

Leistungsnachweis im Fach Programmierung 1

Maximilian von Hohenbüchel
Fabian Cieslik

2020-12-07

Contents

1	Projektvoraussetzungen	III
2	Idee	IV
3	Beschreibung	V
4	Programmablauf	VI
5	Klassendiagramm	VII
6	Programmcode	VIII

1 Projektvoraussetzungen

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



```

47         this.pos[2][1] = y;
48         break;
49     case 4:
50         this.pos[0][0] = x + 1;
51         this.pos[0][1] = y + 1;
52         this.pos[2][0] = x - 1;
53         this.pos[2][1] = y - 1;
54         break;
55     case 5:
56         this.pos[0][0] = x;
57         this.pos[0][1] = y + 1;
58         this.pos[2][0] = x;
59         this.pos[2][1] = y - 1;
60         break;
61     case 6:
62         this.pos[0][0] = x - 1;
63         this.pos[0][1] = y + 1;
64         this.pos[2][0] = x + 1;
65         this.pos[2][1] = y - 1;
66         break;
67     case 7:
68         this.pos[0][0] = x - 1;
69         this.pos[0][1] = y;
70         this.pos[2][0] = x + 1;
71         this.pos[2][1] = y;
72         break;
73     case 8:
74         this.pos[0][0] = x - 1;
75         this.pos[0][1] = y - 1;
76         this.pos[2][0] = x + 1;
77         this.pos[2][1] = y + 1;
78         break;
79     }
80     }else {
81         this.pos[0][0] = 2;
82         this.pos[0][1] = 3;
83         this.pos[1][0] = 2;
84         this.pos[1][1] = 2;
85         this.pos[2][0] = 2;
86         this.pos[2][1] = 1;
87         this.align = 7;
88     }
89 }
90
91 protected short[] clearTrace(short[] myImage){
92     for (int i = 0; i < this.oldpos.length; i++) {
93         myImage[(this.oldpos[i][1] * 48 + this.oldpos[i][0]) * 3 + 0]
94             = (short)0;
95         myImage[(this.oldpos[i][1] * 48 + this.oldpos[i][0]) * 3 + 1]
96             = (short)177;

```

```

95         myImage[(this.oldpos[i][1] * 48 + this.oldpos[i][0]) * 3 + 2]
96             = (short)241;
97     }
98     return myImage;
99 }
100 /**
101  * This method uses the Players values to update the map and return
102  * 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] =
111                 color[this.hp - 1][i][0];
112             myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 1] =
113                 color[this.hp - 1][i][1];
114             myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 2] =
115                 color[this.hp - 1][i][2];
116         }
117     }else {
118         for(int i=0; i < this.pos.length; i++){
119             myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 0] =
120                 0;
121             myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 1] =
122                 177;
123             myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 2] =
124                 241;
125         }
126     }
127     if(this.bullet != null){
128         myImage = this.bullet.clear(myImage);
129     }
130 }
131 return myImage;
132 }
133
134 public short[] isHit(short[] myImage){
135     for(int i = 0; i < this.pos.length; i++){
136         if (hitBullet(myImage, this.pos[i][0], this.pos[i][1])){
137             damage(1);
138         }
139     }
140     myImage = paint(myImage);
141     return myImage;
142 }
143
144 public boolean isAlive(){

```

```

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

```

```

182     protected void rotate(int dir){
183         if(dir == 0){ // Left
184             switch(this.align){
185                 case 1:
186                     if (this.pos[0][0] - 1 >= 0 && this.pos[2][0] + 1 <=
187                         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 <=
196                         47) {
197                         saveOldPos();
198                         this.pos[0][0]--;
199                         this.pos[2][0]++;
200                         changeAlign(-1);
201                     }
202                     break;
203                 case 3:
204                     if (this.pos[0][1] - 1 >= 0 && this.pos[2][1] + 1 <=
205                         23) {
206                         saveOldPos();
207                         this.pos[0][1]--;
208                         this.pos[2][1]++;
209                         changeAlign(-1);
210                     }
211                     break;
212                 case 4:
213                     if (this.pos[0][1] - 1 >= 0 && this.pos[2][1] + 1 <=
214                         23) {
215                         saveOldPos();
216                         this.pos[0][1]--;
217                         this.pos[2][1]++;
218                         changeAlign(-1);
219                     }
220                     break;
221                 case 5:
222                     if (this.pos[0][0] + 1 <= 47 && this.pos[2][0] - 1 >=
223                         0) {
224                         saveOldPos();
225                         this.pos[0][0]++;
226                         this.pos[2][0]--;
227                         changeAlign(-1);
228                     }
229                     break;
230                 case 6:
231                     if (this.pos[0][0] + 1 <= 47 && this.pos[2][0] - 1 >=

```

```

227         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 >=
236             0) {
237                 saveOldPos();
238                 this.pos[0][1]++;
239                 this.pos[2][1]--;
240                 changeAlign(-1);
241             }
242             break;
243     case 8:
244         if (this.pos[0][1] + 1 <= 23 && this.pos[2][1] - 1 >=
245             0) {
246                 saveOldPos();
247                 this.pos[0][1]++;
248                 this.pos[2][1]--;
249                 changeAlign(-1);
250             }
251             break;
252     }
253     }else { // Right
254         switch(this.align){
255             case 1:
256                 if (this.pos[0][0] + 1 <= 47 && this.pos[2][0] - 1 >=
257                     0) {
258                         saveOldPos();
259                         this.pos[0][0]++;
260                         this.pos[2][0]--;
261                         changeAlign(1);
262                     }
263                     break;
264             case 2:
265                 if (this.pos[0][1] + 1 <= 23 && this.pos[2][1] - 1 >=
266                     0) {
267                         saveOldPos();
268                         this.pos[0][1]++;
269                         this.pos[2][1]--;
270                         changeAlign(1);
271                     }
272                     break;
273             case 3:
274                 if (this.pos[0][1] + 1 <= 23 && this.pos[2][1] - 1 >=
275                     0) {
276                         saveOldPos();

```

```

271         this.pos[0][1]++;
272         this.pos[2][1]--;
273         changeAlign(1);
274     }
275     break;
276 case 4:
277     if (this.pos[0][0] - 1 >= 0 && this.pos[2][0] + 1 <=
278         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 <=
287         47) {
288         saveOldPos();
289         this.pos[0][0]--;
290         this.pos[2][0]++;
291         changeAlign(1);
292     }
293     break;
294 case 6:
295     if (this.pos[0][1] - 1 >= 0 && this.pos[2][1] + 1 <=
296         23) {
297         saveOldPos();
298         this.pos[0][1]--;
299         this.pos[2][1]++;
300         changeAlign(1);
301     }
302     break;
303 case 7:
304     if (this.pos[0][1] - 1 >= 0 && this.pos[2][1] + 1 <=
305         23) {
306         saveOldPos();
307         this.pos[0][1]--;
308         this.pos[2][1]++;
309         changeAlign(1);
310     }
311     break;
312 case 8:
313     if (this.pos[0][0] + 1 <= 47 && this.pos[2][0] - 1 >=
314         0) {
315         saveOldPos();
316         this.pos[0][0]++;
317         this.pos[2][0]--;
318         changeAlign(1);
319     }
320     break;

```

```

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

```

```

366         case 5:
367             saveOldPos();
368             for (int i = 0; i < this.pos.length; i++){
369                 this.pos[i][1]++;
370             }
371             break;
372         case 6:
373             saveOldPos();
374             for (int i = 0; i < this.pos.length; i++){
375                 this.pos[i][0]--;
376                 this.pos[i][1]++;
377             }
378             break;
379         case 7:
380             saveOldPos();
381             for (int i = 0; i < this.pos.length; i++){
382                 this.pos[i][0]--;
383             }
384             break;
385         case 8:
386             saveOldPos();
387             for (int i = 0; i < this.pos.length; i++){
388                 this.pos[i][0]--;
389                 this.pos[i][1]--;
390             }
391             break;
392     }
393 }
394 }
395
396 /**
397  * Used to determine if the ship can move forward.
398  * @return the returnvalue says, if the ship can move forward or if
399  *         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 &&
405             !hitIsland(myImage, this.pos[0][0], this.pos[0][1]-1) &&
406             !hitEnemy(myImage, this.pos[0][0], this.pos[0][1]-1));
407         case 2 -> ret = (this.pos[0][1] - 1 > 0 && this.pos[0][0] + 1
408             < 48 &&
409             !hitIsland(myImage, this.pos[0][0]+1, this.pos[0][1]-1) &&
410             !hitEnemy(myImage, this.pos[0][0]+1, this.pos[0][1]-1));
411         case 3 -> ret = (this.pos[0][0] + 1 < 48 &&
412             !hitIsland(myImage, this.pos[0][0]+1, this.pos[0][1]) &&
413             !hitEnemy(myImage, this.pos[0][0]+1, this.pos[0][1]));
414         case 4 -> ret = (this.pos[0][0] + 1 < 48 && this.pos[0][1] +
415             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));
407     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));
408     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));
409     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]));
410     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));
411     }
412     return ret;
413 }
414 /**
415  * 8 1 2
416  * 7 3
417  * 6 5 4
418  */
419 protected void shoot() {
420     int dir1 = (this.align + 2 > 8) ? (this.align + 2 - 8) :
        (this.align + 2);
421     // int dir2 = (this.align - 2 < 1) ? (8 + this.align - 2) :
        (this.align - 2);
422     if(this.bullet == null){
423         this.bullet = new Bullet(dir1, 5, this.pos[1][0],
            this.pos[1][1]);
424     }
425     // bullets.add(new Bullet(dir2, 5, this.pos[1][0],
        this.pos[1][1]));
426 }
427
428 protected boolean hitPlayer(short[] myImage, int x, int y){
429     if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
430         int idx = (y * 48 + x) * 3;
431         return (myImage[idx + 0] == 237 && myImage[idx + 1] == 76 &&
            myImage[idx + 2] == 36) ||
432             (myImage[idx + 0] == 237 && myImage[idx + 1] == 207 &&
            myImage[idx + 2] == 36) ||
433             (myImage[idx + 0] == 123 && myImage[idx + 1] == 237 &&
            myImage[idx + 2] == 36) ||
434             (myImage[idx + 0] == 145 && myImage[idx + 1] == 47 &&
            myImage[idx + 2] == 22) ||
435             (myImage[idx + 0] == 148 && myImage[idx + 1] == 129 &&
            myImage[idx + 2] == 22) ||

```

```

436         (myImage[idx + 0] == 74 && myImage[idx + 1] == 143 &&
437         myImage[idx + 2] == 21) ||
438         (myImage[idx + 0] == 74 && myImage[idx + 1] == 24 &&
439         myImage[idx + 2] == 11) ||
440         (myImage[idx + 0] == 66 && myImage[idx + 1] == 58 &&
441         myImage[idx + 2] == 10) ||
442         (myImage[idx + 0] == 38 && myImage[idx + 1] == 74 &&
443         myImage[idx + 2] == 11);
444     }else {
445         return false;
446     }
447 }
448
449 protected boolean hitEnemy(short[] myImage, int x, int y){
450     if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
451         int idx = (y * 48 + x) * 3;
452         return (myImage[idx + 0] == 31 && myImage[idx + 1] == 69 &&
453         myImage[idx + 2] == 222) ||
454         (myImage[idx + 0] == 19 && myImage[idx + 1] == 43 &&
455         myImage[idx + 2] == 143) ||
456         (myImage[idx + 0] == 10 && myImage[idx + 1] == 22 &&
457         myImage[idx + 2] == 74) ||
458         (myImage[idx + 0] == 31 && myImage[idx + 1] == 222 &&
459         myImage[idx + 2] == 215) ||
460         (myImage[idx + 0] == 21 && myImage[idx + 1] == 138 &&
461         myImage[idx + 2] == 134) ||
462         (myImage[idx + 0] == 11 && myImage[idx + 1] == 74 &&
463         myImage[idx + 2] == 72) ||
464         (myImage[idx + 0] == 153 && myImage[idx + 1] == 23 &&
465         myImage[idx + 2] == 209) ||
466         (myImage[idx + 0] == 94 && myImage[idx + 1] == 15 &&
467         myImage[idx + 2] == 128) ||
468         (myImage[idx + 0] == 55 && myImage[idx + 1] == 10 &&
469         myImage[idx + 2] == 74);
470     }else {
471         return false;
472     }
473 }
474
475 protected boolean hitBullet(short[] myImage, int x, int y){
476     if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
477         int idx = (y * 48 + x) * 3;
478         return (myImage[idx + 0] == 12 && myImage[idx + 1] == 13 &&
479         myImage[idx + 2] == 12);
480     }else {
481         return false;
482     }
483 }
484
485 protected boolean hitIsland(short[] myImage, int x, int y){

```

```

472     if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
473         int idx = (y * 48 + x) * 3;
474         return (myImage[idx + 0] == 196 && myImage[idx + 1] == 156 &&
                myImage[idx + 2] == 53) ||
475             (myImage[idx + 0] == 186 && myImage[idx + 1] == 148 &&
                myImage[idx + 2] == 48) ||
476             (myImage[idx + 0] == 125 && myImage[idx + 1] == 66 &&
                myImage[idx + 2] == 24);
477     }else {
478         return false;
479     }
480 }
481
482
483 /**
484  * This method is used to set the align variable after a succesful
485  * rotation
486  * @param dir the direction the ship rotates to
487  * */
488 protected void changeAlign(int dir){
489     this.align += dir;
490     if(this.align < 1){
491         this.align = 8;
492     }
493     if(this.align > 8){
494         this.align = 1;
495     }
496 }
497
498 public short[] run(int key, short[] myImage){
499     myImage = isHit(myImage);
500     if(key != -1){
501         myImage = clearTrace(myImage);
502         move(key,myImage);
503         if (collide(myImage) == 1){
504             resetMove();
505             if(key == 2){
506                 this.align++;
507             }
508             if(key == 3){
509                 this.align--;
510             }
511         }
512     }
513     if(this.bullet != null){
514         if(this.bullet.getRange() > 0){
515             this.bullet.run(-1, myImage);
516         }else{
517             this.bullet = null;
518         }
519     }

```

```

518     }
519     myImage = paint(myImage);
520     return myImage;
521 }
522
523 protected void resetMove(){
524     for(int i=0; i < this.pos.length; i++){
525         for(int j=0; j < this.pos[i].length; j++){
526             this.pos[i][j] = this.oldpos[i][j];
527         }
528     }
529 }
530
531 /**
532  * Debug method to print shiplocation and locationdifference between
533  * the new and old location.
534  */
535 public void print(String where){
536     System.out.println(where + "\nA: " + this.align);
537     for (int i = 0; i < this.pos.length; i++){
538         System.out.println("X: " + this.pos[i][0] + " Y: " +
539             this.pos[i][1] + " | Xo: " + this.oldpos[i][0] + " Yo: "
540             + this.oldpos[i][1]);
541         // System.out.println("(" + i + ") -> X: " + (this.pos[i][0]
542             - this.oldpos[i][0]) + " Y: " + (this.pos[i][1] -
543             this.oldpos[i][1]));
544     }
545 }
546
547 public int[][] getPos(){
548     return this.pos;
549 }
550
551 public int getHp(){
552     return this.hp;
553 }
554
555 public void setHp(int hp){
556     this.hp = (hp >= 0)? hp : 0;
557 }
558
559 protected void changeColor(short[][][] rgbs){
560     for (int i = 0; i < this.color.length; i++) {
561         for (int j = 0; j < this.color[0].length; j++){
562             for (int k = 0; k < this.color[0][0].length; k++){
563                 this.color[i][j][k] = (rgbs[i][j][k] <= 255 &&
564                     rgbs[i][j][k] >= 0)? rgbs[i][j][k] : 0;
565             }
566         }
567     }
568 }

```

562 }
563 }

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,
16                               72}},{31, 69, 222},{19, 43, 143},{10, 22, 74}},{153, 23,
17                               209},{94, 15, 128},{55, 10, 74}}};
18        changeColor(rgbs);
19        int[][] pos = { {22, 22}, {23, 22}, {24, 22} };
20        this.pos = pos;
21    }
22
23    Enemy(int hp, int x, int y, int o, int r){
24        super(hp, x, y, o);
25        short[][][] rgbs = {{{31, 222, 215},{21, 138, 134},{11, 74,
26                               72}},{31, 69, 222},{19, 43, 143},{10, 22, 74}},{153, 23,
27                               209},{94, 15, 128},{55, 10, 74}}};
28        changeColor(rgbs);
29        this.range = r;
30    }
31
32    public void resetDmg(){
33        this.dmg = 0;
34    }
35
36    public int getDamageReceived(){
37        return this.dmg;
38    }
39
40    /**
41     * Create Routes
42     */
43    private void pathFinder(){
44        this.routing = new ArrayList<int[]>();
45        /**
46         * Start: this.pos[1][0]=x
47         *         this.pos[1][1]=y
48         * 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             this.bullet.run(-1, myImage);
110         }else{
111             this.bullet = null;
112         }
113     }
114     myImage = paint(myImage);
115     return myImage;
116 }
117
118 private void move(short[] myImage){
119     if(this.routing.size() <= 0 ){
120         if(canMove(myImage)){
121             forward(myImage);
122         }else{
123             if(Math.random() > 0.5){
124                 rotate(0);
125             }else {
126                 rotate(1);
127             }
128         }
129     }else {
130         switch(routeDirection(this.pos[1][0], this.pos[1][1],
131             this.routing.get(this.routing.size() - 1))){
132             case 1 -> {
133                 rotateTo(1);
134                 forward(myImage);
135             }
136             case 2 -> {
137                 rotateTo(2);
138                 forward(myImage);
139             }
140             case 3 -> {
141                 rotateTo(3);
142                 forward(myImage);
143             }
144             case 4 -> {

```

```

144         rotateTo(4);
145         forward(myImage);
146     }
147     case 5 -> {
148         rotateTo(5);
149         forward(myImage);
150     }
151     case 6 -> {
152         rotateTo(6);
153         forward(myImage);
154     }
155     case 7 -> {
156         rotateTo(7);
157         forward(myImage);
158     }
159     case 8 -> {
160         rotateTo(8);
161         forward(myImage);
162     }
163     default -> {}
164 }
165 this.routing.remove(this.routing.size() - 1);
166 }
167 }
168
169 private void pR(){
170     for(int[] i : this.routing){
171         System.out.println("| " + i[0] + " | " + i[1] + " |");
172     }
173 }
174
175 private void rotateTo(int newOri){
176     while (this.align != newOri) {
177         rotate(1);
178     }
179 }
180
181 private int routeDirection(int x, int y, int[] gPos){
182     if(x > gPos[0]){
183         if(y > gPos[1]){
184             return 8;
185         }else if(y < gPos[1]){
186             return 6;
187         }else {
188             return 7;
189         }
190     }else if(x > gPos[0]){
191         if(y > gPos[1]) {
192             return 2;
193         }else if(y < gPos[1]){

```

```

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

```

```

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

```

```
290         for (int i = 0; i < this.pos.length; i++){
291             System.out.println("X: " + this.pos[i][0] + " Y: " +
292                 this.pos[i][1]);
292             System.out.println("Xo: " + this.oldpos[i][0] + " Yo: " +
293                 this.oldpos[i][1]);
293             // System.out.println("(" + i + ") -> X: " + (this.pos[i][0]
294                 - this.oldpos[i][0]) + " Y: " + (this.pos[i][1] -
295                 this.oldpos[i][1]));
294         }
295     }
296 }
```

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 &&
19             myImage[idx + 2] == 241);
20     }
21
22     public short[] employFleet(short[] myImage, int num){
23         while (num > 0){
24             int i = (int)(Math.random() * 46) + 1;
25             int j = (int)(Math.random() * 22) + 1;
26             if(
27                 isWater(myImage, (((j-1) * 48 + i ) * 3)) &&
28                 isWater(myImage, (((j-1) * 48 + (i+1)) * 3)) &&
29                 isWater(myImage, (((j-1) * 48 + (i-1)) * 3)) &&
30                 isWater(myImage, ((j * 48 + i ) * 3)) &&
31                 isWater(myImage, ((j * 48 + (i+1)) * 3)) &&
32                 isWater(myImage, ((j * 48 + (i-1)) * 3)) &&
33                 isWater(myImage, (((j+1) * 48 + i ) * 3)) &&
34                 isWater(myImage, (((j+1) * 48 + (i+1)) * 3)) &&
35                 isWater(myImage, (((j+1) * 48 + (i-1)) * 3))) {
36                 addFleetmember(new Enemy(2, i, j, 4, 8));
37                 myImage = paintFleet(myImage);
38                 num--;
39                 continue;
40             }
41         }
42         return myImage;
43     }
44
45     public int getNumberOfAliveShips(){
46         int ret = 0;
47         for (Enemy s : this.fleet){
48             if(s.isAlive()){
49                 ret++;
50             }
51         }
52     }
53 }
```

```

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             myImage = e.isHit(myImage);
116         }
117         return myImage;
118     }
119
120     public short[] paintFleet(short[] myImage){
121         for (Enemy s : this.fleet) {
122             myImage = s.paint(myImage);
123         }
124         return myImage;
125     }
126 }

```

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] =
19             (short)0;
20         myImage[(this.oldpos[1] * 48 + this.oldpos[0]) * 3 + 1] =
21             (short)177;
22         myImage[(this.oldpos[1] * 48 + this.oldpos[0]) * 3 + 2] =
23             (short)241;
24         return myImage;
25     }
26
27     public short[] clear(short[] myImage){
28         myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 0] = (short)0;
29         myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 1] =
30             (short)177;
31         myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 2] =
32             (short)241;
33         return myImage;
34     }
35
36     public short[] paint(short[] myImage){
37         myImage = clearTrace(myImage);
38         if(this.range > 0) {
39             myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 0] = (short)12;
40             myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 1] = (short)13;
41             myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 2] = (short)12;
42         }else {
43             myImage = clear(myImage);
44         }
45         return myImage;
46     }
47
48     public int collide(short[] myImage){
49         return 0;
50     }
51 }
```

```

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 &&
50                 myImage[idx + 2] == 36) ||
51                 (myImage[idx + 0] == 237 && myImage[idx + 1] == 207 &&
52                 myImage[idx + 2] == 36) ||
53                 (myImage[idx + 0] == 123 && myImage[idx + 1] == 237 &&
54                 myImage[idx + 2] == 36) ||
55                 (myImage[idx + 0] == 145 && myImage[idx + 1] == 47 &&
56                 myImage[idx + 2] == 22) ||
57                 (myImage[idx + 0] == 148 && myImage[idx + 1] == 129 &&
58                 myImage[idx + 2] == 22) ||
59                 (myImage[idx + 0] == 74 && myImage[idx + 1] == 143 &&
60                 myImage[idx + 2] == 21) ||
61                 (myImage[idx + 0] == 74 && myImage[idx + 1] == 24 &&
62                 myImage[idx + 2] == 11) ||
63                 (myImage[idx + 0] == 66 && myImage[idx + 1] == 58 &&
64                 myImage[idx + 2] == 10) ||
65                 (myImage[idx + 0] == 38 && myImage[idx + 1] == 74 &&
66                 myImage[idx + 2] == 11);
67         }else {
68             return false;
69         }
70     }
71
72     protected boolean hitEnemy(short[] myImage, int x, int y){
73         if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
74             int idx = (y * 48 + x) * 3;
75             return (myImage[idx + 0] == 31 && myImage[idx + 1] == 69 &&
76                 myImage[idx + 2] == 222) ||
77                 (myImage[idx + 0] == 19 && myImage[idx + 1] == 43 &&
78                 myImage[idx + 2] == 143) ||
79                 (myImage[idx + 0] == 10 && myImage[idx + 1] == 22 &&
80                 myImage[idx + 2] == 74) ||
81                 (myImage[idx + 0] == 31 && myImage[idx + 1] == 222 &&
82                 myImage[idx + 2] == 215) ||
83                 (myImage[idx + 0] == 21 && myImage[idx + 1] == 138 &&
84                 myImage[idx + 2] == 134) ||
85                 (myImage[idx + 0] == 11 && myImage[idx + 1] == 74 &&
86                 myImage[idx + 2] == 72) ||
87                 (myImage[idx + 0] == 153 && myImage[idx + 1] == 23 &&
88                 myImage[idx + 2] == 209) ||
89                 (myImage[idx + 0] == 94 && myImage[idx + 1] == 15 &&
90                 myImage[idx + 2] == 128) ||
91                 (myImage[idx + 0] == 55 && myImage[idx + 1] == 10 &&
92                 myImage[idx + 2] == 74);
93         }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 &&
84             myImage[idx + 2] == 53) ||
85             (myImage[idx + 0] == 186 && myImage[idx + 1] == 148 &&
86                 myImage[idx + 2] == 48) ||
87             (myImage[idx + 0] == 125 && myImage[idx + 1] == 66 &&
88                 myImage[idx + 2] == 24);
89     }else {
90         return false;
91     }
92 }
93
94 public void move(int dir,short[] myImage){}
95
96 private void saveOldPos(){
97     this.oldpos[0] = this.pos[0];
98     this.oldpos[1] = this.pos[1];
99 }
100
101 public boolean move(){
102     if(canMove()){
103         saveOldPos();
104         switch(this.direction){
105             case 1:
106                 this.pos[1]--;
107                 break;
108             case 2:
109                 this.pos[0]++;
110                 this.pos[1]--;
111                 break;
112             case 3:
113                 this.pos[0]++;
114                 break;
115             case 4:
116                 this.pos[0]++;
117                 this.pos[1]++;
118                 break;
119             case 5:
120                 this.pos[1]++;
121                 break;
122             case 6:
123                 this.pos[0]--;
124                 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 <
146             24);
147         case 5 -> ret = (this.pos[1] + 1 < 24);
148         case 6 -> ret = (this.pos[1] + 1 < 24 && this.pos[0] - 1 >=
149             0);
150         case 7 -> ret = (this.pos[0] - 1 >= 0);
151         case 8 -> ret = (this.pos[0] - 1 >= 0 && this.pos[1] - 1 >=
152             0);
153     }
154     return ret;
155 }
156
157 public short[] run(int key, short[] myImage){
158     if(this.range == this.maxRange){
159         if(move()){
160             myImage = paint(myImage);
161         }else {
162             this.range = 0;
163             myImage = clear(myImage);
164         }
165     }else{
166         if (!(hitEnemy(myImage, this.pos[0], this.pos[1]) ||
167             hitPlayer(myImage, this.pos[0], this.pos[1]) ||
168             hitIsland(myImage, this.pos[0], this.pos[1]))){
169             if(move()){
170                 myImage = paint(myImage);
171             }else {
172                 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

```
1 public class Harbor extends Agent{
2     protected short[] color = {125, 66, 24};
3     protected int orient;
4     protected boolean captured = false;
5     protected boolean possession = false;
6
7     Harbor(int orient){
8         this.orient = orient;
9     }
10
11     public int getOrient(){
12         return this.orient;
13     }
14
15     public short[] getColor() {
16         return this.color;
17     }
18
19     @Override
20     short[] paint(short[] myImage) {
21         return new short[0];
22     }
23
24     @Override
25     int collide(short[] myImage) {
26         return -1;
27     }
28
29     @Override
30     void move(int dir, short[] myImage) {
31     }
32
33     @Override
34     short[] run(int key, short[] myImage) {
35         return new short[0];
36     }
37 }
```

Island.java

```
1 import java.util.List;
2
3 public class Island extends Agent {
4     protected short[][] color = {{196, 156, 53},{186, 148, 48}}; //
5         normale Insel, Hafeninsel
6     protected int[][] pos;
7     protected int[] size;
8     protected Harbor harbor = null;
9
10    Island(int[] size, int x, int y){
11        this.pos = new int[size[0]][size[1]][2];
12        for(int i1 = 0; i1 < size[0]; i1++){
13            for(int i2 = 0; i2 < size[1]; i2++){
14                this.pos[i1][i2][0] = x + i1;
15                this.pos[i1][i2][1] = y + i2;
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 +
36                        this.pos[i1][i2][0]) * 3 + 0] = color[0][0]; //
37                        (y * 48 + x) * 3 + 0
38                    myImage[(this.pos[i1][i2][1] * 48 +
39                        this.pos[i1][i2][0]) * 3 + 1] = color[0][1]; //
40                        (y * 48 + x) * 3 + 1
41                    myImage[(this.pos[i1][i2][1] * 48 +
42                        this.pos[i1][i2][0]) * 3 + 2] = color[0][2]; //
43                        (y * 48 + x) * 3 + 2
44                }
45                else{
46                    myImage[(this.pos[i1][i2][1] * 48 +
47                        this.pos[i1][i2][0]) * 3 + 0] = color[1][0]; //
```

```

41         (y * 48 + x) * 3 + 0
myImage[(this.pos[i1][i2][1] * 48 +
42         this.pos[i1][i2][0]) * 3 + 1] = color[1][1]; //
        (y * 48 + x) * 3 + 1
myImage[(this.pos[i1][i2][1] * 48 +
43         this.pos[i1][i2][0]) * 3 + 2] = color[1][2]; //
        (y * 48 + x) * 3 + 2
44 switch(harbor.getOrient()){
45     case 1:
        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     break;
49     case 2:
50     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
51     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
52     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
53     break;
54     case 3:
55     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
56     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
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 + 2] = harbor.getColor()[2]; // (y *
        48 + x) * 3 + 2
58 break;
59 case 4:
60 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
61 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
62 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
63 break;
64 case 5:
65 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
66 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
67 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
68 break;
69 case 6:
70 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
71 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
72 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
73 break;
74 case 7:
75 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
76 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
77 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
78 break;
79 case 8:
80 myImage[(this.pos[0][0][1] * 48 +
        this.pos[0][0][0]) * 3 + 0] =
        harbor.getColor()[0]; // (y * 48 + x) * 3
        + 0
81 myImage[(this.pos[0][0][1] * 48 +
        this.pos[0][0][0]) * 3 + 1] =
        harbor.getColor()[1]; // (y * 48 + x) * 3
        + 1
82 myImage[(this.pos[0][0][1] * 48 +
        this.pos[0][0][0]) * 3 + 2] =
        harbor.getColor()[2]; // (y * 48 + x) * 3
        + 2
83 break;
84     }
85 }
86 }
87 }
88 return myImage;
89 }
90
91 @Override
92 int collide(short[] myImage) {

```

```
93         return -1;
94     }
95
96     @Override
97     void move(int dir, short[] myImage) {
98     }
99
100    @Override
101    short[] run(int key, short[] myImage) {
102        return myImage;
103    }
104
105 }
```

Player.java

```
1 public class Player extends Ship {
2     private int score = 0;
3     private boolean hit = false;
4     private int hitX;
5     private int hitY;
6
7     Player(int hp){
8         super(hp);
9         short[][] rgbs = {{237, 76, 36},{145, 47, 22},{74, 24,
10             11}},{237, 207, 36},{148, 129, 22},{66, 58, 10}},{123,
11             237, 36},{74, 143, 21},{38, 74, 11}};
12         changeColor(rgbs);
13
14         int[][] pos = { {5, 5}, {6, 5}, {7, 5} };
15         this.pos = pos;
16     }
17
18     Player(int hp, int x, int y, int o){
19         super(hp, x, y, o);
20         short[][] rgbs = {{237, 76, 36},{145, 47, 22},{74, 24,
21             11}},{237, 207, 36},{148, 129, 22},{66, 58, 10}},{123,
22             237, 36},{74, 143, 21},{38, 74, 11}};
23         changeColor(rgbs);
24     }
25
26     /**
27      * The method collide looks at the pixels of the ship and look if it
28      * collided with another object
29      */
30
31     public int collide(short[] myImage){
32         int ret = 0;
33         for(int i=0; i < this.pos.length; i++){
34             if (hitIsland(myImage, this.pos[i][0], this.pos[i][1])){
35                 ret = 1;
36             }
37             if (hitBullet(myImage, this.pos[i][0], this.pos[i][1])){
38                 damage(1);
39                 ret = 1;
40             }
41             if (hitEnemy(myImage, this.pos[i][0], this.pos[i][1])) {
42                 this.hit = true;
43                 this.hitX = this.pos[i][0];
44                 this.hitY = this.pos[i][1];
45                 damage(1);
46                 ret = 1;
47             }
48         }
49         return ret;
50     }
51 }
```

```

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

```

```

94
95     public int getScore(){
96         return this.score;
97     }
98
99     public void addScore(int val){
100         this.score += val;
101     }
102
103     /**
104      * Debug method to print shiplocation and locationdifference between
105      * the new and old location.
106      */
107     public void print(){
108         System.out.println("Your ship:\nA: " + this.align);
109         for (int i = 0; i < this.pos.length; i++){
110             System.out.println("X: " + this.pos[i][0] + " Y: " +
111                 this.pos[i][1]);
112             System.out.println("Xo: " + this.oldpos[i][0] + " Yo: " +
113                 this.oldpos[i][1]);
114             // System.out.println("(" + i + ") -> X: " + (this.pos[i][0]
115                 - this.oldpos[i][0]) + " Y: " + (this.pos[i][1] -
116                 this.oldpos[i][1]));
117         }
118     }
119 }

```

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, do not 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            for(int i=0; i<myImage.length; i+=3){
27                myImage[i+0]=(short)0;
28                myImage[i+1]=(short)177;
29                myImage[i+2]=(short)241;
30            }
31
32            System.out.println("Sending to displayThread");
33            Player p = new Player(3, 7, 7, 5);
34            Fleet fleet = new Fleet();
35            World world = new World();
36            myImage = world.createIsland(myImage,7);
37            myImage = fleet.employFleet(myImage, 3);
38            myImage = p.paint(myImage);
39            myImage = fleet.paintFleet(myImage);
40            InternalLedGameThread.showImage(myImage);
41            System.out.println("Drucken Sie eine beliebige Taste um das
        Spiel zu starten.");
42            while(true){
43                if(InternalLedGameThread.getKeyboard() != -1){
```

```

44         break;
45     }
46 }
47 while(p.isAlive()){
48     thisKey = InternalLedGameThread.getKeyboard();
49     myImage = p.run(thisKey, myImage);
50     if(p.getHit()){
51         fleet.distributeDamage(p.getHitX(), p.getHitY());
52     }
53     myImage = fleet.statusUpdate(myImage);
54     if(frame % 10 == 0) {
55         frame = 0;
56         myImage = fleet.executeOrders(myImage);
57     }
58     myImage = world.paintIslands(myImage);
59     InternalLedGameThread.showImage(myImage);
60     frame++;
61     Thread.sleep(100);
62     System.out.println("+++ " + (System.currentTimeMillis() -
        startTime) + " +++");
63     p.damage(fleet.damageControl());
64     fleet.resetDamageControl();
65     if((System.currentTimeMillis() - startTime) > roundtime){
66         myImage = fleet.employFleet(myImage, (3 -
        fleet.getNumberOfAliveShips()));
67         round++;
68         startTime = System.currentTimeMillis();
69         p.addScore(200);
70     }
71     if(fleet.getNumberOfAliveShips() == 0){
72         p.addScore(50);
73     }
74 }
75
76 p.addScore(fleet.getDead() * 50);
77 highscore.add(p.getScore());
78 Collections.sort(highscore);
79 Collections.reverse(highscore);
80 System.out.println("(" + round + ") - Score: " +
    p.getScore());
81 System.out.println("Highscores:");
82 for(int i = 0; i < highscore.size(); i++){
83     System.out.println("(" + (i+1) + ") -> " +
        highscore.get(i));
84 }
85
86 System.out.println("Wollen Sie noch eine Runde spielen?
    (Y/N)\n> ");
87 end = !scan.next().equals("Y");
88 }while(!end);

```


89 }
90 }

World.java

```
1 import java.util.ArrayList;
2 import java.util.List;
3
4 public class World {
5     private List<Island> islands;
6
7     World(){
8         this.islands = new ArrayList<Island>();
9     }
10
11     public void addIslands(Island s){
12         this.islands.add(s);
13     }
14
15     public short[] paintIslands(short[] myImage){
16         for(Island i : this.islands){
17             myImage = i.paint(myImage);
18         }
19         return myImage;
20     }
21
22     private boolean isWater(short[] myImage, int idx){
23         return (myImage[idx] == 0 && myImage[idx + 1] == 177 &&
24             myImage[idx + 2] == 241);
25     }
26
27     public short[] createIsland(short[] myImage, int num){
28         while (num > 0){
29             int i = (int)(Math.random() * 45) + 1;
30             int j = (int)(Math.random() * 21) + 1;
31             if(
32                 isWater(myImage, (((j+2) * 48 + i ) * 3)) &&
33                 isWater(myImage, (((j+2) * 48 + (i+1)) * 3)) &&
34                 isWater(myImage, (((j+2) * 48 + (i+2)) * 3)) &&
35                 isWater(myImage, ((j * 48 + i ) * 3)) &&
36                 isWater(myImage, ((j * 48 + (i+1)) * 3)) &&
37                 isWater(myImage, ((j * 48 + (i+2)) * 3)) &&
38                 isWater(myImage, (((j+1) * 48 + i ) * 3)) &&
39                 isWater(myImage, (((j+1) * 48 + (i+1)) * 3)) &&
40                 isWater(myImage, (((j+1) * 48 + (i+2)) * 3))) {
41                 int[] size = {3,3};
42                 if(Math.random() < 0.75){
43                     addIslands(new Island(size,i,j));
44                 } else {
45                     addIslands(new Island(size,i,j,new
46                         Harbor((int)(Math.random()*7)+1)));
47                 }
48             }
49             myImage = paintIslands(myImage);
50             num--;
51         }
52     }
53 }
```

```
47         continue;
48     }
49 }
50 return myImage;
51 }
52 }
```
