```java
public class Piece
{
     private Color color;
     private int row;
     private int col;

     public Piece(Color pColor, int pRow, int pCol)
     {
          color = pColor;
          row = pRow;
          col = pCol;
     }

     public void moveTo(int pRow, int pCol)
     {
          row = pRow;
          col = pCol;
     }

     public Color getColor()
     {
          return color;
     }

     public int getRow()
     {
          return row;
     }

     public int getCol()
     {
          return col;
     }

     public void draw(Graphics g, int squareSide)
     {
          g.setColor(color);
          g.fillOval(col * squareSide, row * squareSide, squareSide, squareSide);
     }
}
```

```java
public class MovePiece
{
     private Color color;
     private int x;
     private int y;

     public MovePiece(Color pColor, int pX, int pY)
     {
          color = pColor;
          x = pX;
          y = pY;
     }

     public void moveTo(int pX, int pY)
     {
          x = pX;
          y = pY;
     }

     public Color getColor()
     {
          return color;
     }

     public int getX()
     {
          return x;
     }

     public int getY()
     {
          return y;
     }

     public void draw(Graphics g, int squareSide)
     {
          g.setColor(color);
          g.fillOval(x - squareSide / 2, y - squareSide / 2, squareSide, squareSide);
     }
}
```

```java
public class Checkers
{
    private ArrayList<Piece> alPieces = new ArrayList<>();

    // Pièce en déplacement/mouvement (colorée en gris)
    private MovePiece inMove = null;

    public Checkers()
    {
        for (int r = 0; r < 8; r++)
        {
            for (int c = 0; c < 8; c++)
            {
                if ((r + c) % 2 == 1)
                    if (r < 3)
                        alPieces.add(new Piece(Color.BLUE, r, c));
                    else if (r >= 5)
                        alPieces.add(new Piece(Color.RED, r, c));
            }
        }
    }

    public Piece getPieceAt(int pRow, int pCol)
    {
        Piece p = null;
        boolean found = false;
        int i = 0;
        while (!found && i < alPieces.size())
        {
            p = alPieces.get(i);
            if ((pRow == p.getRow()) && (pCol == p.getCol()))
                found = true;
            else
                i++;
        }
        if (found)
            return alPieces.get(i);
        else
            return null;
    }

    public boolean move(int pStartRow, int pStartCol, int pDestRow, int pDestCol)
    {
        Piece p = getPieceAt(pStartRow, pStartCol);
        if (p == null)
            return false;        // Pas de pièce à cette position de départ

        // Enlever la pièce en mouvement
        inMove = null;

        // Pièce existe
        if ((pDestRow < 0) || (pDestCol < 0) || (pDestRow > 7) || (pDestCol > 7))
        {
            // Position en dehors du damier --> effacer la pièce
            alPieces.remove(p);
            return true;
        }

        if (getPieceAt(pDestRow, pDestCol) == null)
        {
            // Pas de pièce à la position d'arrivée --> on bouge la pièce
            p.moveTo(pDestRow, pDestCol);
            return true;
        }

        return false;
    }

    public void beginMove(int pX, int pY)
    {
        inMove = new MovePiece(Color.LIGHT_GRAY, pX, pY);
    }

    public void doMove(int pX, int pY)
    {
        // Bouge la pièce en déplacement vers la nouvelle position
        if (inMove != null)
            inMove.moveTo(pX, pY);
    }

    public void draw(Graphics g, int cellSize)
    {
        // Dessiner la grille
        int c, r;
        for (int i = 0; i < 8; i++)
        {
            r = (i * cellSize);
            for (int j = 0; j < 8; j++)
            {
                c = (j * cellSize);
                if ((i + j) % 2 == 1)
                    g.setColor(Color.GRAY);
                else
                    g.setColor(Color.WHITE);
                g.fillRect(c, r, cellSize, cellSize);
                g.setColor(Color.BLACK);
                g.drawRect(c, r, cellSize, cellSize);
            }
```

```
        }

        // Dessiner les pièces... chaque pièce se dessine elle-même!
        for (int i = 0; i < alPieces.size(); i++)
            alPieces.get(i).draw(g, cellSize);

        // Dessiner la pièce en déplacement
        if (inMove != null)
            inMove.draw(g, cellSize);
    }
}
```

```java
public class DrawPanel extends javax.swing.JPanel
{
     private Checkers checkers = null;
     private int squareSide = 0;

     public void setCheckers(Checkers checkers)
     {
          this.checkers = checkers;
     }

     public int getSquareSide()
     {
          return squareSide;
     }

     public DrawPanel()
     {
          initComponents();
     }

     public void paintComponent(Graphics g)
     {
          int w = getWidth();
          int h = getHeight();

          // Peindre l'arrière fond
          g.setColor(Color.WHITE);
          g.fillRect(0, 0, w, h);

          int l = Math.min(w, h) - 1;
          squareSide = l / 8;

          // Dessiner le damier et les pièces... s'il existe
          if (checkers != null)
               checkers.draw(g, squareSide);
     }
// Skipped: ... initComponents { ... }
    // Variables declaration - do not modify//GEN-BEGIN:variables
    // End of variables declaration//GEN-END:variables
}
```

```
public class MainFrame extends javax.swing.JFrame
{
     private Checkers checkers = null;
     private Piece piece = null;

     public MainFrame()
     {
          initComponents();
     }
// Skipped: ... initComponents { ... }
    private void newButtonActionPerformed(java.awt.event.ActionEvent evt)//GEN-FIRST:event_newButtonActionPerformed
    {//GEN-HEADEREND:event_newButtonActionPerformed
         checkers = new Checkers();
         drawPanel.setCheckers(checkers);

         repaint();
    }//GEN-LAST:event_newButtonActionPerformed

    private void drawPanelMousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_drawPanelMousePressed
         if (checkers != null)
         {
              int row = evt.getY() / drawPanel.getSquareSide();
              int col = evt.getX() / drawPanel.getSquareSide();

              piece = checkers.getPieceAt(row, col);
              if (piece != null)
              {
                   checkers.beginMove(evt.getX(), evt.getY());
                   msgLabel.setText("-");
              }
              else
              {
                   msgLabel.setText("No piece at that position!");
              }

              repaint();
         }
    }//GEN-LAST:event_drawPanelMousePressed

    private void drawPanelMouseReleased(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_drawPanelMouseReleased
         if ((checkers != null) && (piece != null))
         {
              int row = evt.getY() / drawPanel.getSquareSide();
              int col = evt.getX() / drawPanel.getSquareSide();

              if (checkers.move(piece.getRow(), piece.getCol(), row, col))
                   msgLabel.setText("-");
              else
                   msgLabel.setText("Invalid move!");
              piece = null;

              repaint();
         }
    }//GEN-LAST:event_drawPanelMouseReleased

    private void drawPanelMouseDragged(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_drawPanelMouseDragged
         if ((checkers != null) && (piece != null))
         {
              checkers.doMove(evt.getX(), evt.getY());
              repaint();
         }
    }//GEN-LAST:event_drawPanelMouseDragged
// Skipped: ... Look & Feel
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private DrawPanel drawPanel;
    private javax.swing.JComboBox jComboBox1;
    private javax.swing.JLabel msgLabel;
    private javax.swing.JButton newButton;
    // End of variables declaration//GEN-END:variables
}
```