5. Source-Code

5.1. BattleShipGUI.java

```
1.  /* Applet "Schiffe versenken" - BattleShipGUI.java
2.   * Fachakademie für Angewandte Informatik
3.   * Softwaredeveloper - Java
4.   * (C) Ing. Hannes Gastl, 2012
5.   */
6.
7.  package SchiffeVersenken;
8.
9.  import java.applet.*;
10. import java.awt.*;
11. import java.awt.event.*;
12. import java.net.*;
13.
14. public class BattleShipGUI extends Applet
15. {
16.     public BattleShipGUI() {
17.     }
18.     //Variablen
19.     public static BattleShipField field_Player;              //Objekt von BattleShipField
20.     public static BattleShipField field;                     //Objekt von BattleShipField
21.     public static BattleShipField field_PC;                  //Objekt von BattleShipField
22.     public static ShipShot field_Shot;                       //Objekt von ShipShot
23.     static AudioClip soundHitShip;                           //Audio Datei für Treffer Schiff
24.     static AudioClip soundHitWater;                          //Audio Datei für Treffer Wasser
25.     static AudioClip soundWin;                               //Audio Datei für Gewonnen
26.     boolean beginShots = false;                              //wird auf True gesetzt, sobald der Button zum Schiessen gedruckt wird
27.
28.     //Applet wird initialisiert
29.     public void init()
30.     {
31.             //Appletgroesse
32.             setLayout(null);
33.         setSize(750,500);
34.         setBackground(new Color(188,207,230));
35.
36.         soundHitShip = getAudioClip(getDocumentBase(),"ship.au");
37.         soundHitWater = getAudioClip(getDocumentBase(),"water.au");
38.         soundWin = getAudioClip(getDocumentBase(),"win.au");
```



```
39.
40.         field = new BattleShipField(getGraphics());
41.         field_PC = new BattleShipField(getGraphics());
42.         field_Player = new BattleShipField(getGraphics());
43.         field_Shot = new ShipShot(getGraphics());
44.
45.         //Benötigte Button
46.         Button drawShip_PC = new Button("PC Schiffe platzieren");          //Button zum Setzen der Schiffe des PCs
47.         Button drawShip_Player = new Button("eigene Schiffe platzieren");  //Button zum Setzen der Schiffe des Players
48.         Button setShots_Player = new Button("Schüsse abfeuern - Stufe 1"); //Button zum Starten der Schüsse des Players - LEVEL 1
49.         Button setShots2_Player = new Button("Schüsse abfeuern - Stufe 2");//Button zum Starten der Schüsse des Players - LEVEL 2
50.         Button restart = new Button("Neustart");                           //Button zum Neustarten der Applikation
51.         Button help = new Button("Hilfe und Anleitung");                   //Button zum Aufruf der Hilfe
52.
53.         //Position der Button
54.         drawShip_PC.setBounds(580, 25, 160, 40);
55.         drawShip_Player.setBounds(580, 75, 160, 40);
56.         setShots_Player.setBounds(580, 125, 160, 40);
57.         setShots2_Player.setBounds(580, 175, 160, 40);
58.         restart.setBounds(580, 225, 160, 20);
59.         help.setBounds(580, 255, 160, 20);
60.
61.         //Button hinzufügen
62.         add(drawShip_PC);
63.         add(drawShip_Player);
64.         add(setShots_Player);
65.         add(setShots2_Player);
66.         add(restart);
67.         add(help);
68.
69.         //Erstellt die Schiffe des PCs
70.         drawShip_PC.addActionListener(new ActionListener() {
71.            public void actionPerformed(ActionEvent arg0) {
72.                    if (beginShots == false)
73.                            {
74.                            field_PC.drawShip_PC();
75.                            }
76.                    else field_Shot.drawInstruction1("Spiel wurde bereits begonnen!", ShipShot.insAlert);
77.            }
78.         });
79.
80.         //Erstellt die Schiffe des Players
81.         drawShip_Player.addActionListener(new ActionListener() {
```

```
82.           public void actionPerformed(ActionEvent arg0) {
83.                  if (beginShots == false)
84.                         {
85.                         addMouseListener(field_Player);      //Listener für Applet Mausklick -> Objekt BattleShipField
86.                         field_Shot.drawInstruction1("Zum Platzieren der Schiffe entsprechendes Feld anklicken!", ShipShot.insAlert);
87.                         field_Shot.drawInstruction2("Linke Maustaste: Horizontal ::::::: Rechte Maustaste: Vertikal",
                                ShipShot.insStd);
88.                         }
89.                  else field_Shot.drawInstruction1("Spiel wurde bereits begonnen!", ShipShot.insAlert);
90.           }
91.
92.        });
93.
94.        //Startet das Spiel zum Abschiessen der Schiffe --> Level 1
95.        setShots_Player.addActionListener(new ActionListener() {
96.           public void actionPerformed(ActionEvent arg0) {
97.                  if (BattleShipField.drawShipOK && BattleShipField.drawShipOK_PC) //Abfrage ob bereits alle Schiffe gesetzt wurden
98.                  {
99.                         if (beginShots == false)
100.                               {
101.                                      removeMouseListener(field_Player);  //MouseListener für den field_Player deaktivieren
102.                                      field_Shot.drawInstruction1("PC und Player haben alle Schiffe gesetzt!", ShipShot.insStd);
103.                                      field_Shot.drawInstruction2("Mit einem Klick auf ein Feld können die Schüsse ausgelöst
                                              werden!", ShipShot.insStd);
104.                                      addMouseListener(field_Shot);       //Listener für Applet Mausklick -> Objekt ShipShot
105.                                      ShipShot.playLevel2 = false;        //auf Spiellevel 1 setzen
106.                                      beginShots = true;
107.                              }
108.                              else field_Shot.drawInstruction1("Spielstufe wurde bereits gesetzt und das Spiel begonnen!",
                                      ShipShot.insAlert);
109.                      }
110.                      else field_Shot.drawInstruction1("Zuerst müssen alle Schiffe des PCs und des Spielers gesetzt sein!",
                              ShipShot.insAlert);
111.                 }
112.          });
113.
114.          //Startet das Spiel zum Abschiessen der Schiffe --> Level 2
115.          setShots2_Player.addActionListener(new ActionListener() {
116.                 public void actionPerformed(ActionEvent arg0) {
117.                        if (BattleShipField.drawShipOK && BattleShipField.drawShipOK_PC) //Abfrage ob bereits alle Schiffe gesetzt
                               wurden
118.                        {
119.                               if (beginShots == false)
```



```
120.                              {
121.                                     removeMouseListener(field_Player);  //MouseListener für den field_Player deaktivieren
122.                                     field_Shot.drawInstruction1("PC und Player haben alle Schiffe gesetzt!", ShipShot.insStd);
123.                                     field_Shot.drawInstruction2("Mit einem Klick auf ein Feld können die Schüsse ausgelöst
                                             werden!", ShipShot.insStd);
124.                                     addMouseListener(field_Shot);       //Listener für Applet Mausklick -> Objekt ShipShot
125.                                     ShipShot.playLevel2 = true;  //auf Spiellevel 1 setzen
126.                                     beginShots = true;
127.                             }
128.                             else field_Shot.drawInstruction1("Spielstufe wurde bereits gesetzt und das Spiel begonnen!",
                                     ShipShot.insAlert);
129.                     }
130.                     else field_Shot.drawInstruction1("Zuerst müssen alle Schiffe des PCs und des Spielers gesetzt sein!",
                             ShipShot.insAlert);
131.                }
132.         });
133.
134.         //Beenden des Applets und Zerstörung
135.         restart.addActionListener(new ActionListener() {
136.                public void actionPerformed(ActionEvent arg0) {
137.                       restart();
138.                }
139.         });
140.
141.         //Öffnet ein HTML-Fenster
142.         help.addActionListener(new ActionListener() {
143.                public void actionPerformed(ActionEvent arg0) {
144.                     try {
145.                                    getAppletContext().showDocument(
146.                                           new URL("http://www.gastl.cc/downloads/files/BattleShip/BattleShipHelp.html"),
                                              "_blank");
147.                                    } catch (MalformedURLException e) {
148.                                           e.printStackTrace();
149.                                    }
150.                }
151.         });
152.        } // Ende init()
153.
154.        //Applet-Information
155.        public String getAppletInfo()
156.        {
157.               return "Information: (C) Ing. Hannes Gastl, 2012 ";
158.        }
```

```
159.
160.          //start() Methode
161.          public void start()
162.          {
163.          }
164.
165.          //stop() Methode
166.          public void stop()
167.          {
168.          }
169.
170.          //destroy() Methode
171.          public void destroy()
172.          {
173.          }
174.
175.          //Auf Ausgangswerte zurücksetzen
176.          public void restart()
177.          {
178.                 repaint();
179.                 removeListener();
180.
181.                 BattleShipField.drawShipOK = false;
182.                 BattleShipField.drawShipOK_PC = false;
183.                 beginShots = false;
184.
185.                 field_PC.shipCount = 0;
186.                 field_Player.shipCount = 0;
187.
188.                 for (int i=0; i<BattleShipField.ShipLength.length; i++)
189.                 {
190.                        field_Shot.hitShipCounter[i] = 0;
191.                        field_Shot.hitShipCounter_PC[i] = 0;
192.                 }
193.
194.                 field_Shot.win_Player = false;
195.                 field_Shot.end = false;
196.                 field_Shot.hits_Player=0;
197.                 field_Shot.hits_PC=0;
198.                 field_Shot.shots_Player=0;
199.                 field_Shot.shots_PC=0;
200.                 field_Shot.shipSunk_Player=0;
201.                 field_Shot.shipSunk_PC=0;
```



```
202.          }
203.
204.          public void removeListener()
205.          {
206.                 removeMouseListener(field_Player);
207.                 removeMouseListener(field_Shot);
208.          }
209.
210.          public void paint(Graphics g)
211.          {
212.                 showStatus("Das Applet wurde gestartet");
213.                 g.setFont(new Font("SansSerif", Font.BOLD, 12));
214.                 g.setColor(Color.black);
215.                 g.drawString("************** eigenes Spielfeld **************", 25, 20);
216.                 g.drawString("************** Spielfeld des PCs **************", 300, 20);
217.              g.setColor(new Color(154,209,230));
218.              g.fillRect(25,25,250,250);             //Meer des ersten Spielfeldes
219.              g.fillRect(300,25,250,250);            //Meer des zweiten Spielfeldes
220.
221.              g.setColor(Color.black);
222.              field.drawBattleGui(25,25,g);
223.              field.drawBattleGui(300,25,g);
224.              field_Player.setSee();   //Status der Felder auf 0/Meer setzen
225.              field_PC.setSee();             //Status der Felder auf 0/Meer setzen
226.              field_Shot.drawInstruction1("Zum Beginnen Schiffe platzieren. Bitte entsprechende Schaltfläche auswählen!",
                                              ShipShot.insAlert);
227.          }
228. }
```

5.2. BattleShipField.java

```
1.  /* Applet "Schiffe versenken" - BattleShipField.java
2.   * Fachakademie für Angewandte Informatik
3.   * Softwaredeveloper - Java
4.   * (C) Ing. Hannes Gastl, 2012
5.   */
6.  package SchiffeVersenken;
7.
8.  import java.awt.*;
9.  import java.awt.event.*;
10.
11. class BattleShipField extends MouseAdapter
12. {
13.     private Graphics g;
14.     public static ShipShot ss; //Objekt von ShipShot zur Ausgabe der Instruktionen
15.     //Komponenten des Spielfeldes
16.     static int anz = 10;
17.     int abstand = 25;
18.     int startX_PC = 300;
19.     int startX=25;
20.     int startY=25;
21.     static int field[][] = new int[anz][anz];
22.     static int field_PC[][] = new int[anz][anz];
23.     int field_status;
24.     int field_status_PC;
25.             /*
26.              * 0 = Schiff 1 (Länge 2),
27.              * 1 = Schiff 2 (Länge 2),
28.              * 2 = Schiff 3 (Länge 3),
29.              * 3 = Schiff 4 (Länge 4),
30.              * 4 = Schiff 5 (Länge 5),
31.              * 7 = Schiffrand
32.              * 8 = Schuss gesetzt
33.              * 9 = Meer
34.              */
35.     public static int[] ShipLength = {2,2,3,4,5}; // 5 verschiede Schiffe mit Länge 2, 3, 4, und 5
36.     int shipCount = 0;
37.     int shipAnz =  ShipLength.length;   //=5 Gesamt-Anzahl der Schiffe
38.     static boolean drawShipOK = false, drawShipOK_PC = false;//bekommen den Status true sobald alle Schiffe gesetzt wurden
39.
40.     //Konstruktor
41.     public BattleShipField(Graphics g)
```



```
42.     {
43.             this.g=g;
44.             ss = new ShipShot(g); //ShipShot zur Ausgabe der Instruktionen
45.     }
46.
47.     //Set-Methode für den Status des Feldes
48.     public static void setField_PC_Element(int x, int y, int value)
49.     {
50.             field_PC[x][y]=value;
51.     }
52.     public static void setField_Element(int x, int y, int value)
53.     {
54.             field[x][y]=value;
55.     }
56.
57.     //Get-Methode für den Status des Feldes
58.     public static int getField_PC_Element(int x, int y)
59.     {
60.             return field_PC[x][y];
61.     }
62.     public static int getField_Element(int x, int y)
63.     {
64.             return field[x][y];
65.     }
66.
67.     //Get-Methode um die Länge des Schiffes abhängig vom Typ zu bekommen
68.     public static int getShipLength(int typ)
69.     {
70.             return ShipLength[typ];
71.     }
72.
73.     //Liefert X-Koordinate des Mausklickes
74.     public int getX(int posX)
75.     {
76.         if ((posX >= startX) && (posX <= startX+anz*abstand) ) return posX;
77.         else return 0;     //Rückgabe 0 wenn ausserhalb des Feldes
78.      }
79.
80.     //Liefert Y-Koordinate des Mausklickes
81.     public int getY(int posY)
82.     {
83.             if ((posY >= startY) && (posY <= startY+anz*abstand) ) return posY;
84.             else return 0;//Rückgabe 0 wenn ausserhalb des Feldes
```

```
85.     }
86.
87.     //Status des Spielfeldes auf 9/Meer setzen
88.     public void setSee()
89.     {
90.             field_status=9;
91.             field_status_PC=9;
92.             for(int i=0; i<anz; i++)
93.             {
94.                     for(int j=0; j<anz; j++)
95.                     {
96.                             setField_Element(i, j, field_status);
97.                             setField_PC_Element(i, j, field_status_PC);
98.                     }
99.             }
100.                    ss.drawInstruction2("Alle Felder auf Status Meer gesetzt!", ShipShot.insStd);
101.            }
102.
103.            //Zeichnet das Spielfeld
104.            public void drawBattleGui(int startX, int startY, Graphics g)
105.        {
106.                    this.startX=startX;
107.                    this.startY=startY;
108.                    this.g = g;
109.                    for (int i=0; i<=anz; i++)
110.            {
111.                    g.drawLine(startX, startY+abstand*i, startX+anz*abstand,startY+abstand*i);
112.                    g.drawLine(startX+abstand*i ,startY, startX+abstand*i, startY+abstand*anz);
113.            }
114.        }
115.
116.            public void mouseClicked(MouseEvent event)
117.            {
118.                    int fieldX=(getX(event.getX())/abstand)-1; // Div. durch Abstand damit man auf die Kästchen-Nummerierung kommt / -1
                                                                      damit das Feld bei 0/0 anfängt
119.                    int fieldY=(getY(event.getY())/abstand)-1; // Div. durch Abstand damit man auf die Kästchen-Nummerierung kommt    /
                                                                      -1 damit das Feld bei 0/0 anfangt
120.
121.                if ((fieldX >= 0) && (fieldY >= 0))
122.                {
123.                    //System.out.println("\nMausklick (x/y): " +fieldX +" / " +fieldY);
124.                    //System.out.println("Aufruf DrawShip!");
125.                    switch (event.getModifiers())       //16 für Linke Maustaste / 4 für Rechte Maustaste
```



```
126.                            {
127.                            case 16:
128.                                    drawShip(fieldX, fieldY, shipTyp(), true, g);
129.                                    break;
130.                            case 4:
131.                                    drawShip(fieldX, fieldY, shipTyp(), false, g);
132.                                    break;
133.                            default: ss.drawInstruction2("Falsche Maustaste!", ShipShot.insAlert);
134.                            }
135.                }
136.                else ss.drawInstruction1("Platzierung ausserhalb des Spielfeldes nicht möglich!", ShipShot.insAlert);
137.            }
138.
139.            //Schiff Player werden gezeichnet
140.            public void drawShip(int x, int y, int typ, boolean DirectionHorizontal, Graphics g)
141.            {
142.                    g.setColor(Color.orange);
143.                //System.out.println("ShipLength: " +ShipLength[typ]);
144.                if (shipCount < shipAnz)    //Prüft ob die Anzahl der Schiffe noch nicht erreicht wurde
145.                {
146.                    //System.out.println("Feldstatus: " +field[x][y]);
147.                    if (DirectionHorizontal)
148.                    {
149.                            if(enoughSpace(x, y, typ, DirectionHorizontal, field))    //wenn alle Feld-Stati 9/Meer dann kann Schiff
                                                                                          platziert werden
150.                            {
151.                                    g.fill3DRect(abstand+x*abstand+15/2, abstand+y*abstand+15/2, abstand*ShipLength[typ]-15, abstand-15,
                                                    true); //+ und - Beträge für dünneren Balken
152.                                    setShipFields(x, y, typ, DirectionHorizontal, field);    //benötigte Felder auf Status Schiff setzen
153.                                    shipCount++;
154.                                    setShipBorder(x, y, typ, DirectionHorizontal, field);    //umliegende Felder auf Status Schiffrand
                                                                                                 setzen
155.                            }
156.                            else ss.drawInstruction2("Schiff kann nicht platziert werden, da es ein anderes Schiff kreuzt oder berührt!",
                                                            ShipShot.insAlert);
157.                    }
158.                    else          //Vertikal
159.                    {
160.                            if(enoughSpace(x, y, typ, DirectionHorizontal, field))   //wenn alle Feld-Stati 9/Meer dann kann
                                                                                         Schiff platziert werden
161.                            {
162.                                    g.fill3DRect(abstand+x*abstand+15/2, abstand+y*abstand+15/2, abstand-15,
                                                    abstand*ShipLength[typ]-15, true);  //+ und - Beträge für dünneren Balken
```

```
163.                                    setShipFields(x, y, typ, DirectionHorizontal, field);    //benötigte Felder auf Status Schiff
                                                                                                 setzen
164.                                    shipCount++;
165.                                    setShipBorder(x, y, typ, DirectionHorizontal, field);    //umliegende Felder auf Status
                                                                                                 Schiffrand setzen
166.                            }
167.                            else ss.drawInstruction2("Schiff kann nicht platziert werden, da es ein anderes Schiff kreuzt oder
                                                    berührt!", ShipShot.insAlert);
168.            }
169.           }
170.           else ss.drawInstruction2("Anzahl der Schiffe erreicht!", ShipShot.insStd);
171.           ss.drawInstruction1("Anzahl gesetzter Schiffe: " +shipCount, ShipShot.insStd);
172.          //drawShipOK auf true setzen, sobald alle Schiffe gesetzt sind
173.          if (shipCount < shipAnz)
174.                 drawShipOK=false;
175.          else
176.                 drawShipOK=true;
177.          }
178.
179.          //Schiffe vom PC werden gezeichnet
180.          public void drawShip_PC()
181.          {
182.                 int randX, randY, randDirection, typ;
183.                 g.setColor(Color.cyan);
184.                 //System.out.println("\n*************** Schiffe des PCs werden platziert ***************");
185.                 for (int i=0; shipCount<shipAnz; i++)      //Insgesamt 5 Zufallsschiffe werden platziert
186.                 {
187.                        randX = (int)(10*Math.random());           //Zufallskoordinate zwischen 0 und 9
188.                        randY = (int)(10*Math.random());           //Zufallskoordinate zwischen 0 und 9
189.                        randDirection  =(int)(2*Math.random());    //Zufallszahl 0 oder 1 für Ausrichtung
190.                        //System.out.println("\n ..... " +i +"er Durchlauf .....");
191.                        typ = shipTyp();
192.                        //System.out.println("ShipCount vorher: " +shipCount);
193.                        //System.out.println("Zufallskoordinaten: x/y: " +randX +"/" +randY +" | Ausrichtung: " +randDirection +" |
                           Länge: " +ShipLength[typ]);
194.
195.                        if (randDirection == 0 )     //Ausrichtung Horizontal
196.                        {
197.                               if(enoughSpace(randX, randY, typ, true, field_PC))//wenn alle Feld-Stati 9/Meer dann kann Schiff
                                                                                    platziert werden
198.                               {
199.                                      //Schiffe des PCs sichtbar machen bzw. zeichnen - für Tests
```



```
200.                                   //g.fillRect(startX_PC+randX*abstand+15/2, abstand+randY*abstand+15/2, abstand*ShipLength[typ]-
                                       15, abstand-15);
201.                                   setShipFields(randX, randY, typ, true, field_PC); //benötigte Felder auf Status Schiff setzen
202.                                   shipCount++;
203.                                   setShipBorder(randX, randY, typ, true, field_PC); //umliegende Felder auf Status Schiffrand
                                                                                         setzen
204.                           }
205.                           else ss.drawInstruction2("Schiff kann nicht platziert werden, da es ein anderes Schiff kreuzt oder
                                                   berührt!", ShipShot.insAlert);
206.                   }
207.                   else //Ausrichtung vertikal
208.                   {
209.                           if(enoughSpace(randX, randY, typ, false, field_PC))      //wenn alle Feld-Stati 9/Meer dann kann
                                                                                        Schiff platziert werden
210.                           {
211.                                   //Schiffe des PCs sichtbar machen bzw. zeichnen - für Tests
212.                                   //g.fillRect(startX_PC+randX*abstand+15/2, abstand+randY*abstand+15/2, abstand-15,
                                       abstand*ShipLength[typ]-15);
213.                                   setShipFields(randX, randY, typ, false, field_PC);//benötigte Felder auf Status Schiff setzen
214.                                   shipCount++;
215.                                   setShipBorder(randX, randY, typ, false, field_PC);//umliegende Felder auf Status Schiffrand
                                                                                         setzen
216.                           }
217.                           else ss.drawInstruction2("Schiff kann nicht platziert werden, da es ein anderes Schiff kreuzt oder
                                                   berührt!", ShipShot.insAlert);
218.                   }
219.                   //System.out.println("ShipCount nachher: " +shipCount);
220.           }
221.           ss.drawInstruction2("Schiffe des PCs wurden erfolgreich platziert!", ShipShot.insStd);
222.           ss.drawInstruction1("", ShipShot.insStd);
223.           //drawShipOK_PC auf true setzen, sobald alle Schiffe gesetzt sind
224.          if (shipCount < shipAnz)
225.                 drawShipOK_PC=false;
226.          else
227.                 drawShipOK_PC=true;
228.          }
229.
230.          //Auswahl Schiff-Typ
231.          public int shipTyp()
232.          {
233.                 int typ=0;
234.                 switch (shipCount)
235.                 {
```

```
236.                        case 0:
237.                               typ = 0;
238.                               break;
239.                        case 1:
240.                               typ = 1;
241.                               break;
242.                        case 2:
243.                               typ = 2;
244.                               break;
245.                        case 3:
246.                               typ = 3;
247.                               break;
248.                        case 4:
249.                               typ = 4;
250.                               break;
251.                 }
252.                 return typ;
253.          }
254.
255.          //Prüft ob genügend Platz zum Platzieren der Schiffe vorhanden ist
256.          boolean enoughSpace(int x, int y, int typ, boolean DirectionHorizontal, int fieldEnoughSpace[][])
257.          {
258.                 boolean fieldEnoughSpace_free=true;
259.                 if (DirectionHorizontal)
260.          {
261.                        if (x+ShipLength[typ] <= 10) //Prüft ob das Schiff im Spielfeld platz hat
262.                        {
263.                               for (int i=x; i<x+ShipLength[typ]; i++)//Prüft ob die benötigten Felder für das Schiff noch frei sind
264.                         {
265.                               if(fieldEnoughSpace[i][y] == 9)     //wenn alle Feld-Stati 9/Meer dann kann Schiff platziert werden
266.                                      fieldEnoughSpace_free = true;
267.                               else
268.                               {
269.                                      fieldEnoughSpace_free = false;
270.                                      break;
271.                               }
272.                         }
273.                               return fieldEnoughSpace_free;
274.                        }
275.                        else
276.                        {
277.                               ss.drawInstruction2("Schiff kann nicht platziert werden, da nicht genuegend Platz im Spielfeld
                                             vorhanden ist!", ShipShot.insAlert);
```



```
278.                               return false;
279.                        }
280.          }
281.                 else   //Vertikal
282.                 {
283.                        if (y+ShipLength[typ] <= 10) //Prüft ob das Schiff im Spielfeld platz hat
284.                        {
285.                               for (int i=y; i<y+ShipLength[typ]; i++)//Prüft ob die benötigten Felder für das Schiff noch frei sind
286.                  {
287.                        if(fieldEnoughSpace[x][i] == 9)     //wenn alle Feld-Stati 9/Meer dann kann Schiff platziert werden
288.                               fieldEnoughSpace_free = true;
289.                        else
290.                        {
291.                               fieldEnoughSpace_free = false;
292.                               break;
293.                        }
294.                  }
295.                               return fieldEnoughSpace_free;
296.                        }
297.                        else
298.                        {
299.                               ss.drawInstruction2("Schiff kann nicht platziert werden, da nicht genuegend Platz im Spielfeld
                                             vorhanden ist!", ShipShot.insAlert);
300.                               return false;
301.                        }
302.                 }
303.          }
304.
305.          //Setzt die Felder des Schiffes auf Status Schiff - Status abhängig vom Schiff-Typ
306.          public void setShipFields(int x, int y, int typ, boolean DirectionHorizontal, int fieldShip[][])
307.          {
308.                 if (DirectionHorizontal)
309.                 {
310.                        for (int r=0; r<ShipLength[typ]; r++)      //benötigte Felder auf Status Schiff setzen
311.                        {
312.                               fieldShip[x+r][y] = shipCount;      //Status Schiff je nach Typ: 1. DL 0, 2. DL 1 ... 4. DL 4
313.                        }
314.                 }
315.                 else
316.                 {
317.                        for (int r=0; r<ShipLength[typ]; r++)      //benötigte Felder auf Status Schiff setzen
318.                 {
319.                        fieldShip[x][y+r] = shipCount;             //Status Schiff je nach Typ: DL 0, 2. DL 1 ... 4. DL 4
```

```
320.                }
321.                }
322.          }
323.
324.          //Setzt die umliegenden Felder auf Status Schiffrand
325.          public void setShipBorder(int x, int y, int typ, boolean DirectionHorizontal, int fieldBorder[][])
326.          {
327.                if (DirectionHorizontal)
328.                {
329.                      for (int r=0; r<ShipLength[typ]; r++)
330.                      {
331.                            if (x == 0)          //Schiff beginnt am linken Rand
332.                            {
333.                                  fieldBorder[x+ShipLength[typ]][y] = 7;     //Schiffrand hinter dem Schiff
334.                            }
335.                            else
336.                            {
337.                                  if (x+ShipLength[typ]-1 == 9)                    //Schiff endet am rechten Rand
338.                                  {
339.                                        fieldBorder[x-1][y] = 7;                   //Schiffrand vor dem Schiff
340.                                  }
341.                                  else          //Schiff in der Mitte
342.                                  {
343.                                        fieldBorder[x+ShipLength[typ]][y] = 7;     //Schiffrand hinter dem Schiff
344.                                        fieldBorder[x-1][y] = 7;                   //Schiffrand vor dem Schiff
345.                                  }
346.                            }
347.                            if (y == 0)          //Schiff ganz oben
348.                            {
349.                                  fieldBorder[x+r][y+1] = 7;          //Schiffrand unterhalb vom Schiff
350.                            }
351.                            else
352.                            {
353.                                  if (y == 9)          //Schiff ganz unten
354.                                  {
355.                                        fieldBorder[x+r][y-1] = 7;          //Schiffrand oberhalb vom Schiff
356.                                  }
357.                                  else          //Schiff in der Mitte
358.                                  {
359.                                        fieldBorder[x+r][y+1] = 7;   //Schiffrand unterhalb vom Schiff
360.                                  fieldBorder[x+r][y-1] = 7;          //Schiffrand oberhalb vom Schiff
361.                                  }
362.                            }
```



```
363.                      }
364.                }
365.                else   //horizontal
366.                {
367.                      for (int r=0; r<ShipLength[typ]; r++)
368.                      {
369.                            if (y == 0)          //Schiff beginnt am oberen Rand
370.                            {
371.                                  fieldBorder[x][y+ShipLength[typ]] = 7;            //Schiffrand unterhalb vom Schiff
372.                            }
373.                            else
374.                            {
375.                                  if (y+ShipLength[typ]-1 == 9)                    //Schiff endet am unteren Rand
376.                                  {
377.                                        fieldBorder[x][y-1] = 7;                   //Schiffrand oberhalb vom Schiff
378.                                  }
379.                                  else
380.                                  {
381.                                        fieldBorder[x][y-1] = 7;                   //Schiffrand oberhalb vom Schiff
382.                                        fieldBorder[x][y+ShipLength[typ]] = 7;     //Schiffrand unterhalb vom Schiff
383.                                  }
384.                            }
385.                            if (x == 0)          //Schiff am linken Rand
386.                            {
387.                                  fieldBorder[x+1][y+r] = 7;          //Schiffrand rechts vom Schiff
388.                            }
389.                            else
390.                            {
391.                                  if (x == 9)          //Schiff am rechten Rand
392.                                  {
393.                                        fieldBorder[x-1][y+r] = 7;          //Schiffrand links neben dem Schiff
394.                                  }
395.                                  else
396.                                  {
397.                                        fieldBorder[x+1][y+r] = 7;          //Schiffrand rechts vom Schiff
398.                                        fieldBorder[x-1][y+r] = 7;          //Schiffrand links neben dem Schiff
399.                                  }
400.                            }
401.                      }
402.                }
403.          }
404.    }
```

5.3. ShipShot.java

```
1.  /* Applet "Schiffe versenken" - ShipShot.java
2.   * Fachakademie für Angewandte Informatik
3.   * Softwaredeveloper - Java
4.   * (C) Ing. Hannes Gastl, 2012
5.   */
6.
7.  package SchiffeVersenken;
8.
9.  import java.awt.*;
10. import java.awt.event.*;
11.
12. public class ShipShot extends MouseAdapter
13. {
14.     private Graphics g;
15.     int startX_PC = 300;  //2. Spielfeld
16.     int startY = 25;
17.     int anz = 10;
18.     int abstand = 25;
19.     boolean player;                     //true für Player; false für PC
20.     private Color shot;
21.     static Color insStd = Color.black, insAlert = new Color(220,0,0);
22.     int hits_Player=0, hits_PC=0, shots_Player=0, shots_PC=0, shipSunk_Player=0, shipSunk_PC=0;
23.     int shipAnz = BattleShipField.ShipLength.length;            //=5  Gesamt-Anzahl der Schiffe
24.     int hitShipCounter[] = new int [shipAnz];               //Zähler für die Treffer je Schifftyp
25.     int hitShipCounter_PC[] = new int [shipAnz];            //Zähler für die Treffer je Schifftyp
26.     boolean end = false, win_Player = false;
27.     static boolean playLevel2;          //Variable zum Setzen des zweiten Levels
28.
29.     //Konstruktor
30.     public ShipShot(Graphics g)
31.      {
32.            this.g=g;
33.      }
34.
35.     //Liefert X-Koordinate des Mausklicks
36.     public int getX(int posX)
37.     {
38.        if ((posX >= startX_PC) && (posX <= startX_PC+anz*abstand) ) return posX;
39.        else return 0;    //Rückgabe 0 wenn ausserhalb des Feldes
40.     }
41.
```



```
42.     //Liefert Y-Koordinate des Mausklicks
43.     public int getY(int posY)
44.     {
45.            if ((posY >= startY) && (posY <= startY+anz*abstand) ) return posY;
46.            else return 0;//Rückgabe 0 wenn ausserhalb des Feldes
47.     }
48.
49.     //Maus Event zum Schuss setzen des Spielers
50.     public void mouseClicked(MouseEvent event)
51.     {
52.            int fieldX=(getX(event.getX())/abstand)-1-11;     // Div. durch Abstand damit man auf die Kästchen-Nummerierung kommt / -11
                                                                 damit das Feld bei 0/0 anfängt (2. Spielfeld)
53.         int fieldY=(getY(event.getY())/abstand)-1;           // Div. durch Abstand damit man auf die Kästchen-Nummerierung kommt    /
                                                                 -1 damit das Feld bei 0/0 anfängt
54.         player = true;                                       //Als Spieler Player authentifizieren
55.
56.         if (end != true)
57.            {
58.                if ((fieldX >= 0) && (fieldY >= 0))
59.                {
60.                   drawInstruction1("Schuss Player auf: " +fieldX +"/" +fieldY, insStd);
61.                   drawShot(fieldX, fieldY, player, g);
62.                }
63.                else drawInstruction1("Schuss Player ausserhalb des Spielfeldes!", insAlert);
64.            }
65.         else System.out.println("Spiel beendet!");
66.     }
67.
68.     //Zufalls-Schuss setzen des PCs
69.     public void shot_PC()
70.     {
71.            int randX, randY;
72.            randX = (int)(10*Math.random());          //Zufallskoordinate zwischen 0 und 9
73.            randY = (int)(10*Math.random());          //Zufallskoordinate zwischen 0 und 9
74.            player = false;                                      //Als PC authentifizieren
75.            if (end != true)
76.            {
77.                   if ((randX >= 0) && (randY >= 0))
78.                {
79.                         drawInstruction2("Schuss PC auf: " +randX +"/" +randY +" ---> Bitte nächsten Schuss platzieren!", insStd);
80.                   drawShot(randX, randY, player, g);
81.                }
82.                else drawInstruction2("Schuss PC ausserhalb des Spielfeldes!", insAlert);
```

```
83.             }
84.             else System.out.println("Spiel beendet!");
85.     }
86.
87.     //Schuss malen
88.     public void drawShot(int x, int y, boolean player, Graphics g)
89.     {
90.             shotColor(x, y, player);     //Farbe abhängig vom Feld-Status
91.             g.setColor(shot);
92.             boolean lastShotHitShip = false; // für Level 2 -> wird true wenn letzter Treffer des PCs ein Schiff war
93.
94.             if (player)
95.             {
96.                     switch (BattleShipField.getField_PC_Element(x, y))
97.                     {
98.                             case 0:
99.                             case 1:
100.                                    case 2:
101.                                    case 3:
102.                                    case 4:
103.                                            BattleShipGUI.soundHitShip.play();
104.                                            hitShip(x, y, player);
105.                                            break;
106.                                    case 8: //Schuss
107.                                            drawInstruction1("Schuss Player kann nicht platziert werden, da bereits ein Schuss auf dieses
                                                        Feld abgesetzt wurde!", insAlert);
108.                                            break;
109.                                    case 7:
110.                                    case 9://Meer
111.                                            BattleShipGUI.soundHitWater.play();
112.                                            g.fillOval(startX_PC+x*abstand+(abstand/4), abstand+y*abstand+(abstand/4), abstand/2,
                                                        abstand/2);
113.                                            BattleShipField.setField_PC_Element(x, y, 8);            //Feldstatus auf 8 (Schuss) setzen
114.                                            shots_Player++;
115.                                            drawScore();
116.                                            shot_PC();           //PC schiessen lassen
117.                                            break;
118.                                    default: System.out.println("FAILURE: ShipShot.drawShot Feldstatus: "
                                                        +BattleShipField.getField_PC_Element(x, y));
119.                    }
120.            }
121.            else          //Computer
122.            {
```



```
123.                    switch (BattleShipField.getField_Element(x, y))
124.                    {
125.                            case 0:
126.                            case 1:
127.                            case 2:
128.                            case 3:
129.                            case 4:
130.                                    try {
131.                                            Thread.sleep(1000);                //Pause zwischen Schuss vom Player und dem PC
132.                                    } catch(InterruptedException e) {}
133.                                    BattleShipGUI.soundHitShip.play();
134.                                    hitShip(x, y, player);
135.                                    lastShotHitShip = true;
136.                                    if (playLevel2) level2(x, y, lastShotHitShip);    //Wenn Level2 aktiv, dann weitere Treffer
                                                platzieren
137.                                    break;
138.                            case 8: //Schuss
139.                                    drawInstruction2("Schuss PC kann nicht platziert werden, da bereits ein Schuss auf dieses Feld
                                                abgesetzt wurde!", insAlert);
140.                                    shot_PC();           //PC nochmals schiessen lassen, da Zufallskoordinaten auf ein bereits
                                                beschossenes Feld zeigten
141.                                    break;
142.                            case 7:
143.                            case 9: //Meer
144.                                    try {
145.                                            Thread.sleep(1000);                //Pause zwischen Schuss vom Player und dem PC
146.                                    } catch(InterruptedException e) {}
147.                                    BattleShipGUI.soundHitWater.play();
148.                                    g.fillOval(abstand+x*abstand+(abstand/4), abstand+y*abstand+(abstand/4), abstand/2, abstand/2);
149.                                    BattleShipField.setField_Element(x, y, 8);              //Feldstatus auf 8 (Schuss) setzuen
150.                                    shots_PC++;
151.                                    drawScore();
152.                                    break;
153.                            default: System.out.println("FAILURE: ShipShot.drawShot Feldstatus: "
                                                +BattleShipField.getField_PC_Element(x, y));
154.                    }
155.            }
156.    }
157.
158.    //richtige Farbe des Schusses in Abhängigkeit des Status festlegen
159.    public void shotColor(int x, int y, boolean player)
160.    {
161.            if (player)
```

```
162.                {
163.                       switch (BattleShipField.getField_PC_Element(x, y))
164.                       {
165.                              case 7:
166.                              case 9:
167.                                     shot = Color.blue;
168.                                     break;
169.                              case 0:
170.                              case 1:
171.                              case 2:
172.                              case 3:
173.                              case 4:
174.                                     shot = Color.red;
175.                                     break;
176.                              default:
177.                                     System.out.println("FAILURE: ShipShot.shotColor! Feldstatus: "
                                                  +BattleShipField.getField_PC_Element(x, y) +"! ");
178.                                     shot = Color.black;
179.                       }
180.                }
181.                else
182.                {
183.                       switch (BattleShipField.getField_Element(x, y))
184.                       {
185.                              case 7:
186.                              case 9:
187.                                     shot = Color.blue;
188.                                     break;
189.                              case 0:
190.                              case 1:
191.                              case 2:
192.                              case 3:
193.                              case 4:
194.                                     shot = Color.red;
195.                                     break;
196.                              default:
197.                                     System.out.println("FAILURE: ShipShot.shotColor! Feldstatus: "
                                                  +BattleShipField.getField_Element(x, y) +"! ");
198.                                     shot = Color.black;
199.                       }
200.                }
201.         }
202.
```



```
203.         //Kontrolliert ob Schiff versenkt ist, setzt die Feldstati, zeichnet den Schuss und prüft ob Spiel gewonnen
204.         public void hitShip(int x, int y, boolean player)
205.         {
206.                if (player)
207.                {
208.                       g.fillOval(startX_PC+x*abstand+(abstand/4), abstand+y*abstand+(abstand/4), abstand/2, abstand/2);
209.                       shots_Player++;
210.                       hits_Player++;
211.                       hitShipCounter_PC[BattleShipField.getField_PC_Element(x, y)]++;        //Zähler des getroffenen Schiffes wird
                                                                                                  um 1 erhöht
212.                       if(hitShipCounter_PC[BattleShipField.getField_PC_Element(x, y)] ==
                                  BattleShipField.getShipLength(BattleShipField.getField_PC_Element(x, y)))            //Schiff
                                  versenkt: Wenn Anzahl der Treffer und Länge des Schiff-Typs gleich sind
213.                       {
214.                              shipSunk_PC++;
215.                              if (shipSunk_PC < shipAnz)
216.                              {
217.                                     drawInstruction1("=============== Player hat Schiff vom PC versenkt ===============",
                                                  insAlert);
218.                                     System.out.println("=============== Schiff vom PC versenkt ================");
219.                              }
220.                              else
221.                              {
222.                                     win_Player = true;
223.                                     drawInstruction1("=============== Player hat Schiff vom PC versenkt ===============",
                                                  insAlert);
224.                                     drawInstruction2("=================== Player hat gewonnen! ====================", insAlert);
225.                                     System.out.println("=============== Schiff vom PC versenkt ================");
226.                                     System.out.println("================ Player hat gewonnen! =================");
227.                                     outputWin(win_Player);
228.                                     BattleShipGUI.soundWin.play();
229.                                     end = true;    //Ende des Spiels auf true setzen
230.                              }
231.                       }
232.                       else drawInstruction1("Player hat Schiff vom PC getroffen aber noch nicht versenkt!", insStd);
233.                       BattleShipField.setField_PC_Element(x, y, 8);            //Feldstatus auf 8 (Schuss) setzen
234.                       drawScore();
235.                }
236.                else   //Computer
237.                {
238.                       g.fillOval(abstand+x*abstand+(abstand/4), abstand+y*abstand+(abstand/4), abstand/2, abstand/2);
239.                       shots_PC++;
240.                       hits_PC++;
```

```
241.                    hitShipCounter[BattleShipField.getField_Element(x, y)]++;//Zähler des getroffenen Schiffes wird um 1 erhöht
242.                    if(hitShipCounter[BattleShipField.getField_Element(x, y)] ==
                               BattleShipField.getShipLength(BattleShipField.getField_Element(x, y)))        //Schiff versenkt: Wenn
                                       Anzahl der Treffer und Länge des Schiff-Typs gleich sind
243.                    {
244.                           shipSunk_Player++;
245.                           if (shipSunk_Player < shipAnz)
246.                           {
247.                                  drawInstruction1("=============== PC hat Schiff vom Player versenkt ===============",
                                              insAlert);
248.                                  System.out.println("=========== Schiff vom Player versenkt ================");
249.                                  if (playLevel2) shot_PC();          //PC nochmals schiessen lassen, da Schiff getroffen
250.                           }
251.                           else
252.                           {
253.                                  win_Player = false;
254.                                  drawInstruction1("=============== PC hat Schiff vom Player versenkt ===============",
                                              insAlert);
255.                                  drawInstruction2("=================== PC hat gewonnen! ========================", insAlert);
256.                                  System.out.println("=========== Schiff vom Player versenkt ================");
257.                                  System.out.println("=================== PC hat gewonnen! ====================");
258.                                  outputWin(win_Player);
259.                                  BattleShipGUI.soundWin.play();
260.                                  end = true;    //Ende des Spiels auf true setzen
261.                           }
262.                    }
263.                    else drawInstruction1("PC hat Schiff vom Player getroffen aber noch nicht versenkt!", insStd);
264.                    BattleShipField.setField_Element(x, y, 8);               //Feldstatus auf 8 (Schuss) setzen
265.                    drawScore();
266.                    if (playLevel2 == false) shot_PC();        //PC nochmals schiessen lassen, da Schiff getroffen
267.             }
268.      }
269.
270.      //Spiel Level 2 - PC schaut bei Treffer wo das nächste Feld des Schiffes ist und versenkt gleich das ganze Schiff :-)
271.      public void level2(int x, int y, boolean lshs)
272.      {
273.             int newX = x;
274.             int newY = y;
275.             if (lshs)
276.             {
277.                    for (int i=0; i<4; i++)            //Schaut die Felder rund um den letzten Treffer an
278.                    {
279.                           switch (i)
```



```
280.                           {
281.                                  case 0:
282.                                         newX = x;
283.                                         newY = y+1;
284.                                         break;
285.                                  case 1:
286.                                         newX = x;
287.                                         newY = y-1;
288.                                         break;
289.                                  case 2:
290.                                         newX = x+1;
291.                                         newY = y;
292.                                         break;
293.                                  case 3:
294.                                         newX = x-1;
295.                                         newY = y;
296.                                         break;
297.                           }
298.                           if ((newX >= 0 && newX <= 9) && (newY >= 0 && newY <= 9))//wenn die neuen Koordinaten zwischen 0 und 9
                                                                                          liegen
299.                           {
300.                                  if ((BattleShipField.getField_Element(newX, newY) == 0) ||
301.                                                (BattleShipField.getField_Element(newX, newY) == 1) ||
302.                                                (BattleShipField.getField_Element(newX, newY) == 2) ||
303.                                                (BattleShipField.getField_Element(newX, newY) == 3) ||
304.                                                (BattleShipField.getField_Element(newX, newY) == 4))     //wenn neue Koordinaten
                                                                                                  Treffer sind, setze Schuss
305.                                  {
306.                                         if (end != true)
307.                                         {
308.                                                if ((newX >= 0) && (newY >= 0))
309.                                            {
310.                                                       drawInstruction2("Schuss PC auf: " +newX +"/" +newY +" ---> Bitte nächsten
                                                                   Schuss platzieren!", insStd);
311.                                                drawShot(newX, newY, player, g);
312.                                            }
313.                                            else drawInstruction2("Schuss PC ausserhalb des Spielfeldes!", insAlert);
314.                                         }
315.                                         else System.out.println("Spiel beendet!");
316.                                  }
317.                                  else   //neuer Durchlauf um das nächste Nachbar-Feld anszuschauen
318.                                  {
319.                                         newX = x;
```

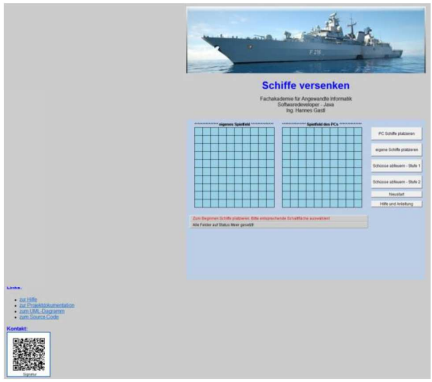
```
320.                                            newY = y;
321.                                            //System.out.println("Level 2: Nächstes Feld kein Treffer! " +i +"ter Versuch!");
322.                                    }
323.                             }
324.                             else
325.                             {
326.                                    newX = x;
327.                                    newY = y;
328.                             }
329.                     }
330.              }
331.       }
332.
333.       //Textfeld für Verständigung des Gewinners
334.       public void outputWin(boolean win_Player)
335.       {
336.           g.setColor(new Color(204,204,204));
337.           g.fill3DRect(10, 300, 560, 40, true);
338.           g.setFont(new Font("SansSerif", Font.PLAIN, 34));
339.              g.setColor(Color.red);
340.              if (win_Player)
341.                     g.drawString("Player hat gewonnen!", 120, 330);
342.              else
343.                     g.drawString("PC hat gewonnen!", 160, 330);
344.       }
345.
346.       //Gibt die Spiel-Instruktionen aus -> Zeile 1
347.       public void drawInstruction1(String str, Color c)
348.       {
349.              int l=14, x=20, y=300;
350.           g.setColor(new Color(204,204,204));
351.           g.fill3DRect(10, y, 560, 20, true);
352.              g.setFont(new Font("SansSerif", Font.PLAIN, l-2));
353.              g.setColor(c);
354.              g.drawString(str, x, y+l);
355.       }
356.
357.       //Gibt die Spiel-Instruktionen aus -> Zeile 2
358.       public void drawInstruction2(String str, Color c)
359.       {
360.              int l=14, x=20, y=320;
361.              g.setColor(new Color(204,204,204));
362.           g.fill3DRect(10, y, 560, 20, true);
```



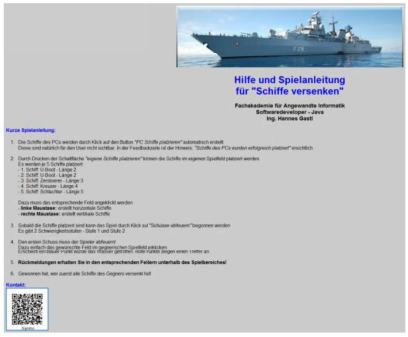
```
363.              g.setFont(new Font("SansSerif", Font.PLAIN, l-2));
364.              g.setColor(c);
365.              g.drawString(str, x, y+l);
366.       }
367.
368.       //Gibt die Zwischenergebnisse aus
369.       public void drawScore()
370.       {
371.              int l=14, x=20, y=350;
372.              g.setColor(new Color(255,204,102));
373.              g.fill3DRect(10, y, 560, 120, true);
374.              g.setFont(new Font("SansSerif", Font.BOLD, l-2));
375.              g.setColor(Color.black);
376.              g.drawString("*************** aktueller Spielstand ***************", 20 , y+l);
377.              g.setFont(new Font("SansSerif", Font.PLAIN, l-2));
378.              g.drawString("Treffer PC:", x, y+l*3);            g.drawString(""+hits_PC, x+95, y+l*3);
379.              g.setColor(Color.blue);
380.              g.drawString("Treffer Player:", x, y+l*4); g.drawString(""+hits_Player, x+95, y+l*4);
381.              g.setColor(Color.black);
382.              g.drawString("Schüsse PC:", x, y+l*5);            g.drawString(""+shots_PC, x+95, y+l*5);
383.              g.setColor(Color.blue);
384.              g.drawString("Schüsse Player:", x, y+l*6); g.drawString(""+shots_Player, x+95, y+l*6);
385.              g.setColor(Color.black);
386.              g.drawString("versenkte Schiffe vom PC:", x, y+l*7);            g.drawString(""+shipSunk_PC, x+180, y+l*7);
387.              g.setColor(Color.blue);
388.              g.drawString("versenkte Schiffe vom Player:", x, y+l*8); g.drawString(""+shipSunk_Player, x+180, y+l*8);
389.       }
390.
391.       /* für interne Tests
392.       public static void drawShipHits()
393.       {
394.              System.out.println("\n*************** aktuelle Treffer je Schiff ***************");
395.              for (int i=0; i<shipAnz; i++)
396.              {
397.                     System.out.println("Player Schiff Typ: " +i +"\t| Treffer: " +hitShipCounter[i]);
398.                     System.out.println("PC Schiff Typ: " +i +"\t| Treffer: " +hitShipCounter_PC[i]);
399.              }
400.              System.out.println("\n************* aktuelle Treffer je Schiff ENDE ***************\n");
401.       }
402.       */
403.   }
```

5.4. index.html

```
<html>
<head>
  <title>Schiffe versenken</title>
  <style>
    body {
    style=font-size: 14px;
    font-family:Arial;
    color:#000000; }
  </style>
</head>
<body bgcolor="#CCCCCC">
  <div align="center"><img src="data/battleship.jpg"></div>
  <h1 align="center"><font color="#0000FF">Schiffe versenken</font></font></h1>
  <p align="center">
    Fachakademie für Angewandte Informatik<br>Softwaredeveloper - Java<br>Ing. Hannes Gastl</p>
  </p>
  <p align="center">
    <applet code="SchiffeVersenken.BattleShipGUI" codebase="bin" align="center" width="750" height="500"></applet>
  </p>
  <p>
    <b><font color="#0000FF">Links:</font></b>
    <ul>
      <li><a href="BattleShipHelp.html" target="_blank">zur Hilfe</a></li>
      <li><a href="data/projektdoku.pdf" target="_blank">zur Projektdokumentation</li>
      <li><a href="data/SchiffeVersenken.jpg" target="_blank">zum UML-Diagramm</a></li>
      <li><a href="data/code.pdf" target="_blank">zum Source-Code</a></li>
    </ul>
  </p>
  <p>
    <b><font color="#0000FF">Kontakt:</font></b><br>
      <a href="mailto:office@gastl.cc"><img src="data/qrcode.jpg"></a>
  </p>
</body>
</html>
```



5.5. BattleShipHelp.html




```
<html>
<head>
  <title>Schiffe versenken - Hilfe</title>
  <style>
    body {
    style=font-size: 14px;
    font-family:Arial;
    color:#000000; }
  </style>
</head>
<body bgcolor="#CCCCCC">
  <div align="center"><img src="data/battleship.jpg"> </div>
  <h1 align="center"><font color="#0000FF">Hilfe und Spielanleitung<br>für "Schiffe versenken"</font></font></h1>
  <h3 align="center">Fachakademie für Angewandte Informatik<br>Softwaredeveloper - Java<br>Ing. Hannes Gastl</p></h3>
  <p>
    <b><font color="#0000FF">Kurze Spielanleitung:</font></b>
    <ol>
      <li>Die Schiffe des PCs werden durch Klick auf den Button <em>"PC Schiffe platzieren"</em> automatisch erstellt<br>
      Diese sind natürlich für den User nicht sichtbar. In der Feedbackzeile ist der Hinweis: <em>"Schiffe des PCs wurden erfolgreich platziert"</em>
      ersichtlich.<br><br></li>
      <li>Durch Drücken der Schaltfläche <em>"eigene Schiffe platzieren"</em> können die Schiffe im eigenen Spielfeld platziert werden.<br>
      Es werden je 5 Schiffe platziert:<br>
        - 1. Schiff: U-Boot - Länge 2<br>
        - 2. Schiff: U-Boot - Länge 2<br>
        - 3. Schiff: Zerstoerer - Länge 3<br>
        - 4. Schiff: Kreuzer - Länge 4<br>
        - 5. Schiff: Schlachter - Länge 5<br></li><br>
        Dazu muss das entsprechende Feld angeklickt werden<br>
        - <b>linke Maustase:</b> erstellt horizontale Schiffe<br>
        - <b>rechte Maustase:</b> erstellt vertikale Schiffe<br><br></li>
      <li>Sobald die Schiffe platziert sind kann das Spiel durch Klick auf <em>"Schüsse abfeuern"</em> begonnen werden<br>
      Es gibt 2 Schwierigkeitsstufen - Stufe 1 und Stufe 2<br><br></li>
      <li>Den ersten Schuss muss der Spieler abfeuern!<br>
      Dazu einfach das gewünschte Feld im gegnerischen Spielfeld anklicken.<br>
      Erscheint ein blauer Punkt wurde das Wasser getroffen. Rote Punkte zeigen einen Treffer an.<br><br></li>
      <li><b>Rückmeldungen erhalten Sie in den entsprechenden Feldern unterhalb des Spielbereiches!</b><br><br></li>
      <li>Gewonnen hat, wer zuerst alle Schiffe des Gegners versenkt hat!<br></li>
    </ol>
  </p>
  <p>
    <b><font color="#0000FF">Kontakt:</font></b><br>
      <a href="mailto:office@gastl.cc"><img src="data/qrcode.jpg"></a>
  </p>
</body>
</html>
```