

# Aufgabe 5: Bauernopfer

Team: „Was ist JAVA?“

Team-ID: 00107

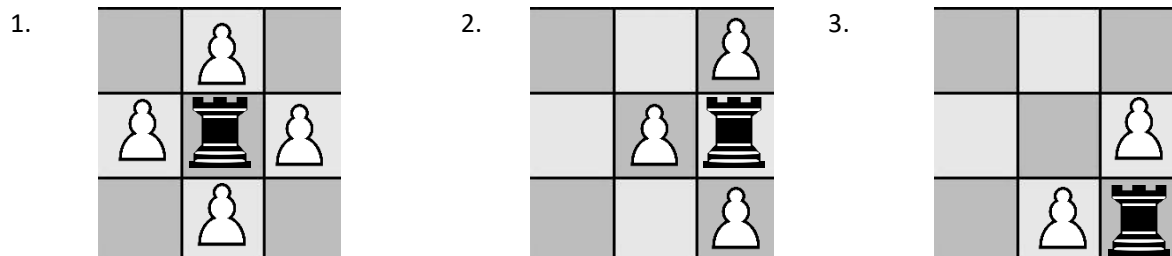
3. November 2017

## Inhaltsverzeichnis

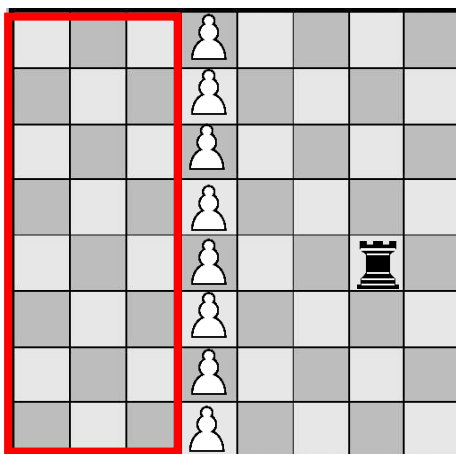
1.	Lösungsidee	S.1
2.	Umsetzung	S.2
3.	Beispiele	S.2
4.	Quellcode	S.3
5.	Lösungen der Aufgaben	S.8

## Lösungsidee

Unsere Idee zur Lösung der Aufgabe „Bauernopfer“ war es die Bewegung des Turms einzuschränken. Zuerst sollte man betrachten, wann der Turm von den Bauern gefangen wird. Dies passiert in drei Konstellationen:



Man erkennt das in allen Fällen, dass alle Bewegungsrichtungen des Turms abgedeckt werden und somit keine Fluchtmöglichkeiten offen sind.

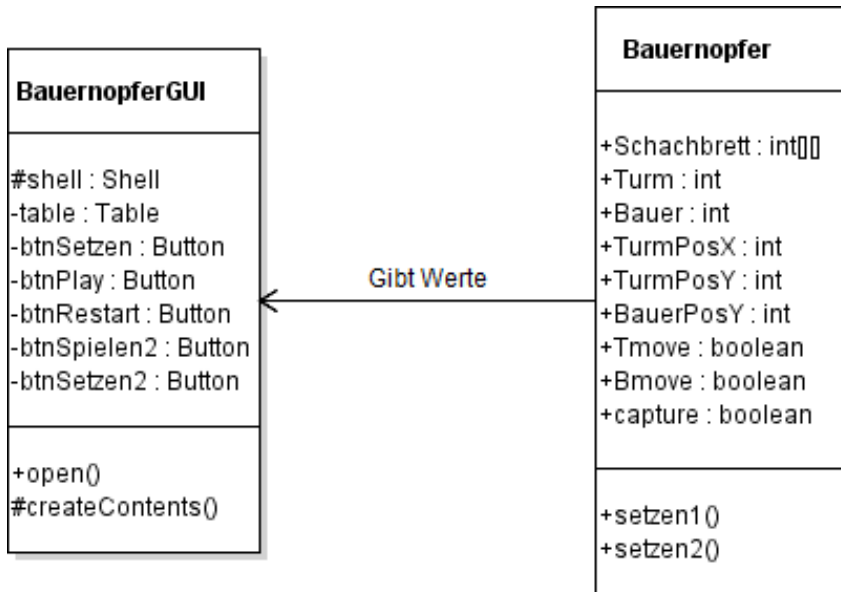


Dies brachte uns auf die Idee das Spielfeld immer kleiner zu machen und somit den Turm einzugrenzen. Die Bauern müssen hierfür das Spielfeld halbieren, damit der Turm nur noch die Hälfte des Schachbretts zu Verfügung hat.

Der Ansatz würde dann wie in der Abbildung links aussehen. Dementsprechend würde der rot markierte Bereich nicht mehr relevant für den Spielverlauf werden, da der Turm keine Möglichkeit hat diesen zu betreten

## Aufgabe 5: Bauernopfer

### Umsetzung

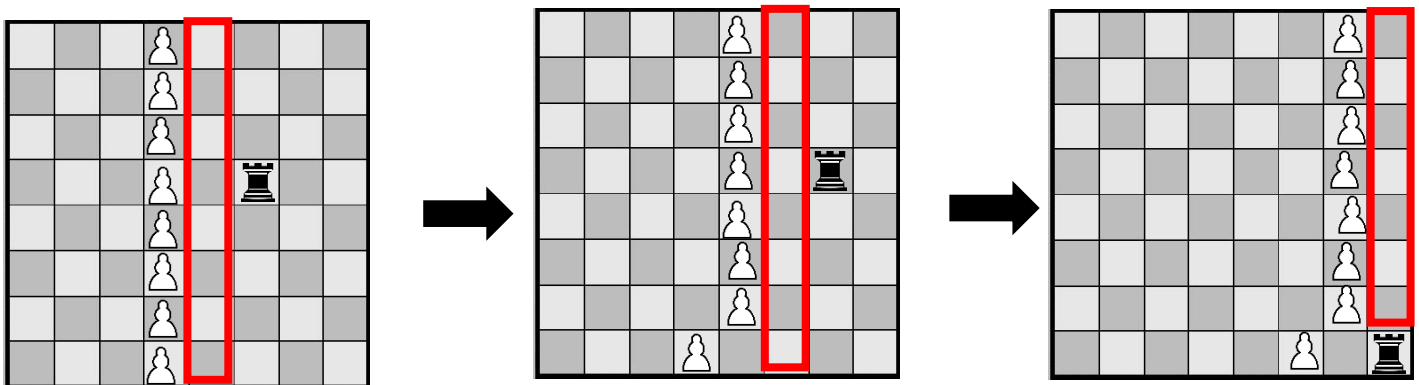


Unser Programm besteht aus zwei Klassen, der Graphischen Oberfläche **BauernopferGUI** und der Klasse **Bauernopfer**. Zuerst wird in der Klasse **Bauernopfer** das ganze Spiel virtuell „gespielt“. Dies passiert auf dem zweidimensionalen Array `Schachbrett[][]`. Für jeden Zugwechsel wird eine neue Konstellation des Schachbretts auf dem Array abgebildet und wird an die GUI weitergeleitet, die dann diese Werte auf einer Tabelle in Formen von „B“ für Bauer und „T“ für Turm dargestellt. Die GUI haben wir mittels SWT visualisiert.

### Beispiele

Man sieht hier, dass der Turm keine Fluchtmöglichkeit besitzt um von den Bauern zu fliehen. Der Turm darf sich auch nicht in den rot markierten Bereich begeben, sonst würde ihn ein Bauer danach fangen.

Die Bauern müssen nur sich einzeln nacheinander in Richtung des Turms bewegen und würde dann früher oder später gefangen werden. Somit würde das Spiel auf den Fall 2. oder 3. aus der Lösungsidee zulaufen



### Quellcode

```
public class Bauernopfer {
    public int Schachbrett[][] = new int[8][8];
    public int Turm = 2 ;
    public int Bauer = 1;
```

## Aufgabe 5: Bauernopfer

```

public int TurmPosX;
public int TurmPosY;
public int BauerPosY;
public int Züge = 0;
public boolean Tmove = false;
public boolean Bmove = true;
public boolean capture = false;

// Aufgabe 1
public void setzen1() {
//setzen
    TurmPosX = 0;
    TurmPosY = 7;
    BauerPosY = 3;
    for (int i = 0; i < 8 ; i++) {
        Schachbrett [i][BauerPosY] = Bauer;
    } // end of for
    Schachbrett [TurmPosX][TurmPosY] = Turm;
}

// Aufgabe 2
public void setzen2() {
    TurmPosX = 7;
    TurmPosY = 0;
    // setzen
    for (int i = 0; i < 7 ; i++ ) {
        Schachbrett[i][i] = Bauer;
    } // end of for
    Schachbrett [TurmPosX][TurmPosY] = Turm;
}
}

```

// setzt die Fi-  
guren für Auf-  
gabe 1

// setzt die Fi-  
guren für Auf-  
gabe 2

```

import org.eclipse.swt.widgets.Display;
public class BauernopferGUI {
    protected Shell shell;
    private Table table;
    private Button btnSetzen;
    private Button btnPlay;
    private Button btnRestart;
    private Button btnSpielen2;
    private Button btnSetzen2;
    Bauernopfer bf = new Bauernopfer();

    protected void createContents() {
        shell = new Shell();
        shell.setSize(292, 220);
        shell.setText("SWT Application");

        table = new Table(shell, SWT.BORDER | SWT.FULL_SELECTION);
        table.setBounds(10, -13, 183, 180);
        table.setHeaderVisible(true);
        table.setLinesVisible(true);

        for (int i = 0; i < 8; i++) {
            TableColumn tblclmnNewColumn = new TableColumn(table, SWT.NONE);
            tblclmnNewColumn.setWidth(22);
            tblclmnNewColumn.setText("");
        }
    }
}

```

## Aufgabe 5: Bauernopfer

```

    }
    int temp = 0;
    int zaehler = 1;
    for (int i = 0; i < 8; i++) {
        TableItem tableItem = new TableItem(table, SWT.NONE);
        for (int i1 = 0; i1 < 4; i1++) {
            tableItem.setBackground(i1+temp, SWTRe-
sourceManager.getColor(SWT.COLOR_WIDGET_NORMAL_SHADOW));
            tableItem.setText("");
            temp++;
        }
        if(zaehler == 2) {
            temp = 0;
            zaehler = 1;
        }
        else{
            temp = 1;
            zaehler = 2;
        }
    }
    btnPlay = new Button(shell, SWT.NONE);
    btnPlay.setBounds(199, 41, 75, 25);
    btnPlay.setText("Spielen!");
    btnPlay.addListener(SWT.Selection, new Listener(){
        @Override
        public void handleEvent(Event arg0) {
            // TODO Auto-generated method stub
            bf.TurmPosX = 0;
            bf.TurmPosY = 7;
            bf.BauerPosY = 3;
            int temp = 0;

            //Bewegen
            while (bf.capture != true) {
                if (bf.Schachbrett[bf.TurmPosX][bf.TurmPosY-1] == bf.Bauer &&
                    [bf.Schachbrett[bf.TurmPosX-1][bf.TurmPosY] == bf.Bauer)
                {
                    bf.capture = true;
                } // end of if

                else {

                    //Bauer Zug
                    if (bf.Bmove == true) {
                        if (bf.Schachbrett[bf.TurmPosX][bf.TurmPosY-1] == bf.Bauer) {
                            while (temp < 7) {
                                bf.BauerPosY = 0;
                                bf.Schachbrett [bf.BauerPosY+ temp][bf.TurmPosY-1] = 0;
                                bf.Schachbrett [bf.BauerPosY+ temp][bf.TurmPosY] = bf.Bauer;
                                temp++;
                            } // end of while
                        } // end of if
                        else {
                            bf.Schachbrett [bf.TurmPosX+temp][bf.BauerPosY-1] = 0;
                            bf.Schachbrett [bf.TurmPosX+temp][bf.BauerPosY] = bf.Bauer;
                        } // end of if-else

                        bf.Bmove = false;
                        bf.Tmove = true;
                        bf.Züge++;
                    }
                }
            }
        }
    });

```

// erstellt Ta-  
belle und färbt  
sie schwarz und  
weiß

//Button wird  
erstellt um Auf-  
gabe 1 zu spie-  
len. Dieser ent-  
hält die Aktua-  
lisierung der  
Tabelle der  
Werte, sowie die  
besten Zugmög-  
lichkeiten der  
Kontrahenten

## Aufgabe 5: Bauernopfer

```

        for (int i = 0; i < 8; i++) {

            for (int i3 = 0; i3 < 8; i3++) {
                if (bf.Schachbrett[i][i3] == 1) {
                    TableItem row = table.getItem(i);
                    row.setText(i3, "B");

                }
            }
            else {
                TableItem row = table.getItem(i);
                row.setText(i3, " ");

            }

            //Turm Zug
            if (bf.Tmove == true) {
                if (bf.Schachbrett[bf.TurmPosX][bf.TurmPosY-1]
== bf.Bauer && bf.Schachbrett[7][7] != bf.Turm) {
                    bf.Schachbrett [bf.TurmPosX][bf.TurmPosY] = 0
                    bf.TurmPosX = bf.TurmPosX+1;
                    bf.Schachbrett [bf.TurmPosX][bf.TurmPosY] = bf.Turm;
                } // end of if
            }
            else {
                if (temp+bf.TurmPosX >= 7) {
                    temp = 0;
                    bf.BauerPosY++;
                } // end of if
            }
            else {
                temp++;
            } // end of if-else

        } // end of if-else
        bf.Bmove = true;
        bf.Tmove = false;
        bf.Züge++;
        for (int i = 0; i < 8; i++) {

            for (int i3 = 0; i3 < 8; i3++) {
                if (i == bf.TurmPosX && i3 == bf.TurmPosY) {
                    TableItem row = table.getItem(i);
                    row.setText(i3, "T");

                }
            }

        }

        btnRestart = new Button(shell, SWT.NONE);
        btnRestart.setBounds(199, 72, 75, 25);
        btnRestart.setText("Restart");
        btnRestart.addSelectionListener(new SelectionAdapter() {
            @Override
            public void widgetSelected(SelectionEvent e) {
                bf.createSchachbrett();
                table.dispose();
                table = new Table(shell, SWT.BORDER | SWT.FULL_SELECTION);
                table.setBounds(10, -13, 183, 180);
                table.setHeaderVisible(true);
                table.setLinesVisible(true);

            }

        });

        btnSpielen2 = new Button(shell, SWT.NONE);
        btnSpielen2.setBounds(199, 134, 75, 25);
        btnSpielen2.setText("Spielen2");
        btnSpielen2.addSelectionListener(new SelectionAdapter() {
            @Override
            public void widgetSelected(SelectionEvent e) {

```

//Button der das  
Brett neu gene-  
riert

//Button wird  
erstellt um Auf-  
gabe 2 zu spie-  
len. Dieser ent-

## Aufgabe 5: Bauernopfer

```

        bf.TurmPosX = 7;
        bf.TurmPosY = 0;
        bf.BauerPosY = 6;
        int temp = 0;
        int temp2 = 0;
        boolean check = false;

        while (bf.capture != true) {
            if (bf.Schachbrett[bf.TurmPosX][bf.TurmPosY+1] == bf.Bauer &&
bf.Schachbrett[bf.TurmPosX-1][bf.TurmPosY] == bf.Bauer) {
                bf.capture = true;
            } // end of if

            else {

                for (int i = 0; i < 8 ; i++ ) {
                    if (bf.Schachbrett[bf.TurmPosX][i] == bf.Bauer) {
                        check = true;
                        i = 8;
                    } // end of if
                    else {
                        check = false;
                    } // end of if-else
                } // end of for

                // Zug Bauer
                if (bf.Bmove == true) {
                    if (check != true) {
                        bf.Schachbrett[bf.TurmPosX-1][bf.BauerPosY-temp] = 0;
                        bf.Schachbrett[bf.TurmPosX][bf.BauerPosY-temp] = bf.Bauer;
                        temp++;
                    } // end of if
                    else {
                        if (bf.Schachbrett[temp2][bf.BauerPosY] == bf.Bauer &&
bf.Schachbrett[temp2+2][bf.BauerPosY+1] == bf.Bauer ) {
                            bf.Schachbrett[temp2][bf.BauerPosY] = 0;
                            temp2 += 1;
                            bf.Schachbrett[temp2][bf.BauerPosY] = bf.Bauer;
                            bf.BauerPosY++;
                            temp += 2;
                        } // end of if
                        else {
                            if (temp + 1 <= 7) {
                                bf.Schachbrett[temp][bf.BauerPosY] = 0;
                                temp++;
                                bf.Schachbrett[temp][bf.BauerPosY] = bf.Bauer;
                                bf.BauerPosY++;
                            } // end of if
                            else {
                                bf.BauerPosY = 0;
                                bf.Schachbrett[temp2][bf.BauerPosY] = 0;
                                temp2++;
                                bf.Schachbrett[temp2][bf.BauerPosY] = bf.Bauer;
                                bf.BauerPosY++;
                                temp = temp2+1;
                            } // end of if-else
                        } // end of if-else
                    }
                }
            }
        }
    
```

hält die Aktualisierung der Tabelle der Werte, sowie die besten Zugmöglichkeiten der Kontrahenten

## Aufgabe 5: Bauernopfer

```

        } // end of if-else
        bf.Bmove = false;
        bf.Tmove = true;
        for (int i = 0; i < 8; i++) {

            for (int i3 = 0; i3 < 8; i3++) {
                if (bf.Schachbrett[i][i3] == 1) {
                    TableItem row = table.getItem(i)
                    row.setText(i3, "B");

                }
                else {
                    TableItem row = table.getItem(i);
                    row.setText(i3, " ");
                }
            }
        }

    } // end of if

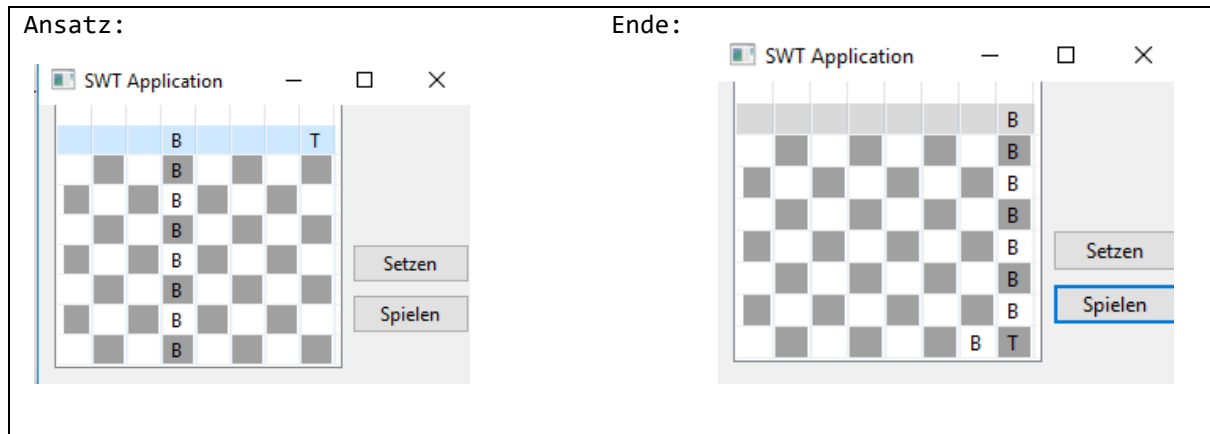
    // Zug Turm
    if (bf.Tmove == true) {
        if (check != true) {
            if (bf.Schachbrett[bf.TurmPosX-2][bf.TurmPosY] == bf.Bauer) {
                bf.Schachbrett[bf.TurmPosX][bf.TurmPosY] = 0;
                bf.TurmPosX = 7;
                bf.Schachbrett[bf.TurmPosX][bf.TurmPosY] = bf.Turm;
                bf.BauerPosY = 0;
                temp = 0;
                temp2 = temp;

            } // end of if
            else {
                bf.Schachbrett[bf.TurmPosX][bf.TurmPosY] = 0;
                bf.TurmPosX -= 1;
                bf.Schachbrett[bf.TurmPosX][bf.TurmPosY] = bf.Turm;
            } // end of if-else
        } // end of if
        bf.Bmove = true;
        bf.Tmove = false;
    }

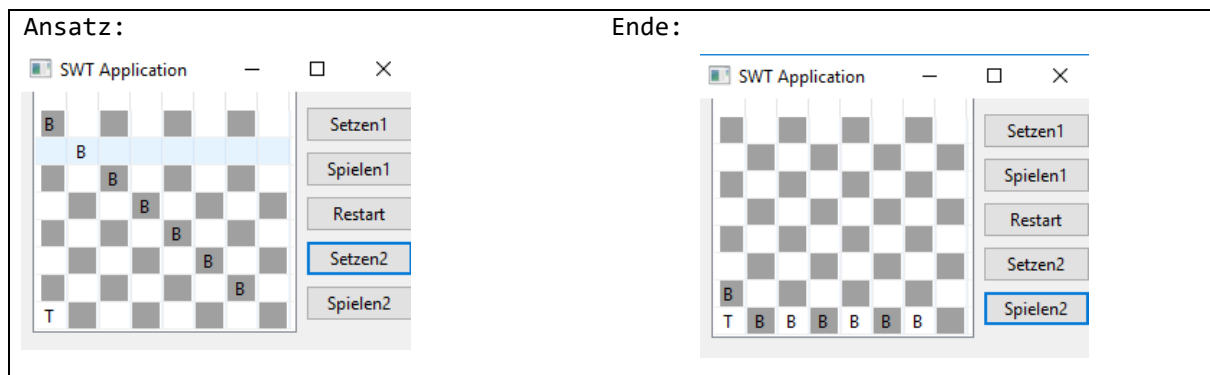
```

## Lösungen der Aufgaben

### AUFGABE 1



### AUFGABE 2



### AUFGABE 3

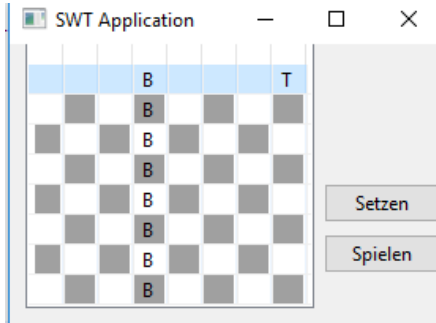
Für  $k = 6$  und  $l = 76$ .

### AUFGABE 4

Man nehme den Ansatz von Aufgabe 1 und somit wäre auch die Bewegung auf der Diagonalen geblockt.

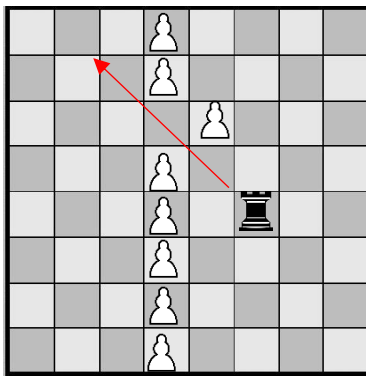
## Aufgabe 5: Bauernopfer

Ansatz:

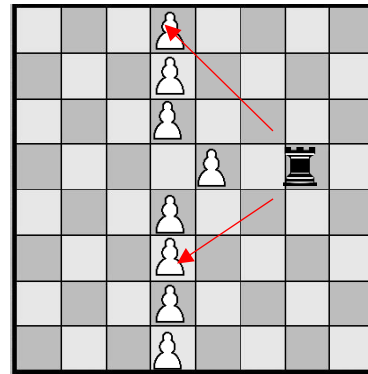


Die Bauern aber müssen ihre Bewegung sich der Dame anpassen, sonst hätte sie doch eine Fluchtmöglichkeit.

FALSCH



RICHTIG



Der Bauer der vor der Dame steht müsste somit sich immer zuerst sich bewegen um dir Dame zu fangen.