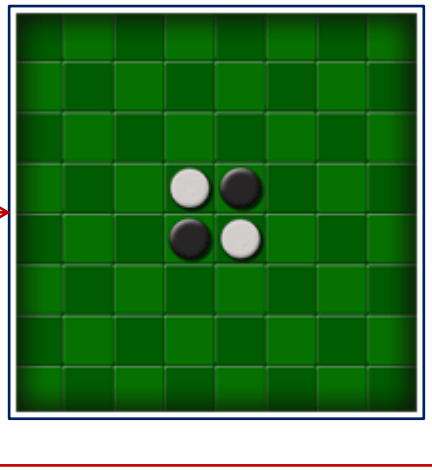
Teoria para Jogos.

Protótipo de tabuleiro para o jogo *Reversi*.

O jogo inicia na posição mostrada na figura, em um tabuleiro consistindo em 64 quadrados em uma disposição 8x8.



Cada peça do *reversi* tem um lado negro e um lado branco. Na sua vez, você coloca uma peça no tabuleiro com sua cor virada para cima. Você deve colocar a peça de forma que uma peça do oponente, ou uma sequência de peças do oponente, fique flanqueada por suas peças. Todas as peças adversárias que estiverem entre as suas peças mudam de cor, tornando-se suas.

Teoria para Jogos.

Classe peça, como herança a partir da classe JButton.

```
package grafica;
import java.awt.*;
import javax.swing.*;

public class Peca extends  JButton{
    private static int tamanho = 64;
    private Estado estado;
    public Peca(){
        super();
        estado = Estado.VAZIO;
    }
    @Override
    public Dimension getMaximumSize() { return getPreferredSize(); }
    @Override
    public Dimension getMinimumSize() { return getPreferredSize(); }
    @Override
    public Dimension getPreferredSize() { return new Dimension(tamanho,tamanho); }
```

```
package grafica;
public enum Estado { VAZIO, PRETO, BRANCO}
```

Alguns métodos na classe peça serão modificados para se adequar ao contexto da aplicação.

Teoria para Jogos.

As peças são brancas, pretas ou estão em vazio.

```
@Override
protected void paintComponent(Graphics g){
    super.paintComponent(g);
    Graphics2D g2d = (Graphics2D)g;
    // Não preenchemos botões vazios.
    if (estado != Estado.VAZIO){
        if (estado == Estado.BRANCO)
            g2d.setColor(Color.WHITE);
        else if (estado == Estado.PRETO)
            g2d.setColor(Color.BLACK);
        g2d.fillOval(6,6,getWidth()-12,getHeight()-12);
    }
    // Pintamos a borda da peça independente do estado.
    g2d.setColor(Color.GRAY);
    g2d.drawOval(6,6,getWidth()-12,getHeight()-12);
}
```

```
void setEstado(Estado estado) {
    this.estado = estado;
}
```

O método *paintComponente* também será modificado para desenhar a peça de forma adequada.

Teoria para Jogos.

Uma vez criada a classe que representa a peça no jogo resta agora criar o tabuleiro.

```
import javax.swing.*;
import java.awt.*;
public class Tabuleiro extends JPanel{
    private Peca[][] tabuleiro;
    public Tabuleiro(){
        setLayout(new GridLayout(8,8));
        tabuleiro = new Peca[8][8];
        for(int l=0;l<8;l++)
            for(int c=0;c<8;c++){
                tabuleiro[c][l] = new Peca();
                add(tabuleiro[c][l]);
            }
        tabuleiro[3][3].setEstado(Estado.BRANCO);
        tabuleiro[4][4].setEstado(Estado.BRANCO);
        tabuleiro[3][4].setEstado(Estado.PRETO);
        tabuleiro[4][3].setEstado(Estado.PRETO);
    }
}
```

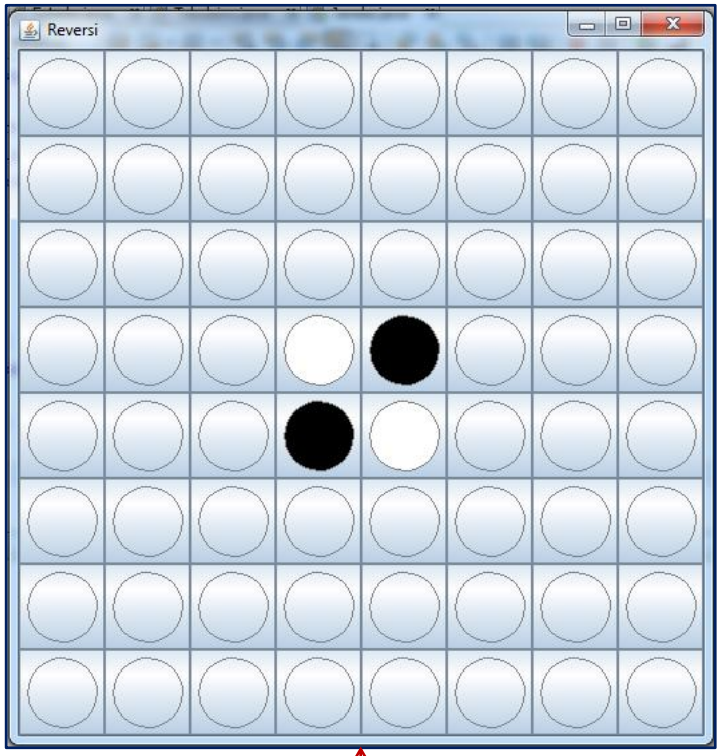
→ 64 casas.

As 4 casas no meio do tabuleiro começam cheias.

Teoria para Jogos.

Criando a janela gráfica para o tabuleiro do jogo.

```
import javax.swing.*;
public class Janela extends JFrame{
    public Janela(){
        super("Reversi");
        getContentPane().add(new Tabuleiro());
        pack();
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public static void main(String[] args){
        new Janela();
    }
}
```



É possível melhorar a aparência da janela gráfica.