

    @Override
    public Color formatOColor() {
        // TODO Auto-generated method stub
        return Color.MAGENTA;
    }

}

public void paintComponent(Graphics g) {
            Graphics2D g2 = (Graphics2D) g;
            Rectangle2D.Double boardSize = new
Rectangle2D.Double(0,0,300,300);
        g2.setColor(this.formatter.formatBoardColor());//changes boardcolor

        g2.fill(boardSize);
            for (int i = 0; i < 3; i++) {
                  for (int j = 0; j < 3; j++) {
                        Rectangle2D.Double square = new
Rectangle2D.Double(i * SQUARE_SIDE, j * SQUARE_SIDE, SQUARE_SIDE,
                                    SQUARE_SIDE);
                        g2.setColor(this.formatter.formatLineColor());
                        g2.draw(square);
```

```
                }
            }


            //Draw x and o//
            Font biggerFont = new Font("Arial", Font.PLAIN, 90);
            g2.setFont(biggerFont);
            for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < 3; j++)
                {
                    System.out.println(i + " " +  j + " " +
moves[i][j]);
                    if(moves[i][j] == PLAYER_ONE)
                    {
                        g2.setColor(this.formatter.formatXColor());
                        g2.drawString("X", i*100+10, j*100+90);
                    }
                    else if (moves[i][j] == PLAYER_TWO)
                    {
                        g2.setColor(this.formatter.formatOColor());
                        g2.drawString("O", i*100+10, j*100+90);
                    }
                }
            }

      }
```