Leistungsnachweis im Fach Programmierung 1

Maximilian von Hohenbühel
Fabian Cieslik

2020-12-07 - 2020-01-17

Contents

1 Projektvorraussetzungen

Beschreibung: Die Projektaufgabe besteht darin, ein einfaches Spiel zu implementieren. Die Wahl des Spiels bleibt Ihnen überlassen, beachten Sie jedoch, dass sich im Rahmen von 36 Stunden Arbeitszeit nur sehr begrenzte Spielideen auch umsetzen lassen.
Details: Sie programmieren ein Spiel für ein sehr eingeschränktes Display. Dieses enthält nur 24×48 Bildpunkte (Pixel), d.h. 24 Reihen mit jeweils 48 Spalten. Jeder Bildpunkt kann 16 Millionen Farben annehmen, wobei die Rot, Grün und Blau-Komponente mit jeweils einem Byte angesprochen wird. Als Steuermöglichkeit stehen Ihnen vier Tasten zur Verfügung, die wie im Cursorblock üblich angeordnet sind. Es gibt nur einen Spieler. Die Zeit für eine Spielrunde sollte bei 20-30 Sekunden liegen.
Zur Ein- und Ausgabe erhalten Sie eine Klasse mit zwei Methoden:

- *public int getKeyboard()*
  Liefert die vier Cursortasten der Tastatur folgende Werte zurück:
  - 0 -> "hoch"
  - 1 -> "runter"
  - 2 -> "links"
  - 3 -> "rechts"
  - -1 -> keine Taste

- *public void showImage(short[] image)*
  Zeigt ein komplettes Bild auf dem Display an, wobei der erste Wert des Arrays die Rot-Komponente des linken oben Bildpunkts ist und der letzte Wert die Blau-Komponente von 0 bis 255 des rechten unteren Bildpunktes. Das übergebene Array muss exakt $24 * 48 * 3$ Elemente haben für die 24 Zeilen, 48 Spalten und 3 Farbkomponenten pro Pixel. Das Display wird zeilenweise durchlaufen.

Spielumfang:

- Eine *interaktive Spielerfigur*
- Eine *automatisch gesteuerte Spielerfigur*
- Einen Hintergrund
- Ein *Score-System*
- Ein *Highscore-System*
- Implementierungsvorgaben:
  - Eine generische Klasse
  - Drei davon abgeleitete Klassen (Spieler, Hintergrund, Gegner/NPC)

2 Idee

Name:
MP - Mari proelium
Spiel:
Der Spieler steuert ein Schiff und probiert so lang zu überleben wie möglich. Es existieren Gegner, die sich zufällig bewegen.
Das Spiel wird in einer Vogelperspektive gespielt, man hat dadurch jederzeit den Überblick der gesamten Karte.
Das Spielprinzip der Runden wird in dem Sinne implementiert, dass alle 30 Sekunden neue Gegner auftauchen und man für jede überlebte Runde zusätzliche Punkte bekommt.
Das Punktesystem wird von der Zeit, die man am Leben ist und der Anzahl der besiegten feindlichen Schiffe beeinflusst.
Die Steuerung wird auf die vier verfügbaren Tasten aufgeteilt, sodass man ohne Probleme sein Schiff steuern kann und zugleich auch schiessen kann. Die Kollisionsinteraktionen mit feindlichen Schiffen und eventuellen Häfen wird vom System übernommen.

3 Beschreibung

Karte:

- **Aussehen**
  Die Karte hat einen blauen Hintergrund, der den Ozean darstellt. Auf dem Ozean kann es vorkommen, dass es verschiedene Inseln geben kann. Inseln werden als braune oder grüne Pixel dargestellt. Die Platzierung der Inseln wird zufällig am Anfang des Spiels festgelegt und wird nur bei einem kompletten Neustart verändert. Auf den Inseln können Häfen generiert werden, die einem einen Bonus geben, falls man sie erreichen sollte.

Spieler:

- **Aussehen**
  Ein zwei bzw. drei Pixel langes Schiff.

- **Fähigkeiten**
  - Links: 45 Grad Drehung gegen den Uhrzeigersinn
  - Rechts: 45 Grad Drehung in den Uhrzeigersinn
  - Oben: Vorwärts Bewegung nach vorne
  - Unten: Benutzen der Schiffsinternen Kanone

Gegner:

- **Aussehen**
  Ein zwei bzw. drei Pixel langes Schiff.

- **Fähigkeiten**
  - Zufälliges Bewegen auf der Karte
  - Bei Spielersicht wird Geschossen

Punkte:

- **Punktequellen**
  - Beim treffen eines Gegners
  - Beim besiegen eines Gegners
  - Besiegen aller gerade lebender Geger
  - Fürs überleben einer Runde

- **Highscore**
  Punkte werden in der Konsole als Highscore nach jedem Tod des Spielers ausgegeben

4 Programmablauf

- **Vorbereitung**
  Es werden alle Spielnotwendigen Variablen deklariert und initialisiert. In einer *Do-While* Schleife wird daraufhin gestarted um mehrere Spiele hintereinander spielen zu Können. Am Start der Schleife wird die Karte, der Spieler und die Gegner erstellt und gezeichnet und auf eine Eingabe des Benutzers gewartet. Bei Eingabe wird der Spielablauf gestartet. Nach dem Tod des Spielers wird der Punktestand ausgegeben und die Möglichkeit geboten ein neues Spiel zu Starten.

- **Spielablauf**
  Zuerst wird der Spieler bewegt und auf Kollisionen überprüft, danach die Gegner. Anschließend wird überprüft ob man eine Runde überlebt hat.

5 Klassendiagramm

6 Programmcode

Ship.java

```
1  public class Ship extends Agent {
2      protected int hp;
3      protected int[][] pos;
4      protected int[][] oldpos;
5      protected int align;
6      protected short[][][] color = new short[3][3][3];
7      protected Bullet bullet = null;
8      protected Bullet bullet2 = null;
9
10     Ship(int hp){
11         this.pos = new int[3][2];
12         this.oldpos = new int[3][2];
13         this.hp = hp;
14         this.align = 7;
15         this.pos[0][0] = 2;
16         this.pos[0][1] = 3;
17         this.pos[1][0] = 2;
18         this.pos[1][1] = 2;
19         this.pos[2][0] = 2;
20         this.pos[2][1] = 1;
21     }
22
23     Ship(int hp, int x, int y, int orient){
24         this.pos = new int[3][2];
25         this.oldpos = new int[3][2];
26         this.hp = hp;
27         if (x >= 1 && x <= 46 && y >= 1 && y <= 22) {
28             this.align = orient;
29             this.pos[1][0] = x;
30             this.pos[1][1] = y;
31             switch(this.align){
32                 case 1:
33                     this.pos[0][0] = x;
34                     this.pos[0][1] = y - 1;
35                     this.pos[2][0] = x;
36                     this.pos[2][1] = y + 1;
37                     break;
38                 case 2:
39                     this.pos[0][0] = x + 1;
40                     this.pos[0][1] = y - 1;
41                     this.pos[2][0] = x - 1;
42                     this.pos[2][1] = y + 1;
43                     break;
44                 case 3:
45                     this.pos[0][0] = x + 1;
46                     this.pos[0][1] = y;
```

```
47                  this.pos[2][0] = x - 1;
48                  this.pos[2][1] = y;
49                  break;
50              case 4:
51                  this.pos[0][0] = x + 1;
52                  this.pos[0][1] = y + 1;
53                  this.pos[2][0] = x - 1;
54                  this.pos[2][1] = y - 1;
55                  break;
56              case 5:
57                  this.pos[0][0] = x;
58                  this.pos[0][1] = y + 1;
59                  this.pos[2][0] = x;
60                  this.pos[2][1] = y - 1;
61                  break;
62              case 6:
63                  this.pos[0][0] = x - 1;
64                  this.pos[0][1] = y + 1;
65                  this.pos[2][0] = x + 1;
66                  this.pos[2][1] = y - 1;
67                  break;
68              case 7:
69                  this.pos[0][0] = x - 1;
70                  this.pos[0][1] = y;
71                  this.pos[2][0] = x + 1;
72                  this.pos[2][1] = y;
73                  break;
74              case 8:
75                  this.pos[0][0] = x - 1;
76                  this.pos[0][1] = y - 1;
77                  this.pos[2][0] = x + 1;
78                  this.pos[2][1] = y + 1;
79                  break;
80          }
81      }else {
82          this.pos[0][0] = 2;
83          this.pos[0][1] = 3;
84          this.pos[1][0] = 2;
85          this.pos[1][1] = 2;
86          this.pos[2][0] = 2;
87          this.pos[2][1] = 1;
88          this.align = 7;
89      }
90  }
91
92  protected short[] clearTrace(short[] myImage){
93      for (int i = 0; i < this.oldpos.length; i++) {
94          myImage[(this.oldpos[i][1] * 48 + this.oldpos[i][0]) * 3 + 0]
                = (short)0;
95          myImage[(this.oldpos[i][1] * 48 + this.oldpos[i][0]) * 3 + 1]
```

```
                = (short)177;
96          myImage[(this.oldpos[i][1] * 48 + this.oldpos[i][0]) * 3 + 2]
                = (short)241;
97      }
98      return myImage;
99  }
100
101 /**
102  * This method uses the Players values to update the map and return
        it always.
103  * @param myImage the Pixel array given from the {@link GameMain}
104  * @return the updated maparray
105  * */
106 public short[] paint(short[] myImage){
107     myImage = clearTrace(myImage);
108     if(this.hp > 0) {
109         for(int i=0; i < this.pos.length; i++){
110             myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 0] =
                    color[this.hp - 1][i][0];
111             myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 1] =
                    color[this.hp - 1][i][1];
112             myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 2] =
                    color[this.hp - 1][i][2];
113         }
114     }else {
115         for(int i=0; i < this.pos.length; i++){
116             myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 0] =
                    0;
117             myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 1] =
                    177;
118             myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 2] =
                    241;
119         }
120         if(this.bullet != null){
121             myImage = this.bullet.clear(myImage);
122         }
123     }
124     return myImage;
125 }
126
127 public short[] isHit(short[] myImage){
128     for(int i = 0; i < this.pos.length; i++){
129         if (hitBullet(myImage, this.pos[i][0], this.pos[i][1])){
130             damage(1);
131         }
132     }
133     myImage = paint(myImage);
134     return myImage;
135 }
136
```

```
137     public boolean isAlive(){
138         return (this.hp > 0);
139     }
140 
141     protected boolean comparePixel(short r1, short g1, short b1, short
            r2, short g2, short b2){
142         return (r1 == r2 && g1 == g2 && b1 == b2);
143     }
144 
145     /**
146      * The method collide looks at the pixels of the ship and look if it
              collided with another object
147      * */
148     public int collide(short[] myImage){
149         return -1;
150     }
151 
152     /**
153      * This method takes the userinput and changes the
              position/direction of the ship
154      * @param dir represents the given userinput
155      *          0 - up
156      *          1 - down
157      *          2 - left
158      *          3 - right
159      * */
160     protected void move(int dir, short[] myImage){
161         switch(dir){
162             case 0: // Hoch
163                 forward(myImage);
164                 break;
165             case 1: // Runter
166                 shoot();
167                 break;
168             case 2: // Links
169                 rotate(0);
170                 break;
171             case 3: // Rechts
172                 rotate(1);
173                 break;
174         }
175     }
176 
177     /**
178      * This method will be called by the move method and rotates the
              ship in the given direction.
179      * @param dir represents the direction which the ship takes to
              rotate.
180      *          0 - Left
181      *          1 - Right
```

```
182      * */
183     protected void rotate(int dir){
184         if(dir == 0){ // Left
185             switch(this.align){
186                 case 1:
187                     if (this.pos[0][0] - 1 >= 0 && this.pos[2][0] + 1 <=
                            47) {
188                         saveOldPos();
189                         this.pos[0][0]--;
190                         this.pos[2][0]++;
191                         changeAlign(-1);
192                     }
193                     break;
194                 case 2:
195                     if (this.pos[0][0] - 1 >= 0 && this.pos[2][0] + 1 <=
                            47) {
196                         saveOldPos();
197                         this.pos[0][0]--;
198                         this.pos[2][0]++;
199                         changeAlign(-1);
200                     }
201                     break;
202                 case 3:
203                     if (this.pos[0][1] - 1 >= 0 && this.pos[2][1] + 1 <=
                            23) {
204                         saveOldPos();
205                         this.pos[0][1]--;
206                         this.pos[2][1]++;
207                         changeAlign(-1);
208                     }
209                     break;
210                 case 4:
211                     if (this.pos[0][1] - 1 >= 0 && this.pos[2][1] + 1 <=
                            23) {
212                         saveOldPos();
213                         this.pos[0][1]--;
214                         this.pos[2][1]++;
215                         changeAlign(-1);
216                     }
217                     break;
218                 case 5:
219                     if (this.pos[0][0] + 1 <= 47 && this.pos[2][0] - 1 >=
                            0) {
220                         saveOldPos();
221                         this.pos[0][0]++;
222                         this.pos[2][0]--;
223                         changeAlign(-1);
224                     }
225                     break;
226                 case 6:
```

```
227                 if (this.pos[0][0] + 1 <= 47 && this.pos[2][0] - 1 >=
                        0) {
228                     saveOldPos();
229                     this.pos[0][0]++;
230                     this.pos[2][0]--;
231                     changeAlign(-1);
232                 }
233                 break;
234             case 7:
235                 if (this.pos[0][1] + 1 <= 23 && this.pos[2][1] - 1 >=
                        0) {
236                     saveOldPos();
237                     this.pos[0][1]++;
238                     this.pos[2][1]--;
239                     changeAlign(-1);
240                 }
241                 break;
242             case 8:
243                 if (this.pos[0][1] + 1 <= 23 && this.pos[2][1] - 1 >=
                        0) {
244                     saveOldPos();
245                     this.pos[0][1]++;
246                     this.pos[2][1]--;
247                     changeAlign(-1);
248                 }
249                 break;
250         }
251     }else { // Right
252         switch(this.align){
253             case 1:
254                 if (this.pos[0][0] + 1 <= 47 && this.pos[2][0] - 1 >=
                        0) {
255                     saveOldPos();
256                     this.pos[0][0]++;
257                     this.pos[2][0]--;
258                     changeAlign(1);
259                 }
260                 break;
261             case 2:
262                 if (this.pos[0][1] + 1 <= 23 && this.pos[2][1] - 1 >=
                        0) {
263                     saveOldPos();
264                     this.pos[0][1]++;
265                     this.pos[2][1]--;
266                     changeAlign(1);
267                 }
268                 break;
269             case 3:
270                 if (this.pos[0][1] + 1 <= 23 && this.pos[2][1] - 1 >=
                        0) {
```

```
271                     saveOldPos();
272                     this.pos[0][1]++;
273                     this.pos[2][1]--;
274                     changeAlign(1);
275                 }
276                 break;
277             case 4:
278                 if (this.pos[0][0] - 1 >= 0 && this.pos[2][0] + 1 <=
                        47) {
279                     saveOldPos();
280                     this.pos[0][0]--;
281                     this.pos[2][0]++;
282                     changeAlign(1);
283                 }
284                 break;
285             case 5:
286                 if (this.pos[0][0] - 1 >= 0 && this.pos[2][0] + 1 <=
                        47) {
287                     saveOldPos();
288                     this.pos[0][0]--;
289                     this.pos[2][0]++;
290                     changeAlign(1);
291                 }
292                 break;
293             case 6:
294                 if (this.pos[0][1] - 1 >= 0 && this.pos[2][1] + 1 <=
                        23) {
295                     saveOldPos();
296                     this.pos[0][1]--;
297                     this.pos[2][1]++;
298                     changeAlign(1);
299                 }
300                 break;
301             case 7:
302                 if (this.pos[0][1] - 1 >= 0 && this.pos[2][1] + 1 <=
                        23) {
303                     saveOldPos();
304                     this.pos[0][1]--;
305                     this.pos[2][1]++;
306                     changeAlign(1);
307                 }
308                 break;
309             case 8:
310                 if (this.pos[0][0] + 1 <= 47 && this.pos[2][0] - 1 >=
                        0) {
311                     saveOldPos();
312                     this.pos[0][0]++;
313                     this.pos[2][0]--;
314                     changeAlign(1);
315                 }
```

```
316                 break;
317             }
318         }
319     }
320 
321     /**
322      * Method to save the ship position from one move ago.
323      * */
324     protected void saveOldPos(){
325         for(int i = 0; i < this.pos.length; i++){
326             this.oldpos[i][0] = this.pos[i][0];
327             this.oldpos[i][1] = this.pos[i][1];
328         }
329     }
330 
331     protected void damage(int amount){
332         this.hp -= amount;
333     }
334 
335     /**
336      * Used to move the ship in the direction it is aligned to.
337      * */
338     protected void forward(short[] myImage){
339         if(canMove(myImage)){
340             switch(this.align){
341                 case 1:
342                     saveOldPos();
343                     for (int i = 0; i < this.pos.length; i++){
344                         this.pos[i][1]--;
345                     }
346                     break;
347                 case 2:
348                     saveOldPos();
349                     for (int i = 0; i < this.pos.length; i++){
350                         this.pos[i][0]++;
351                         this.pos[i][1]--;
352                     }
353                     break;
354                 case 3:
355                     saveOldPos();
356                     for (int i = 0; i < this.pos.length; i++){
357                         this.pos[i][0]++;
358                     }
359                     break;
360                 case 4:
361                     saveOldPos();
362                     for (int i = 0; i < this.pos.length; i++){
363                         this.pos[i][0]++;
364                         this.pos[i][1]++;
365                     }
```

```
366                 break;
367             case 5:
368                 saveOldPos();
369                 for (int i = 0; i < this.pos.length; i++){
370                     this.pos[i][1]++;
371                 }
372                 break;
373             case 6:
374                 saveOldPos();
375                 for (int i = 0; i < this.pos.length; i++){
376                     this.pos[i][0]--;
377                     this.pos[i][1]++;
378                 }
379                 break;
380             case 7:
381                 saveOldPos();
382                 for (int i = 0; i < this.pos.length; i++){
383                     this.pos[i][0]--;
384                 }
385                 break;
386             case 8:
387                 saveOldPos();
388                 for (int i = 0; i < this.pos.length; i++){
389                     this.pos[i][0]--;
390                     this.pos[i][1]--;
391                 }
392                 break;
393         }
394     }
395 }
396
397 /**
398  * Used to determine if the ship can move forward.
399  * @return the returnvalue says, if the ship can move forward or if
        the ship would move outside the map.
400  * */
401 protected boolean canMove(short[] myImage){
402     boolean ret = false;
403     switch(this.align){
404         case 1 -> ret = (this.pos[0][1] - 1 >= 0 &&
              !hitIsland(myImage,this.pos[0][0],this.pos[0][1]-1) &&
              !hitEnemy(myImage,this.pos[0][0],this.pos[0][1]-1));
405         case 2 -> ret = (this.pos[0][1] - 1 > 0 && this.pos[0][0] + 1
              < 48 &&
              !hitIsland(myImage,this.pos[0][0]+1,this.pos[0][1]-1) &&
              !hitEnemy(myImage,this.pos[0][0]+1,this.pos[0][1]-1));
406         case 3 -> ret = (this.pos[0][0] + 1 < 48 &&
              !hitIsland(myImage,this.pos[0][0]+1,this.pos[0][1]) &&
              !hitEnemy(myImage,this.pos[0][0]+1,this.pos[0][1]));
407         case 4 -> ret = (this.pos[0][0] + 1 < 48 && this.pos[0][1] +
```

```
                1 < 24 &&
                !hitIsland(myImage,this.pos[0][0]+1,this.pos[0][1]+1) &&
                !hitEnemy(myImage,this.pos[0][0]+1,this.pos[0][1]+1));
408         case 5 -> ret = (this.pos[0][1] + 1 < 24 &&
                !hitIsland(myImage,this.pos[0][0],this.pos[0][1]+1) &&
                !hitEnemy(myImage,this.pos[0][0],this.pos[0][1]+1));
409         case 6 -> ret = (this.pos[0][1] + 1 < 24 && this.pos[0][0] -
                1 >= 0 &&
                !hitIsland(myImage,this.pos[0][0]-1,this.pos[0][1]+1) &&
                !hitEnemy(myImage,this.pos[0][0]-1,this.pos[0][1]+1));
410         case 7 -> ret = (this.pos[0][0] - 1 >= 0 &&
                !hitIsland(myImage,this.pos[0][0]-1,this.pos[0][1]) &&
                !hitEnemy(myImage,this.pos[0][0]-1,this.pos[0][1]));
411         case 8 -> ret = (this.pos[0][0] - 1 >= 0 && this.pos[0][1] -
                1 >= 0 &&
                !hitIsland(myImage,this.pos[0][0]-1,this.pos[0][1]-1) &&
                !hitEnemy(myImage,this.pos[0][0]-1,this.pos[0][1]-1));
412         }
413         return ret;
414     }
415     /**
416      * 8 1 2
417      * 7   3
418      * 6 5 4
419      * */
420     protected void shoot() {
421         int dir1 = (this.align + 2 > 8) ? (this.align + 2 - 8) :
                (this.align + 2);
422         int dir2 = (this.align - 2 < 1) ? (8 + this.align - 2) :
                (this.align - 2);
423         if(this.bullet == null){
424             this.bullet = new Bullet(dir1, 5, this.pos[1][0],
                    this.pos[1][1]);
425         }
426         if(this.bullet2 == null){
427             this.bullet2 = new Bullet(dir2, 5, this.pos[1][0],
                    this.pos[1][1]);
428         }
429     }
430
431     protected boolean hitPlayer(short[] myImage, int x, int y){
432         if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
433             int idx = (y * 48 + x) * 3;
434             return (myImage[idx + 0] == 237 && myImage[idx + 1] == 76 &&
                    myImage[idx + 2] == 36) ||
435                 (myImage[idx + 0] == 237 && myImage[idx + 1] == 207 &&
                        myImage[idx + 2] == 36) ||
436                 (myImage[idx + 0] == 123 && myImage[idx + 1] == 237 &&
                        myImage[idx + 2] == 36) ||
437                 (myImage[idx + 0] == 145 && myImage[idx + 1] == 47 &&
```

```
                    myImage[idx + 2] == 22) ||
438             (myImage[idx + 0] == 148 && myImage[idx + 1] == 129 &&
                    myImage[idx + 2] == 22) ||
439             (myImage[idx + 0] == 74 && myImage[idx + 1] == 143 &&
                    myImage[idx + 2] == 21) ||
440             (myImage[idx + 0] == 74 && myImage[idx + 1] == 24 &&
                    myImage[idx + 2] == 11) ||
441             (myImage[idx + 0] == 66 && myImage[idx + 1] == 58 &&
                    myImage[idx + 2] == 10) ||
442             (myImage[idx + 0] == 38 && myImage[idx + 1] == 74 &&
                    myImage[idx + 2] == 11);
443         }else {
444             return false;
445         }
446     }
447
448     protected boolean hitEnemy(short[] myImage, int x, int y){
449         if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
450             int idx = (y * 48 + x) * 3;
451             return (myImage[idx + 0] == 31 && myImage[idx + 1] == 69 &&
                        myImage[idx + 2] == 222) ||
452                 (myImage[idx + 0] == 19 && myImage[idx + 1] == 43 &&
                        myImage[idx + 2] == 143) ||
453                 (myImage[idx + 0] == 10 && myImage[idx + 1] == 22 &&
                        myImage[idx + 2] == 74) ||
454                 (myImage[idx + 0] == 31 && myImage[idx + 1] == 222 &&
                        myImage[idx + 2] == 215) ||
455                 (myImage[idx + 0] == 21 && myImage[idx + 1] == 138 &&
                        myImage[idx + 2] == 134) ||
456                 (myImage[idx + 0] == 11 && myImage[idx + 1] == 74 &&
                        myImage[idx + 2] == 72) ||
457                 (myImage[idx + 0] == 153 && myImage[idx + 1] == 23 &&
                        myImage[idx + 2] == 209) ||
458                 (myImage[idx + 0] == 94 && myImage[idx + 1] == 15 &&
                        myImage[idx + 2] == 128) ||
459                 (myImage[idx + 0] == 55 && myImage[idx + 1] == 10 &&
                        myImage[idx + 2] == 74);
460         }else {
461             return false;
462         }
463     }
464
465     protected boolean hitBullet(short[] myImage, int x, int y){
466         if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
467             int idx = (y * 48 + x) * 3;
468             return (myImage[idx + 0] == 12 && myImage[idx + 1] == 13 &&
                        myImage[idx + 2] == 12);
469         }else {
470             return false;
471         }
```

```
472     }
473
474     protected boolean hitIsland(short[] myImage, int x, int y){
475         if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
476             int idx = (y * 48 + x) * 3;
477             return (myImage[idx + 0] == 196 && myImage[idx + 1] == 156 &&
                        myImage[idx + 2] == 53) ||
478                    (myImage[idx + 0] == 186 && myImage[idx + 1] == 148 &&
                           myImage[idx + 2] == 48) ||
479                    (myImage[idx + 0] == 125 && myImage[idx + 1] == 66 &&
                           myImage[idx + 2] == 24);
480         }else {
481             return false;
482         }
483     }
484
485
486     /**
487      * This method is used to set the align variable after a succesful
             rotation
488      * @param dir the direction the ship rotates to
489      * */
490     protected void changeAlign(int dir){
491         this.align += dir;
492         if(this.align < 1){
493             this.align = 8;
494         }
495         if(this.align > 8){
496             this.align = 1;
497         }
498     }
499
500     public short[] run(int key, short[] myImage){
501         myImage = isHit(myImage);
502         if(key != -1){
503             myImage = clearTrace(myImage);
504             move(key,myImage);
505             if (collide(myImage) == 1){
506                 resetMove();
507                 if(key == 2){
508                     this.align++;
509                 }
510                 if(key == 3){
511                     this.align--;
512                 }
513             }
514         }
515         if(this.bullet != null){
516             if(this.bullet.getRange() > 0){
517                 myImage = this.bullet.run(-1, myImage);
```

```
518             }else{
519                 this.bullet = null;
520             }
521         }
522         if(this.bullet2 != null){
523             if(this.bullet2.getRange() > 0){
524                 myImage = this.bullet2.run(-1, myImage);
525             }else{
526                 this.bullet2 = null;
527             }
528         }
529         myImage = paint(myImage);
530         return myImage;
531     }
532 
533     protected void resetMove(){
534         for(int i=0; i < this.pos.length; i++){
535             for(int j=0; j < this.pos[i].length; j++){
536                 this.pos[i][j] = this.oldpos[i][j];
537             }
538         }
539     }
540 
541     /**
542      * Debug method to print shiplocation and locationdifference between
             the new and old location.
543      * */
544     public void print(String where){
545         System.out.println(where + "\nA: " + this.align);
546         for (int i = 0; i < this.pos.length; i++){
547             System.out.println("X: " + this.pos[i][0] + " Y: " +
                    this.pos[i][1] + " | Xo: " + this.oldpos[i][0] + " Yo: "
                    + this.oldpos[i][1]);
548             // System.out.println("(" + i + ") -> X: " + (this.pos[i][0]
                    - this.oldpos[i][0]) + " Y: " + (this.pos[i][1] -
                    this.oldpos[i][1]));
549         }
550     }
551 
552     public int[][] getPos(){
553         return this.pos;
554     }
555 
556     public int getHp(){
557         return this.hp;
558     }
559 
560     public void setHp(int hp){
561         this.hp = (hp >= 0)? hp : 0;
562     }
```

```
563
564     protected void changeColor(short[][][] rgbs){
565         for (int i = 0; i < this.color.length; i++) {
566             for (int j = 0; j < this.color[0].length; j++){
567                 for (int k = 0; k < this.color[0][0].length; k++){
568                     this.color[i][j][k] = (rgbs[i][j][k] <= 255 &&
                            rgbs[i][j][k] >= 0)? rgbs[i][j][k] : 0;
569                 }
570             }
571         }
572     }
573 }
```

Agent.java

```
1  public abstract class Agent {
2
3      abstract short[] paint(short[] myImage);
4
5      abstract int collide(short[] myImage);
6
7      abstract void move(int dir,short[] myImage);
8
9      abstract short[] run(int key, short[] myImage);
10 }
```

Enemy.java

```
1  import java.util.List;
2  import java.util.ArrayList;
3  public class Enemy extends Ship {
4      private int range;
5      private int dmg = 0;
6      private int PX;
7      private int PY;
8      private int RouteX = -1;
9      private int RouteY = -1;
10     private boolean detectedPlayer = false;
11     private List<int[]> routing = new ArrayList<int[]>();
12
13     Enemy(int hp){
14         super(hp);
15         short[][][] rgbs = {{{31, 222, 215},{21, 138, 134},{11, 74,
               72}},{{31, 69, 222},{19, 43, 143},{10, 22, 74}},{{153, 23,
               209},{94, 15, 128},{55, 10, 74}}};
16         changeColor(rgbs);
17         int[][] pos = { {22, 22}, {23, 22}, {24, 22} };
18         this.pos = pos;
19     }
20
21     Enemy(int hp, int x, int y, int o, int r){
22         super(hp, x, y, o);
23         short[][][] rgbs = {{{31, 222, 215},{21, 138, 134},{11, 74,
               72}},{{31, 69, 222},{19, 43, 143},{10, 22, 74}},{{153, 23,
               209},{94, 15, 128},{55, 10, 74}}};
24         changeColor(rgbs);
25         this.range = r;
26     }
27
28     public void resetDmg(){
29         this.dmg = 0;
30     }
31
32     public int getDamageReceived(){
33         return this.dmg;
34     }
35
36     /**
37      *  Create Routes
38      * */
39     private void pathFinder(){
40         this.routing = new ArrayList<int[]>();
41         /**
42          *  Start: this.pos[1][0]=x
43          *         this.pos[1][1]=y
44          *  End:   this.RouteX
```

```
45          *         this.RouteY
46          * */
47         this.PX = this.RouteX;
48         this.PY = this.RouteY;
49         int pX = this.pos[1][0];
50         int pY = this.pos[1][1];
51         while(pX != this.RouteX && pY != this.RouteY){
52             int[] ia = {this.RouteX, this.RouteY};
53             switch(routeDirection(pX, pY, ia)){
54                 case 1 -> {
55                     int[] rt = {pX, --pY};
56                     this.routing.add(rt);
57                 }
58                 case 2 -> {
59                     int[] rt = {++pX, --pY};
60                     this.routing.add(rt);
61                 }
62                 case 3 -> {
63                     int[] rt = {++pX, pY};
64                     this.routing.add(rt);
65                 }
66                 case 4 -> {
67                     int[] rt = {++pX, ++pY};
68                     this.routing.add(rt);
69                 }
70                 case 5 -> {
71                     int[] rt = {pX, ++pY};
72                     this.routing.add(rt);
73                 }
74                 case 6 -> {
75                     int[] rt = {--pX, ++pY};
76                     this.routing.add(rt);
77                 }
78                 case 7 -> {
79                     int[] rt = {--pX, pY};
80                     this.routing.add(rt);
81                 }
82                 case 8 -> {
83                     int[] rt = {--pX, --pY};
84                     this.routing.add(rt);
85                 }
86                 default -> {
87                     break;
88                 }
89             }
90         }
91     }
92
93     public short[] run(short[] myImage){
94         if(this.RouteX != -1 && this.RouteY != -1){
```

```
95              pathFinder();
96              pR();
97              System.out.println(this.routing.get(0)[0]);
98          }
99          myImage = clearTrace(myImage);
100         if(playerInVision(myImage) && this.bullet == null){
101             shoot();
102         }
103         move(myImage);
104         if (collide(myImage) != 0){
105             resetMove();
106         }
107         if(this.bullet != null){
108             if(this.bullet.getRange() > 0){
109                 myImage = this.bullet.run(-1, myImage);
110             }else{
111                 this.bullet = null;
112             }
113         }
114         if(this.bullet2 != null){
115             if(this.bullet2.getRange() > 0){
116                 myImage = this.bullet2.run(-1, myImage);
117             }else{
118                 this.bullet2 = null;
119             }
120         }
121         myImage = paint(myImage);
122         return myImage;
123     }
124
125     private void move(short[] myImage){
126         if(this.routing.size() <= 0 ){
127             if(canMove(myImage)){
128                 forward(myImage);
129             }else{
130                 if(Math.random() > 0.5){
131                     rotate(0);
132                 }else {
133                     rotate(1);
134                 }
135             }
136         }else {
137             switch(routeDirection(this.pos[1][0], this.pos[1][1],
                        this.routing.get(this.routing.size() - 1))){
138                 case 1 -> {
139                     rotateTo(1);
140                     forward(myImage);
141                 }
142                 case 2 -> {
143                     rotateTo(2);
```

```
144                 forward(myImage);
145             }
146             case 3 -> {
147                 rotateTo(3);
148                 forward(myImage);
149             }
150             case 4 -> {
151                 rotateTo(4);
152                 forward(myImage);
153             }
154             case 5 -> {
155                 rotateTo(5);
156                 forward(myImage);
157             }
158             case 6 -> {
159                 rotateTo(6);
160                 forward(myImage);
161             }
162             case 7 -> {
163                 rotateTo(7);
164                 forward(myImage);
165             }
166             case 8 -> {
167                 rotateTo(8);
168                 forward(myImage);
169             }
170             default -> {}
171         }
172         this.routing.remove(this.routing.size() - 1);
173     }
174 }
175
176 private void pR(){
177     for(int[] i : this.routing){
178         System.out.println("| " + i[0] + " | " + i[1] + " |");
179     }
180 }
181
182 private void rotateTo(int newOri){
183     while (this.align != newOri) {
184         rotate(1);
185     }
186 }
187
188 private int routeDirection(int x, int y, int[] gPos){
189     if(x > gPos[0]){
190         if(y > gPos[1]){
191             return 8;
192         }else if(y < gPos[1]){
193             return 6;
```

```
194             }else {
195                 return 7;
196             }
197         }else if(x < gPos[0]){
198             if(y > gPos[1]) {
199                 return 2;
200             }else if(y < gPos[1]){
201                 return 4;
202             }else {
203                 return 3;
204             }
205         }else {
206             if(y > gPos[1]) {
207                 return 1;
208             }else if(y < gPos[1]){
209                 return 5;
210             }
211         }
212         return -1;
213     }
214 
215     /**
216      * Method to detect if the player is visible for the enemy ship.
217      * */
218     private boolean playerInVision(short[] myImage){
219         int difx;
220         int dify;
221         if(this.hp > 0){
222             for (int i = 0 - this.range; i <= this.range; i++) {
223                 difx = this.pos[1][0] + i;
224                 for (int j = 0 - this.range; j <= this.range; j++) {
225                     dify = this.pos[1][1] + j;
226                     if ((Math.pow(difx - this.pos[0][1], 2)+Math.pow(dify
                          - this.pos[1][1], 2)) <= Math.pow(this.range, 2))
                          {
227                         if(hitPlayer(myImage, difx,dify)) {
228                             this.detectedPlayer = true;
229                             this.PX = difx;
230                             this.PY = dify;
231                             return true;
232                         }
233                     }
234                 }
235             }
236         }
237         return false;
238     }
239 
240     public void setRouteX(int PX){
241         this.PX = PX;
```

```
242     }
243
244     public void setRouteY(int PY){
245         this.PY = PY;
246     }
247
248     public int getPX(){
249         return this.PX;
250     }
251
252     public int getPY(){
253         return this.PY;
254     }
255
256     public boolean getPlayerDetected(){
257         return this.detectedPlayer;
258     }
259
260     /**
261      * The method collide looks at the pixels of the ship and look if it
            collided with another object
262      * */
263     public int collide(short[] myImage){
264         int ret = 0;
265         for(int i=0; i < this.pos.length; i++){
266             int idx = (this.pos[i][1] * 48 + this.pos[i][0]) * 3;
267             if (hitIsland(myImage, this.pos[i][0], this.pos[i][1])){
268                 ret = 1;
269             }
270             if (hitBullet(myImage, this.pos[i][0], this.pos[i][1])){
271                 damage(1);
272                 ret = 2;
273             }
274             if(hitPlayer(myImage, this.pos[i][0], this.pos[i][1])) {
275                 damage(1);
276                 this.dmg += 1;
277                 ret = 1;
278             }
279         }
280         return ret;
281     }
282
283     public boolean includesPos(int x, int y){
284         for (int i = 0; i < this.pos.length; i++){
285             if(this.pos[i][0] == x && this.pos[i][1] == y){
286                 return true;
287             }
288         }
289         return false;
290     }
```

```
291
292     /**
293      * Debug method to print shiplocation and locationdifference between
             the new and old location.
294      * */
295     public void print(){
296         System.out.println("Enemy ship:\nA: " + this.align);
297         for (int i = 0; i < this.pos.length; i++){
298             System.out.println("X: " + this.pos[i][0] + " Y: " +
                    this.pos[i][1]);
299             System.out.println("Xo: " + this.oldpos[i][0] + " Yo: " +
                    this.oldpos[i][1]);
300             // System.out.println("(" + i + ") -> X: " + (this.pos[i][0]
                    - this.oldpos[i][0]) + " Y: " + (this.pos[i][1] -
                    this.oldpos[i][1]));
301         }
302     }
303 }
```

Fleet.java

```
1  import java.util.List;
2  import java.util.ArrayList;
3  public class Fleet {
4      private List<Enemy> fleet;
5      private int PX;
6      private int PY;
7      private boolean detected = false;
8
9      Fleet(){
10         this.fleet = new ArrayList<Enemy>();
11     }
12
13     public void addFleetmember(Enemy s){
14         this.fleet.add(s);
15     }
16
17     private boolean isWater(short[] myImage, int idx){
18         return (myImage[idx] == 0 && myImage[idx + 1] == 177 &&
               myImage[idx + 2] == 241);
19     }
20
21     public short[] employFleet(short[] myImage, int num){
22         while (num > 0){
23             int i = (int)(Math.random() * 46) + 1;
24             int j = (int)(Math.random() * 22) + 1;
25             if(    isWater(myImage, (((j-1) * 48 + i ) * 3)) &&
26                    isWater(myImage, (((j-1) * 48 + (i+1)) * 3)) &&
27                    isWater(myImage, (((j-1) * 48 + (i-1)) * 3)) &&
28                    isWater(myImage, ((j   * 48 + i   ) * 3)) &&
29                    isWater(myImage, ((j   * 48 + (i+1)) * 3)) &&
30                    isWater(myImage, ((j   * 48 + (i-1)) * 3)) &&
31                    isWater(myImage, (((j+1) * 48 + i ) * 3)) &&
32                    isWater(myImage, (((j+1) * 48 + (i+1)) * 3)) &&
33                    isWater(myImage, (((j+1) * 48 + (i-1)) * 3))){
34                 addFleetmember(new Enemy(2, i, j, 4, 10));
35                 myImage = paintFleet(myImage);
36                 num--;
37                 continue;
38                     }
39         }
40         return myImage;
41     }
42
43     public int getNumberOfAliveShips(){
44         int ret = 0;
45         for (Enemy s : this.fleet){
46             if(s.isAlive()){
47                 ret++;
```

```
48              }
49          }
50          return ret;
51      }
52
53      public void resetDamageControl(){
54          for (Enemy e : this.fleet){
55              e.resetDmg();
56          }
57      }
58      public int damageControl(){
59          int ret = 0;
60          for (Enemy e : this.fleet){
61              ret += e.getDamageReceived();
62          }
63          return ret;
64      }
65
66      public void distributeDamage(int x, int y){
67          for (Enemy e : fleet){
68              if(e.includesPos(x, y)){
69                  e.damage(1);
70                  break;
71              }
72          }
73      }
74
75      public int getDead(){
76          int ret = 0;
77          for (Enemy e : fleet){
78              if(!e.isAlive()){
79                  ret++;
80              }
81          }
82          return ret;
83      }
84
85      public void printing(){
86          for(Enemy e : this.fleet){
87              e.print("text");
88          }
89      }
90
91      private void broadcastPosition(){
92          for(Enemy e : this.fleet){
93              if(e.getPlayerDetected()){
94                  this.detected = true;
95                  this.PX = e.getPX();
96                  this.PY = e.getPY();
97              }
```

```
98          }
99      }
100
101     public short[] executeOrders(short[] myImage){
102         broadcastPosition();
103         for (Enemy s : this.fleet){
104             if(s.isAlive()){
105                 s.setRouteX(this.PX);
106                 s.setRouteY(this.PY);
107                 myImage = s.run(myImage);
108             }
109         }
110         return myImage;
111     }
112
113     public short[] statusUpdate(short[] myImage){
114         for(Enemy e : this.fleet){
115             if(e.isAlive()){
116                 myImage = e.isHit(myImage);
117             }
118         }
119         return myImage;
120     }
121
122     public short[] paintFleet(short[] myImage){
123         for (Enemy s : this.fleet) {
124             if(s.isAlive()){
125                 myImage = s.paint(myImage);
126             }
127         }
128         return myImage;
129     }
130 }
```

Bullet.java

```
1  public class Bullet extends Agent {
2      private int direction;
3      private int range;
4      private int maxRange;
5      private int[] pos = new int[2];
6      private int[] oldpos = new int[2];
7      private boolean hasHit = false;
8
9      Bullet(int dir, int range, int x, int y){
10         this.direction = dir;
11         this.range = range;
12         this.maxRange = range;
13         this.pos[0] = x;
14         this.pos[1] = y;
15     }
16
17     private short[] clearTrace(short[] myImage){
18         myImage[(this.oldpos[1] * 48 + this.oldpos[0]) * 3 + 0] =
               (short)0;
19         myImage[(this.oldpos[1] * 48 + this.oldpos[0]) * 3 + 1] =
               (short)177;
20         myImage[(this.oldpos[1] * 48 + this.oldpos[0]) * 3 + 2] =
               (short)241;
21         return myImage;
22     }
23
24     public short[] clear(short[] myImage){
25             myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 0] = (short)0;
26             myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 1] =
                   (short)177;
27             myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 2] =
                   (short)241;
28         return myImage;
29     }
30     public short[] paint(short[] myImage){
31         myImage = clearTrace(myImage);
32         if(this.range > 0) {
33             myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 0] = (short)12;
34             myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 1] = (short)13;
35             myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 2] = (short)12;
36         }else {
37             myImage = clear(myImage);
38         }
39         return myImage;
40     }
41
42     public int collide(short[] myImage){
43         return 0;
```

```
44      }
45
46      protected boolean hitPlayer(short[] myImage, int x, int y){
47          if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
48          int idx = (y * 48 + x) * 3;
49          return (myImage[idx + 0] == 237 && myImage[idx + 1] == 76 &&
                    myImage[idx + 2] == 36) ||
50              (myImage[idx + 0] == 237 && myImage[idx + 1] == 207 &&
                    myImage[idx + 2] == 36) ||
51              (myImage[idx + 0] == 123 && myImage[idx + 1] == 237 &&
                    myImage[idx + 2] == 36) ||
52              (myImage[idx + 0] == 145 && myImage[idx + 1] == 47 &&
                    myImage[idx + 2] == 22) ||
53              (myImage[idx + 0] == 148 && myImage[idx + 1] == 129 &&
                    myImage[idx + 2] == 22) ||
54              (myImage[idx + 0] == 74 && myImage[idx + 1] == 143 &&
                    myImage[idx + 2] == 21) ||
55              (myImage[idx + 0] == 74 && myImage[idx + 1] == 24 &&
                    myImage[idx + 2] == 11) ||
56              (myImage[idx + 0] == 66 && myImage[idx + 1] == 58 &&
                    myImage[idx + 2] == 10) ||
57              (myImage[idx + 0] == 38 && myImage[idx + 1] == 74 &&
                    myImage[idx + 2] == 11);
58          }else {
59              return false;
60          }
61      }
62
63      protected boolean hitEnemy(short[] myImage, int x, int y){
64          if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
65          int idx = (y * 48 + x) * 3;
66          return (myImage[idx + 0] == 31 && myImage[idx + 1] == 69 &&
                    myImage[idx + 2] == 222) ||
67              (myImage[idx + 0] == 19 && myImage[idx + 1] == 43 &&
                    myImage[idx + 2] == 143) ||
68              (myImage[idx + 0] == 10 && myImage[idx + 1] == 22 &&
                    myImage[idx + 2] == 74) ||
69              (myImage[idx + 0] == 31 && myImage[idx + 1] == 222 &&
                    myImage[idx + 2] == 215) ||
70              (myImage[idx + 0] == 21 && myImage[idx + 1] == 138 &&
                    myImage[idx + 2] == 134) ||
71              (myImage[idx + 0] == 11 && myImage[idx + 1] == 74 &&
                    myImage[idx + 2] == 72) ||
72              (myImage[idx + 0] == 153 && myImage[idx + 1] == 23 &&
                    myImage[idx + 2] == 209) ||
73              (myImage[idx + 0] == 94 && myImage[idx + 1] == 15 &&
                    myImage[idx + 2] == 128) ||
74              (myImage[idx + 0] == 55 && myImage[idx + 1] == 10 &&
                    myImage[idx + 2] == 74);
75          }else {
```

```
76              return false;
77          }
78      }
79
80      protected boolean hitIsland(short[] myImage, int x, int y){
81          if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
82              int idx = (y * 48 + x) * 3;
83              return (myImage[idx + 0] == 196 && myImage[idx + 1] == 156 &&
                    myImage[idx + 2] == 53) ||
84                  (myImage[idx + 0] == 186 && myImage[idx + 1] == 148 &&
                        myImage[idx + 2] == 48) ||
85                  (myImage[idx + 0] == 125 && myImage[idx + 1] == 66 &&
                        myImage[idx + 2] == 24);
86          }else {
87              return false;
88          }
89      }
90
91
92      public void move(int dir,short[] myImage){}
93
94      private void saveOldPos(){
95          this.oldpos[0] = this.pos[0];
96          this.oldpos[1] = this.pos[1];
97      }
98
99      public boolean move(){
100         if(canMove()){
101             saveOldPos();
102             switch(this.direction){
103                 case 1:
104                     this.pos[1]--;
105                     break;
106                 case 2:
107                     this.pos[0]++;
108                     this.pos[1]--;
109                     break;
110                 case 3:
111                     this.pos[0]++;
112                     break;
113                 case 4:
114                     this.pos[0]++;
115                     this.pos[1]++;
116                     break;
117                 case 5:
118                     this.pos[1]++;
119                     break;
120                 case 6:
121                     this.pos[0]--;
122                     this.pos[1]++;
```

```
123                 break;
124             case 7:
125                 this.pos[0]--;
126                 break;
127             case 8:
128                 this.pos[0]--;
129                 this.pos[1]--;
130                 break;
131         }
132         this.range--;
133         return true;
134     }else {
135         return false;
136     }
137 }
138 
139 private boolean canMove(){
140     boolean ret = false;
141     switch(this.direction){
142         case 1 -> ret = (this.pos[1] - 1 >= 0);
143         case 2 -> ret = (this.pos[1] - 1 > 0 && this.pos[0] + 1 < 48);
144         case 3 -> ret = (this.pos[0] + 1 < 48);
145         case 4 -> ret = (this.pos[0] + 1 < 48 && this.pos[1] + 1 <
                24);
146         case 5 -> ret = (this.pos[1] + 1 < 24);
147         case 6 -> ret = (this.pos[1] + 1 < 24 && this.pos[0] - 1 >=
                0);
148         case 7 -> ret = (this.pos[0] - 1 >= 0);
149         case 8 -> ret = (this.pos[0] - 1 >= 0 && this.pos[1] - 1 >=
                0);
150     }
151     return ret;
152 }
153 
154 public short[] run(int key, short[] myImage){
155     if(this.range == this.maxRange){
156         if(move()){
157         myImage = paint(myImage);
158         }else {
159             this.range = 0;
160             myImage = clear(myImage);
161         }
162     }else{
163     if (!(hitEnemy(myImage, this.pos[0], this.pos[1]) ||
             hitPlayer(myImage, this.pos[0], this.pos[1]) ||
             hitIsland(myImage, this.pos[0], this.pos[1]))){
164         if(move()){
165         myImage = paint(myImage);
166         }else {
167             this.range = 0;
```

```
168                 myImage = clear(myImage);
169             }
170         }else{
171             this.range = 0;
172             this.hasHit = true;
173             myImage = paint(myImage);
174         }
175         }
176         return myImage;
177     }
178 
179     public boolean getHasHit(){
180         return this.hasHit;
181     }
182 
183     public int getRange(){
184         return this.range;
185     }
186 }
```

Harbor.java

```java
public class Harbor extends Agent{
    protected short[] color = {125, 66, 24};
    protected int orient;
    protected boolean captured = false;
    protected int possession = -1;
    protected int[] pos;
    protected int[] enemyPos;
    protected Bullet bullet = null;

    Harbor(int orient){
        this.orient = orient;
        this.pos = new int[2];
        this.enemyPos = new int[2];
    }

    public int getOrient(){
        return this.orient;
    }

    public short[] getColor() {
        return this.color;
    }

    public void setPos(int y, int x){
        if(x >= 0 && x <= 47){
            this.pos[0] = x;
        }
        if(y >= 0 && y <= 23){
            this.pos[1] = y;
        }
    }

    @Override
    short[] paint(short[] myImage) {
        return new short[0];
    }

    @Override
    int collide(short[] myImage) {
        return -1;
    }

    @Override
    void move(int dir,short[] myImage) {
    }

    @Override
    short[] run(int key, short[] myImage) {
```

```
49          if(this.captured){
50              if(detectShip(15, myImage) == this.possession) {
51                  shoot();
52              }
53          } else {
54              int poss = detectShip(1, myImage);
55              if(poss != -1) {
56                  this.captured = true;
57                  if(poss == 0) {
58                      this.possession = 1;
59                      this.color[2] += 150;
60                  }else {
61                      this.possession = 0;
62                      this.color[1] += 150;
63                  }
64              }
65          }
66          if(this.bullet != null){
67              if(this.bullet.getRange() > 0){
68                  myImage = this.bullet.run(-1, myImage);
69              }else{
70                  this.bullet = null;
71              }
72          }
73          return myImage;
74      }
75      private int routeDirection(int x, int y, int[] gPos){
76          if(x > gPos[0]){
77              if(y > gPos[1]){
78                  return 8;
79              }else if(y < gPos[1]){
80                  return 6;
81              }else {
82                  return 7;
83              }
84          }else if(x < gPos[0]){
85              if(y > gPos[1]) {
86                  return 2;
87              }else if(y < gPos[1]){
88                  return 4;
89              }else {
90                  return 3;
91              }
92          }else {
93              if(y > gPos[1]) {
94                  return 1;
95              }else if(y < gPos[1]){
96                  return 5;
97              }
98          }
```

```
99          return -1;
100     }
101     protected void shoot() {
102         int orient = routeDirection(this.pos[0], this.pos[1],
                this.enemyPos);
103         if(this.bullet == null){
104             this.bullet = new Bullet(orient, 8, this.pos[0], this.pos[1]);
105         }
106     }
107     protected boolean hitPlayer(short[] myImage, int x, int y){
108         if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
109             int idx = (y * 48 + x) * 3;
110             return (myImage[idx + 0] == 237 && myImage[idx + 1] == 76 &&
                    myImage[idx + 2] == 36) ||
111                 (myImage[idx + 0] == 237 && myImage[idx + 1] == 207 &&
                        myImage[idx + 2] == 36) ||
112                 (myImage[idx + 0] == 123 && myImage[idx + 1] == 237 &&
                        myImage[idx + 2] == 36) ||
113                 (myImage[idx + 0] == 145 && myImage[idx + 1] == 47 &&
                        myImage[idx + 2] == 22) ||
114                 (myImage[idx + 0] == 148 && myImage[idx + 1] == 129 &&
                        myImage[idx + 2] == 22) ||
115                 (myImage[idx + 0] == 74 && myImage[idx + 1] == 143 &&
                        myImage[idx + 2] == 21) ||
116                 (myImage[idx + 0] == 74 && myImage[idx + 1] == 24 &&
                        myImage[idx + 2] == 11) ||
117                 (myImage[idx + 0] == 66 && myImage[idx + 1] == 58 &&
                        myImage[idx + 2] == 10) ||
118                 (myImage[idx + 0] == 38 && myImage[idx + 1] == 74 &&
                        myImage[idx + 2] == 11);
119         }else {
120             return false;
121         }
122     }
123     protected boolean hitEnemy(short[] myImage, int x, int y){
124         if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
125             int idx = (y * 48 + x) * 3;
126             return (myImage[idx + 0] == 31 && myImage[idx + 1] == 69 &&
                    myImage[idx + 2] == 222) ||
127                 (myImage[idx + 0] == 19 && myImage[idx + 1] == 43 &&
                        myImage[idx + 2] == 143) ||
128                 (myImage[idx + 0] == 10 && myImage[idx + 1] == 22 &&
                        myImage[idx + 2] == 74) ||
129                 (myImage[idx + 0] == 31 && myImage[idx + 1] == 222 &&
                        myImage[idx + 2] == 215) ||
130                 (myImage[idx + 0] == 21 && myImage[idx + 1] == 138 &&
                        myImage[idx + 2] == 134) ||
131                 (myImage[idx + 0] == 11 && myImage[idx + 1] == 74 &&
                        myImage[idx + 2] == 72) ||
132                 (myImage[idx + 0] == 153 && myImage[idx + 1] == 23 &&
```

```
                    myImage[idx + 2] == 209) ||
133                 (myImage[idx + 0] == 94 && myImage[idx + 1] == 15 &&
                        myImage[idx + 2] == 128) ||
134                 (myImage[idx + 0] == 55 && myImage[idx + 1] == 10 &&
                        myImage[idx + 2] == 74);
135         }else {
136             return false;
137         }
138     }
139     protected int detectShip(int range, short[] myImage){
140         int difx;
141         int dify;
142         for (int i = -range; i <= range; i++) {
143             difx = this.pos[0] + i;
144             for (int j = -range; j <= range; j++) {
145                 dify = this.pos[1] + j;
146                 if ((Math.pow(difx - this.pos[0], 2)+Math.pow(dify -
                        this.pos[1], 2)) <= Math.pow(range, 2)) {
147                     if(hitPlayer(myImage, difx, dify)) {
148                         this.enemyPos[0] = difx;
149                         this.enemyPos[1] = dify;
150                         return 1;
151                     }
152                     if(hitEnemy(myImage, difx, dify)) {
153                         this.enemyPos[0] = difx;
154                         this.enemyPos[1] = dify;
155                         return 0;
156                     }
157                 }
158             }
159         }
160         return -1;
161     }
162 }
```

Island.java

```
1  import java.util.List;
2
3  public class Island extends Agent {
4      protected short[][] color = {{196, 156, 53},{186, 148, 48}}; //
           normale Insel, Hafeninsel
5      protected int[][][] pos;
6      protected int[] size;
7      protected Harbor harbor = null;
8
9      Island(int[] size, int x, int y){
10         this.pos = new int[size[0]][size[1]][2];
11         for(int i1 = 0; i1 < size[0]; i1++){
12             for(int i2 = 0; i2 < size[1]; i2++){
13                 this.pos[i1][i2][0] = x + i1;
14                 this.pos[i1][i2][1] = y + i2;
15             }
16         }
17     }
18
19     Island(int[] size, int x, int y, Harbor harbor){
20         this.pos = new int[size[0]][size[1]][2];
21         for(int i1 = 0; i1 < size[0]; i1++){
22             for(int i2 = 0; i2 < size[1]; i2++){
23                 this.pos[i1][i2][0] = x + i1;
24                 this.pos[i1][i2][1] = y + i2;
25             }
26         }
27         this.harbor = harbor;
28     }
29
30     @Override
31     short[] paint(short[] myImage) {
32         for(int i1 = 0; i1 < this.pos.length; i1++){
33             for(int i2 = 0; i2 < this.pos[i1].length; i2++){
34                 if(harbor == null){
35                     myImage[(this.pos[i1][i2][1] * 48 +
                           this.pos[i1][i2][0]) * 3 + 0] = color[0][0]; //
                           (y * 48 + x) * 3 + 0
36                     myImage[(this.pos[i1][i2][1] * 48 +
                           this.pos[i1][i2][0]) * 3 + 1] = color[0][1]; //
                           (y * 48 + x) * 3 + 1
37                     myImage[(this.pos[i1][i2][1] * 48 +
                           this.pos[i1][i2][0]) * 3 + 2] = color[0][2]; //
                           (y * 48 + x) * 3 + 2
38                 }
39                 else{
40                     myImage[(this.pos[i1][i2][1] * 48 +
                           this.pos[i1][i2][0]) * 3 + 0] = color[1][0]; //
```

```
                    (y * 48 + x) * 3 + 0
41                  myImage[(this.pos[i1][i2][1] * 48 +
                    this.pos[i1][i2][0]) * 3 + 1] = color[1][1]; //
                    (y * 48 + x) * 3 + 1
42                  myImage[(this.pos[i1][i2][1] * 48 +
                    this.pos[i1][i2][0]) * 3 + 2] = color[1][2]; //
                    (y * 48 + x) * 3 + 2
43                  switch(harbor.getOrient()){
44                      case 1:
45                          myImage[(this.pos[0][(int)(this.pos[0].length/2)][1]
                                * 48 +
                                this.pos[0][(int)(this.pos[0].length/2)][0])
                                * 3 + 0] = harbor.getColor()[0]; // (y *
                                48 + x) * 3 + 0
46                          myImage[(this.pos[0][(int)(this.pos[0].length/2)][1]
                                * 48 +
                                this.pos[0][(int)(this.pos[0].length/2)][0])
                                * 3 + 1] = harbor.getColor()[1]; // (y *
                                48 + x) * 3 + 1
47                          myImage[(this.pos[0][(int)(this.pos[0].length/2)][1]
                                * 48 +
                                this.pos[0][(int)(this.pos[0].length/2)][0])
                                * 3 + 2] = harbor.getColor()[2]; // (y *
                                48 + x) * 3 + 2
48                          harbor.setPos(this.pos[0][(int)(this.pos[0].length/2)][1],this.pos[0][(int)(t
49                          break;
50                      case 2:
51                          myImage[(this.pos[0][(int)(this.pos[0].length-1)][1]
                                * 48 +
                                this.pos[0][(int)(this.pos[0].length-1)][0])
                                * 3 + 0] = harbor.getColor()[0]; // (y *
                                48 + x) * 3 + 0
52                          myImage[(this.pos[0][(int)(this.pos[0].length-1)][1]
                                * 48 +
                                this.pos[0][(int)(this.pos[0].length-1)][0])
                                * 3 + 1] = harbor.getColor()[1]; // (y *
                                48 + x) * 3 + 1
53                          myImage[(this.pos[0][(int)(this.pos[0].length-1)][1]
                                * 48 +
                                this.pos[0][(int)(this.pos[0].length-1)][0])
                                * 3 + 2] = harbor.getColor()[2]; // (y *
                                48 + x) * 3 + 2
54                          harbor.setPos(this.pos[0][(int)(this.pos[0].length-1)][1],this.pos[0][(int)(t
55                          break;
56                      case 3:
57                          myImage[(this.pos[(int)(this.pos.length/2)][(int)(this.pos[0].length-1)][1]
                                * 48 +
                                this.pos[(int)(this.pos.length/2)][(int)(this.pos[0].length-1)][0])
                                * 3 + 0] = harbor.getColor()[0]; // (y *
                                48 + x) * 3 + 0
```

```
58                          myImage[(this.pos[(int)(this.pos.length/2)][(int)(this.pos[0].length-1)][1]
                                * 48 +
                                this.pos[(int)(this.pos.length/2)][(int)(this.pos[0].length-1)][0])
                                * 3 + 1] = harbor.getColor()[1]; // (y *
                                48 + x) * 3 + 1
59                          myImage[(this.pos[(int)(this.pos.length/2)][(int)(this.pos[0].length-1)][1]
                                * 48 +
                                this.pos[(int)(this.pos.length/2)][(int)(this.pos[0].length-1)][0])
                                * 3 + 2] = harbor.getColor()[2]; // (y *
                                48 + x) * 3 + 2
60                          harbor.setPos(this.pos[(int)(this.pos.length/2)][(int)(this.pos[0].length-1)]
61                          break;
62                      case 4:
63                          myImage[(this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length-1)][1]
                                * 48 +
                                this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length-1)][0])
                                * 3 + 0] = harbor.getColor()[0]; // (y *
                                48 + x) * 3 + 0
64                          myImage[(this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length-1)][1]
                                * 48 +
                                this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length-1)][0])
                                * 3 + 1] = harbor.getColor()[1]; // (y *
                                48 + x) * 3 + 1
65                          myImage[(this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length-1)][1]
                                * 48 +
                                this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length-1)][0])
                                * 3 + 2] = harbor.getColor()[2]; // (y *
                                48 + x) * 3 + 2
66                          harbor.setPos(this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length-1)]
67                          break;
68                      case 5:
69                          myImage[(this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length/2)][1]
                                * 48 +
                                this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length/2)][0])
                                * 3 + 0] = harbor.getColor()[0]; // (y *
                                48 + x) * 3 + 0
70                          myImage[(this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length/2)][1]
                                * 48 +
                                this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length/2)][0])
                                * 3 + 1] = harbor.getColor()[1]; // (y *
                                48 + x) * 3 + 1
71                          myImage[(this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length/2)][1]
                                * 48 +
                                this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length/2)][0])
                                * 3 + 2] = harbor.getColor()[2]; // (y *
                                48 + x) * 3 + 2
72                          harbor.setPos(this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length/2)]
73                          break;
74                      case 6:
75                          myImage[(this.pos[(int)(this.pos.length-1)][0][1]
```

```
                        * 48 +
                        this.pos[(int)(this.pos.length-1)][0][0])
                        * 3 + 0] = harbor.getColor()[0]; // (y *
                        48 + x) * 3 + 0
76                  myImage[(this.pos[(int)(this.pos.length-1)][0][1]
                        * 48 +
                        this.pos[(int)(this.pos.length-1)][0][0])
                        * 3 + 1] = harbor.getColor()[1]; // (y *
                        48 + x) * 3 + 1
77                  myImage[(this.pos[(int)(this.pos.length-1)][0][1]
                        * 48 +
                        this.pos[(int)(this.pos.length-1)][0][0])
                        * 3 + 2] = harbor.getColor()[2]; // (y *
                        48 + x) * 3 + 2
78                  harbor.setPos(this.pos[(int)(this.pos.length-1)][0][1],this.pos[(int)(this.po
79                  break;
80              case 7:
81                  myImage[(this.pos[(int)(this.pos.length/2)][0][1]
                        * 48 +
                        this.pos[(int)(this.pos.length/2)][0][0])
                        * 3 + 0] = harbor.getColor()[0]; // (y *
                        48 + x) * 3 + 0
82                  myImage[(this.pos[(int)(this.pos.length/2)][0][1]
                        * 48 +
                        this.pos[(int)(this.pos.length/2)][0][0])
                        * 3 + 1] = harbor.getColor()[1]; // (y *
                        48 + x) * 3 + 1
83                  myImage[(this.pos[(int)(this.pos.length/2)][0][1]
                        * 48 +
                        this.pos[(int)(this.pos.length/2)][0][0])
                        * 3 + 2] = harbor.getColor()[2]; // (y *
                        48 + x) * 3 + 2
84                  harbor.setPos(this.pos[(int)(this.pos.length/2)][0][1],this.pos[(int)(this.po
85                  break;
86              case 8:
87                  myImage[(this.pos[0][0][1] * 48 +
                        this.pos[0][0][0]) * 3 + 0] =
                        harbor.getColor()[0]; // (y * 48 + x) * 3
                        + 0
88                  myImage[(this.pos[0][0][1] * 48 +
                        this.pos[0][0][0]) * 3 + 1] =
                        harbor.getColor()[1]; // (y * 48 + x) * 3
                        + 1
89                  myImage[(this.pos[0][0][1] * 48 +
                        this.pos[0][0][0]) * 3 + 2] =
                        harbor.getColor()[2]; // (y * 48 + x) * 3
                        + 2
90                  harbor.setPos(this.pos[0][0][1],this.pos[0][0][0]);
91                  break;
92          }
```

```
93                 }
94             }
95         }
96         return myImage;
97     }
98 
99     public boolean hasHarbor(){
100        return (this.harbor != null) ? true : false;
101    }
102 
103    public Harbor getHarbor(){
104        return this.harbor;
105    }
106 
107    @Override
108    int collide(short[] myImage) {
109        return -1;
110    }
111 
112    @Override
113    void move(int dir,short[] myImage) {
114    }
115 
116    @Override
117    short[] run(int key, short[] myImage) {
118        return myImage;
119    }
120 
121 }
```

Player.java

```
1  public class Player extends Ship {
2      private Score score;
3      private boolean hit = false;
4      private int hitX;
5      private int hitY;
6
7      Player(int hp){
8          super(hp);
9          this.score = new Score();
10         short[][][] rgbs = {{{237, 76, 36},{145, 47, 22},{74, 24,
               11}},{{237, 207, 36},{148, 129, 22},{66, 58, 10}},{{123,
               237, 36},{74, 143, 21},{38, 74, 11}}};
11         changeColor(rgbs);
12
13         int[][] pos = { {5, 5}, {6, 5}, {7, 5} };
14         this.pos = pos;
15     }
16
17     Player(int hp, int x, int y, int o){
18         super(hp, x, y, o);
19         this.score = new Score();
20         short[][][] rgbs = {{{237, 76, 36},{145, 47, 22},{74, 24,
               11}},{{237, 207, 36},{148, 129, 22},{66, 58, 10}},{{123,
               237, 36},{74, 143, 21},{38, 74, 11}}};
21         changeColor(rgbs);
22     }
23
24     /**
25      * The method collide looks at the pixels of the ship and look if it
            collided with another object
26      * */
27     public int collide(short[] myImage){
28         int ret = 0;
29         for(int i=0; i < this.pos.length; i++){
30             if (hitIsland(myImage, this.pos[i][0], this.pos[i][1])){
31                 ret = 1;
32             }
33             if (hitBullet(myImage, this.pos[i][0], this.pos[i][1])){
34                 damage(1);
35                 ret = 1;
36             }
37             if(hitEnemy(myImage, this.pos[i][0], this.pos[i][1])) {
38                 this.hit = true;
39                 this.hitX = this.pos[i][0];
40                 this.hitY = this.pos[i][1];
41                 damage(1);
42                 ret = 1;
43             }
```

```
44          }
45          return ret;
46      }
47
48      public int[][] getPos(){
49          return this.pos;
50      }
51
52      public int getHitX(){
53          return this.hitX;
54      }
55      public int getHitY(){
56          return this.hitY;
57      }
58      public boolean getHit(){
59          return this.hit;
60      }
61
62      public void resetHit(){
63          this.hit = false;
64          this.hitX = -1;
65          this.hitY = -1;
66      }
67
68      public short[] run(int key, short[] myImage){
69          myImage = isHit(myImage);
70          if(key != -1){
71              myImage = clearTrace(myImage);
72              move(key,myImage);
73              if (collide(myImage) == 1){
74                  resetMove();
75                  if(key == 2){
76                      this.align++;
77                  }
78                  if(key == 3){
79                      this.align--;
80                  }
81              }
82          }
83          if(this.bullet != null){
84              if(this.bullet.getHasHit()){
85                  addScore(50);
86              }
87              if(this.bullet.getRange() > 0){
88                  myImage = this.bullet.run(-1, myImage);
89              }else{
90                  this.bullet = null;
91              }
92          }
93          if(this.bullet2 != null){
```

```
94              if(this.bullet2.getHasHit()){
95                  addScore(50);
96              }
97              if(this.bullet2.getRange() > 0){
98                  myImage = this.bullet2.run(-1, myImage);
99              }else{
100                 this.bullet2 = null;
101             }
102         }
103         myImage = paint(myImage);
104         return myImage;
105     }
106
107     public Score getScore(){
108         return this.score;
109     }
110
111     public void addScore(int val){
112         this.score.add(val);
113     }
114
115     /**
116      * Debug method to print shiplocation and locationdifference between
            the new and old location.
117      * */
118     public void print(){
119         System.out.println("Your ship:\nA: " + this.align);
120         for (int i = 0; i < this.pos.length; i++){
121             System.out.println("X: " + this.pos[i][0] + " Y: " +
                    this.pos[i][1]);
122             System.out.println("Xo: " + this.oldpos[i][0] + " Yo: " +
                    this.oldpos[i][1]);
123             // System.out.println("(" + i + ") -> X: " + (this.pos[i][0]
                    - this.oldpos[i][0]) + " Y: " + (this.pos[i][1] -
                    this.oldpos[i][1]));
124         }
125     }
126     private static boolean isWater(short[] myImage, int idx){
127         return (myImage[idx] == 0 && myImage[idx + 1] == 177 &&
                myImage[idx + 2] == 241);
128     }
129     public static Player spawn(short[] myImage, int num){
130         Player ret = null;
131         while (num > 0){
132             int i = (int)(Math.random() * 46) + 1;
133             int j = (int)(Math.random() * 22) + 1;
134             if(    isWater(myImage, (((j-1) * 48 + i ) * 3)) &&
135                    isWater(myImage, (((j-1) * 48 + (i+1)) * 3)) &&
136                    isWater(myImage, (((j-1) * 48 + (i-1)) * 3)) &&
137                    isWater(myImage, ((j   * 48 + i   ) * 3)) &&
```

```
138                     isWater(myImage, ((j   * 48 + (i+1)) * 3)) &&
139                     isWater(myImage, ((j   * 48 + (i-1)) * 3)) &&
140                     isWater(myImage, (((j+1) * 48 + i ) * 3)) &&
141                     isWater(myImage, (((j+1) * 48 + (i+1)) * 3)) &&
142                     isWater(myImage, (((j+1) * 48 + (i-1)) * 3))){
143                 ret = new Player(3,i,j,(int)(Math.random()*7)+1);
144                 num--;
145                 continue;
146             }
147         }
148         return ret;
149     }
150 }
```

GameMain.java

```
1  import java.util.ArrayList;
2  import java.util.Collections;
3  import java.util.List;
4  import java.util.Scanner;
5  public class GameMain {
6
7      static public void main(String[] passedArgs) throws
           InterruptedException {
8          short[] myImage = new short[24*48*3];
9          List<Integer> highscore = new ArrayList<Integer>();
10         int thisKey=0;
11         int frame = 0;
12         int round = 1;
13         long startTime = System.currentTimeMillis();
14         long roundtime = 30000;
15
16         // This is initialization, donot change this
17         InternalLedGameThread.run();
18
19         // Now we show some introductory message and wait 3s before we
               switch to purple
20         System.out.println("Willkommen bei Mari proelium!\n In kuerze
               wird das Spiel beginnen und Ihr Punktestand wird mit den
               anderen Spielern verglichen!\n");
21         Thread.sleep(1000);
22
23         boolean end = false;
24         Scanner scan = new Scanner(System.in);
25         do {
26             System.out.println("Sending to displayThread");
27             Fleet fleet = new Fleet();
28             World world = new World();
29             myImage = world.parseImage("intro.mvh");
30             InternalLedGameThread.showImage(myImage);
31             Thread.sleep(1500);
32             myImage = world.fade(myImage);
33             InternalLedGameThread.showImage(myImage);
34             Thread.sleep(500);
35             myImage = world.fade(myImage);
36             InternalLedGameThread.showImage(myImage);
37             Thread.sleep(500);
38             myImage = world.fade(myImage);
39             InternalLedGameThread.showImage(myImage);
40             Thread.sleep(500);
41             myImage = world.parseImage("round1.mvh");
42             InternalLedGameThread.showImage(myImage);
43             Thread.sleep(500);
44             myImage = world.clear();
```

```
45              myImage = world.createIsland(myImage, 5);
46              myImage = fleet.employFleet(myImage, 3);
47              Player p = Player.spawn(myImage,1);
48              myImage = p.paint(myImage);
49              myImage = fleet.paintFleet(myImage);
50              InternalLedGameThread.showImage(myImage);
51              System.out.println("Drucken Sie eine beliebige Taste um das
                    Spiel zu starten.");
52              while(true){
53                  if(InternalLedGameThread.getKeyboard() != -1){
54                      break;
55                  }
56              }
57              while(p.isAlive()){
58                  thisKey = InternalLedGameThread.getKeyboard();
59                  myImage = p.run(thisKey, myImage);
60                  if(p.getHit()){
61                      fleet.distributeDamage(p.getHitX(), p.getHitY());
62                  }
63                  myImage = fleet.statusUpdate(myImage);
64                  if(frame % 4 == 0) {
65                      frame = 0;
66                      myImage = fleet.executeOrders(myImage);
67                  }
68                  myImage = world.runHarbor(myImage);
69                  myImage = world.paintIslands(myImage);
70                  InternalLedGameThread.showImage(myImage);
71                  frame++;
72                  Thread.sleep(100);
73                  System.out.println("+++ " + (System.currentTimeMillis() -
                        startTime) + " +++");
74                  p.damage(fleet.damageControl());
75                  fleet.resetDamageControl();
76                  if((System.currentTimeMillis() - startTime) > roundtime){
77                      myImage = fleet.employFleet(myImage, (3 -
                            fleet.getNumberOfAliveShips()));
78                      round++;
79                      if(round == 16){
80                          round = 1;
81                      }
82                      startTime = System.currentTimeMillis();
83                      p.addScore(200);
84                      String s = "round" + round + ".mvh";
85                      System.out.println(s);
86                      myImage = world.parseImage(s);
87                      InternalLedGameThread.showImage(myImage);
88                      Thread.sleep(1000);
89                      myImage = world.clear();
90                  }
91                  if(fleet.getNumberOfAliveShips() == 0){
```

```
92                  p.addScore(50);
93              }
94          }
95          world.clear();
96          myImage = world.parseImage("gameover.mvh");
97          InternalLedGameThread.showImage(myImage);
98          p.addScore(fleet.getDead() * 50);
99          highscore.add(p.getScore().getScore());
100         Collections.sort(highscore);
101         Collections.reverse(highscore);
102         System.out.println("(" + round + ") - Score: " +
                p.getScore().getScore());
103         System.out.println("Highscores:");
104         for(int i = 0; i < highscore.size(); i++){
105             System.out.println("(" + (i+1) + ") -> " +
                    highscore.get(i));
106         }
107         myImage = world.parseImage("commands.mvh");
108         InternalLedGameThread.showImage(myImage);
109         Thread.sleep(1500);
110         while(true){
111             thisKey = InternalLedGameThread.getKeyboard();
112             if(thisKey != -1){
113                 break;
114             }
115         }
116         switch(thisKey){
117             case 0 -> {
118                 end = false;
119             }
120             case 1 -> {
121                 end = true;
122                 System.exit(0);
123             }
124             case 2 -> {
125                 end = true;
126                 System.exit(0);
127             }
128             case 3 -> {
129                 end = true;
130                 System.exit(0);
131             }
132         }
133     }while(!end);
134 }
135 }
```

World.java

```
1  import java.util.ArrayList;
2  import java.util.List;
3  import java.io.File;
4  import java.io.FileNotFoundException;
5  import java.util.Scanner;
6  public class World {
7      private List<Island> islands;
8
9      World(){
10         this.islands = new ArrayList<Island>();
11     }
12
13     public void addIslands(Island s){
14         this.islands.add(s);
15     }
16
17     public short[] paintIslands(short[] myImage){
18         for(Island i : this.islands){
19             myImage = i.paint(myImage);
20         }
21         return myImage;
22     }
23
24     private boolean isWater(short[] myImage, int idx){
25         return (myImage[idx] == 0 && myImage[idx + 1] == 177 &&
               myImage[idx + 2] == 241);
26     }
27
28     private boolean canPlaceIsland(short[] myImage, int rad, int x, int
           y){
29         boolean ret = true;
30         for(int i = -rad; i <= rad; i++){
31             for(int j = -rad; j <= rad; j++){
32                 if((y+i) >= 0 && (y+i) <= 23 && (x+j) <= 47 && (x+j) >= 0
                       && isWater(myImage, (((y+i) * 48 + (x+j)) * 3)) &&
                       ret){
33                     ret = true;
34                 }else {
35                     return false;
36                 }
37             }
38         }
39         return ret;
40     }
41
42     public short[] createIsland(short[] myImage, int num){
43         while (num > 0){
44             int i = (int)(Math.random() * 45) + 1;
```

```
45            int j = (int)(Math.random() * 21) + 1;
46            if(canPlaceIsland(myImage,5,i,j)){
47                int[] size = {3,3};
48                if(Math.random() < 0.5){
49                    addIslands(new Island(size,i,j));
50                } else {
51                    addIslands(new Island(size,i,j,new
                          Harbor((int)(Math.random()*7)+1)));
52                }
53                myImage = paintIslands(myImage);
54                num--;
55            }
56        }
57        return myImage;
58    }
59
60    public short[] runHarbor(short[] myImage){
61        for(Island i : this.islands){
62            if(i.hasHarbor()){
63                myImage = i.getHarbor().run(-1,myImage);
64            }
65        }
66        return myImage;
67    }
68
69    public short[] clear(){
70        short[] ret = new short[24*48*3];
71        for(int i=0; i<ret.length; i+=3){
72            ret[i+0]=(short)0;
73            ret[i+1]=(short)177;
74            ret[i+2]=(short)241;
75        }
76        return ret;
77    }
78
79    public short[] fade(short[] myImage){
80        for (int i = 0; i < myImage.length; i += 3){
81            double gr = myImage[i + 1] / (myImage[i]+1);
82            double br = myImage[i + 2] / (myImage[i]+1);
83            myImage[i] = (short) (myImage[i] * 0.7);
84            myImage[i + 1] = (short) (myImage[i] * gr);
85            myImage[i + 2] = (short) (myImage[i] * br);
86        }
87        return myImage;
88    }
89
90    public short[] parseImage(String what){
91        short[] ret = new short[24*48*3];
92        Scanner myReader = null;
93        try {
```

```
94      myReader = new Scanner(new
            File("Game"+File.separator+"src"+File.separator+"de"+File.separator+"thdeg"+File.separator+"as
95              while (myReader.hasNextLine()) {
96                  String data = myReader.nextLine();
97                  if(data.length() > 1 && !data.substring(0,1).equals("/")){
98                  if(data.substring(0,1).equals("B")){
99                      for(int i = 0; i < ret.length; i += 3){
100                         String[] s = data.substring(1).split("-");
101                         ret[i] = Short.parseShort(s[0]);
102                         ret[i + 1] = Short.parseShort(s[1]);
103                         ret[i + 2] = Short.parseShort(s[2]);
104                     }
105                 }else{
106                     String[] pos = data.substring(0,
                            data.indexOf(':')).split("-");
107                     String[] rgb =
                            data.substring(data.indexOf(':')+1).split("-");
108                     ret[(Integer.parseInt(pos[1]) * 48 +
                            Integer.parseInt(pos[0])) * 3 + 0] =
                            Short.parseShort(rgb[0]);
109                     ret[(Integer.parseInt(pos[1]) * 48 +
                            Integer.parseInt(pos[0])) * 3 + 1] =
                            Short.parseShort(rgb[1]);
110                     ret[(Integer.parseInt(pos[1]) * 48 +
                            Integer.parseInt(pos[0])) * 3 + 2] =
                            Short.parseShort(rgb[2]);
111                 }
112                 }
113             }
114         } catch (FileNotFoundException e) {
115             e.printStackTrace();
116         }
117         myReader.close();
118         return ret;
119     }
120 }
```

Score.java

```java
public class Score {
    int score;

    Score(){}

    public void add(int val){
        this.score += val;
    }

    public void setScore(int score){
        this.score = score;
    }

    public int getScore(){
        return this.score;
    }
}
```