

Course: CS3642 Artificial Intelligence Section W01

Student name: Aaron Cummings

Student ID: 000857610

Assignment #: 1

Due Date: February 21, 2021

Signature:



Score:

Design of my Agent: (code on second page)

The agent I implemented is a simple reflex agent. The implementation is in Java using switch and if statements since the number of actions in the environment is limited.

Tasks that my agent can solve:

I wanted to task my agent to play a small text-based game I made. It is a simple roll playing game where you are told what environment you are in and what your options are, as well as your current player statuses. I found that following some basic rules, it was possible to get to a point where the game was nearly impossible to lose with some luck. Thus, I found this to be a perfect environment to implement a reflex agent. I was correct that with enough tries the agent could successfully make its chances of loosing almost zero, so for this implementation I am turning in there is a “winning status” at a point where the game will end.

```
Status: healthy
Level: 13
Gold: 46
you are at a:
Town
train rest move
Input Command
train
you hired someone to train you in your skills -15 gold

Status: healthy
Level: 14
Gold: 31
you are at a:
Town
train rest move
Input Command
train
you hired someone to train you in your skills -15 gold

Status: healthy
Level: 15
Gold: 16
you are at a:
Town
train rest move
Input Command
move

Status: healthy
Level: 15
Gold: 16
you are at a:
Enemy Level: 15
Attack Run
Input Command
run
you were injured trying to escape

Status: injured
Level: 15
Gold: 16
you are at a:
Enemy Level: 13
Run
Input Command
run
you were killed trying to escape
you died (Score: 10)
```

```
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Win at level 60, Highest score 55. Let reflex agent play? y/n or test
test
how many trials?
100
```

```
Status: injured
Level: 22
Gold: 35
you are at a:
Plains
Move
Input Command
move

Status: injured
Level: 22
Gold: 35
you are at a:
Enemy Level: 18
Run
Input Command
run
you were killed trying to escape
Number of trials: 100 Sum: 945 Avg: 9 Max: 55 Min: 0 Winner Count: 6
```

Agent Implementation (repeated in the source code)

```
World.java x Player.java x Cell.java x Environment.java x Enemy.java x Town.java x Chest.java x
282 //SIMPLE REFLEX AGENT
283 //Precepts currentPlayer.status , currentCell.getCellType()
284 //Given the Environment it is in, and it's 'status' returns a valid command
285 @ static String reflexAgent(World world){
286     String command = "";
287     switch (world.currentPlayer.Status){
288         case "healthy":
289             switch (world.currentCell.getCellType()){
290                 case "town" :
291                     if (world.currentPlayer.gold >= 25){
292                         command = "train";
293                     }else{
294                         command = "move";
295                     }
296                     break;
297                 case "env":
298                     command = "move";
299                     break;
300                 case "chest":
301                     command = "open";
302                     break;
303                 case "enemy":
304                     if (world.currentCell.getLevel() >= world.currentPlayer.level){
305                         command = "run";
306                     }else{
307                         command = "attack";
308                     }
309                     break;
310                 default:
311                     break;
312             }
313             break;
314             //the injured status limits actions, and changes others.
315         case "injured":
316             switch (world.currentCell.getCellType()){
317                 case "town" :
318                     if (world.currentPlayer.gold >= 10){
319                         command = "rest";
320                     }else{
321                         command = "move";
322                     }
323                     break;
324                 case "env":
325                     command = "move";
326                     break;
327                 case "chest":
328                     command = "open";
329                     break;
330                 case "enemy":
331                     command = "run";
332                     break;
333                 default:
334                     break;
335             }
336             break;
337         default:
338             break;
339     }
340     return command;
341 }
```

Source Code World.java

```
1  import java.io.IOException;
2  import java.util.*;
3
4  //Contains Main
5  public class World {
6
7      //keeps track of the number of cells traversed
8      int worldCells = 1;
9      //the current state of the environment
10     //environment history is not saved
11     Cell currentCell;
12     //current state of the player
13     Player currentPlayer = new Player();
14
15     //constructor, first state will always be an environment cell
16     public World() { currentCell = new Environment(); }
17
18
19
20     //Main
21     //Winning is set at player level 60
22     public static void main(String args[]) throws IOException {
23
24         //present the user with options to play the game, let the reflex agent play the game,
25         //or let the agent play the game several times
26         Scanner scanner = new Scanner(System.in);
27         System.out.println("Win at level 60, Highest score 55. Let reflex agent play? y/n or test");
28         String command = scanner.nextLine();
29         switch (command){
30             case "y":
31                 reflexAgent();
32                 break;
33             case "n":
34                 realPlayer();
35                 break;
36             case "test":
37                 System.out.println("how many trials?");
38                 int x = scanner.nextInt();
39                 int[] arr = new int[x];
40                 int sum = 0;
41                 int avg = 0;
42                 int min = arr[0];
43                 int max = arr[0];
44                 int wincount = 0;
45                 for (int i = 0 ; i < x; i++){
46                     arr[i]=reflexAgentTrial();
47                 }
48                 for (int y = 0; y < x ; y++){
49                     sum = sum +arr[y];
50                 }
51                 for (int g = 1; g < x ; g++) {
52                     if (min > arr[g]) {
53                         min = arr[g];
54                     }
55                 }
56                 for (int g = 1; g < x ; g++) {
57                     if (max < arr[g]) {
58                         max = arr[g];
59                     }
60                 }
61                 for (int g = 0; g < x ; g++) {
62                     if (55 == arr[g]) {
63                         wincount++;
64                     }
65                 }
66                 System.out.println("Average score: " + sum/x);
67                 System.out.println("Minimum score: " + min);
68                 System.out.println("Maximum score: " + max);
69                 System.out.println("Win count: " + wincount);
70             }
71         }
72     }
73 }
```

```

63         wincount++;
64     }
65 }
66
67     avg = sum/x;
68     System.out.println("Number of trials: "+x+" Sum: "+sum+" Avg: "+avg+" Max: "+max+" Min: "+min
69         +" Winner Count: "+wincount);
70     default: break;
71 }
72
73 //waits for input before terminating
74 System.in.read();
75 }
76
77 //precondition: valid command
78 //changes the state of the environment based on command
79 public void commandIn(String nextCommand) {
80     //the currentCell.command(nextCommand) decides if the input is to be accepted
81     //if it fails the environment will stay the same and it shows an error
82     if (currentCell.command(nextCommand) == true) {
83         switch (nextCommand){
84             case "move":
85                 getNextCell();
86                 break;
87             case "run":
88                 Random rand = new Random();
89                 int chance = rand.nextInt( bound: 100) +1;
90                 if (chance >= 30) {getNextCell();}
91                 else if (currentPlayer.Status == "healthy"){
92                     currentPlayer.Status = "injured";
93                     System.out.println("you were injured trying to escape");
94                     getNextCell();
95                 } else if (currentPlayer.Status == "injured") {
96                     currentPlayer.Status = "dead";
97                     System.out.println("you were killed trying to escape");
98                 }
99                 break;
100             case "attack":
101                 if (currentCell.attack() == true){
102                     Random rand2 = new Random();
103                     int gold = rand2.nextInt( bound: 10)+1;
104                     System.out.println("Defeated enemy and found "+gold+" gold");
105                     currentPlayer.addGold(gold);
106                     getNextCell();
107                 }else{
108                     currentPlayer.Status = "injured";
109                 }
110                 break;
111             case "rest":
112                 if (currentPlayer.gold>=10) {
113                     currentPlayer.Status = "healthy";
114                     System.out.println("you spent a night in the inn -10 gold");
115                     currentPlayer.gold = currentPlayer.gold - 10;
116                 }else{
117                     System.out.println("you need atleast 10 gold for a room to rest in the inn");
118                 }
119                 break;
120             case "train":
121                 if (currentPlayer.gold>=15) {
122                     currentPlayer.levelUp();

```

```

123         System.out.println("you hired someone to train you in your skills -15 gold");
124         currentPlayer.gold = currentPlayer.gold - 15;
125     }else{
126         System.out.println("you need atleast 15 gold to train your skills");
127     }
128     break;
129     case "open":
130         Random rand3 = new Random();
131         int chestgold = 10 + rand3.nextInt( bound: 41);
132         currentPlayer.addGold(chestgold);
133         System.out.println("the chest contained "+chestgold+" gold");
134         getNextCell();
135         break;
136     }
137 } else {
138     System.out.println("Invalid command");
139 }
140 }
141
142 //This method updates the current environment to the next one
143 //15/30 Environment Cell, 10/30 Enemy Cell, 1/30 Chest Cell, 4/30 Town Cell
144 //This is ugly, thought more I would add more.
145 public void getNextCell(){
146     worldCells++;
147     Random rand = new Random();
148     int cell = rand.nextInt( bound: 30) + 1;
149     switch (cell){
150         case 1:
151         case 2:
152         case 3:
153         case 4:
154         case 5:
155         case 6:
156         case 7:
157         case 8:
158         case 9:
159         case 10:
160         case 11:
161         case 12:
162         case 13:
163         case 14:
164         case 15:
165             currentCell = new Environment();
166             break;
167         case 16:
168         case 17:
169         case 18:
170         case 19:
171         case 20:
172         case 21:
173         case 22:
174         case 23:
175         case 24:
176         case 25:
177             currentCell = new Enemy(currentPlayer);
178             break;
179         case 26:
180             currentCell = new Chest();
181             break;
182

```

```

183         default:currentCell = new Town();
184     }
185 }
186
187
188 //Prints the current cell desc as well as the command options
189 public void getCell() throws IOException {...}
190
191
192 //Plays the game from user input
193 public static void realPlayer(){
194     World newWorld = new World();
195     Scanner scanner = new Scanner(System.in);
196     String command;
197     //Game continues until status is "dead", at Level 55, play dies of old age
198     try {
199         while (newWorld.currentPlayer.Status != "dead") {
200             System.out.println();
201             System.out.println("Status: " + newWorld.currentPlayer.Status);
202             System.out.println("Level: " + newWorld.currentPlayer.level);
203             System.out.println("Gold: " + newWorld.currentPlayer.gold);
204             newWorld.getCell();
205             System.out.println("Input Command");
206             command = scanner.nextLine();
207             newWorld.commandIn(command);
208             if (newWorld.currentPlayer.level >= 60){
209                 newWorld.currentPlayer.Status = "dead";
210                 System.out.println("you grew old and died");
211             }
212         }
213     } catch (IllegalStateException | NoSuchElementException | IOException e) {
214         // System.in has been closed
215         System.out.println("System.in was closed; exiting");
216     }
217     System.out.println("you died (Score: " + (newWorld.currentPlayer.level - 5) + ")");
218 }
219
220
221 //Plays the game from a reflex agent
222 static void reflexAgent(){
223     World newWorld = new World();
224     Scanner scanner = new Scanner(System.in);
225     String command;
226     //Game continues until status is "dead", at Level 55, play dies of old age
227     try {
228         while (newWorld.currentPlayer.Status != "dead") {
229             System.out.println();
230             System.out.println("Status: " + newWorld.currentPlayer.Status);
231             System.out.println("Level: " + newWorld.currentPlayer.level);
232             System.out.println("Gold: " + newWorld.currentPlayer.gold);
233             newWorld.getCell();
234             System.out.println("Input Command");
235             command = reflexAgent(newWorld);
236             System.out.println(command);
237             newWorld.commandIn(command);
238             if (newWorld.currentPlayer.level >= 60){
239                 newWorld.currentPlayer.Status = "dead";
240                 System.out.println("you grew old and died");
241             }
242         }
243     } catch (IllegalStateException | NoSuchElementException | IOException e) {
244         // System.in has been closed

```

```

247         System.out.println("System.in was closed; exiting");
248     }
249     System.out.println("you died (Score: " + (newWorld.currentPlayer.level - 5) + ")");
250 }
251
252 //returns score from the reflex agent playing the game
253 static int reflexAgentTrial(){
254     World newWorld = new World();
255     Scanner scanner = new Scanner(System.in);
256     String command;
257     //Game continues until status is "dead", at Level 55, play dies of old age
258     try {
259         while (newWorld.currentPlayer.Status != "dead") {
260             System.out.println();
261             System.out.println("Status: " + newWorld.currentPlayer.Status);
262             System.out.println("Level: " + newWorld.currentPlayer.level);
263             System.out.println("Gold: " + newWorld.currentPlayer.gold);
264             newWorld.getCell();
265             System.out.println("Input Command");
266             command = reflexAgent(newWorld);
267             System.out.println(command);
268             newWorld.commandIn(command);
269             if (newWorld.currentPlayer.level >= 60){
270                 newWorld.currentPlayer.Status = "dead";
271                 System.out.println("you grew old and died");
272             }
273         }
274     } catch (IllegalStateException | NoSuchElementException | IOException e) {
275         // System.in has been closed
276         System.out.println("System.in was closed; exiting");
277     }
278     return (newWorld.currentPlayer.level - 5);
279 }
280
281 //SIMPLE REFLEX AGENT
282 //Precepts currentPlayer.status , currentCell.getCellType()
283 //Given the Environment it is in, and it's 'status' returns a valid command
284 @ static String reflexAgent(World world){
285     String command = "";
286     switch (world.currentPlayer.Status){
287         case "healthy":
288             switch (world.currentCell.getCellType()){
289                 case "town" :
290                     if (world.currentPlayer.gold >= 25){
291                         command = "train";
292                     }else{
293                         command = "move";
294                     }
295                     break;
296                 case "env":
297                     command = "move";
298                     break;
299                 case "chest":
300                     command = "open";
301                     break;
302                 case "enemy":
303                     if (world.currentCell.getLevel() >= world.currentPlayer.level){
304                         command = "run";
305                     }else{
306                         command = "attack";
307                     }

```

```

307         }
308         break;
309     default:
310         break;
311     }
312     break;
313     //the injured status limits actions, and changes others.
314     case "injured":
315         switch (world.currentCell.getCellType()){
316             case "town" :
317                 if (world.currentPlayer.gold >= 10){
318                     command = "rest";
319                 }else{
320                     command = "move";
321                 }
322                 break;
323             case "env":
324                 command = "move";
325                 break;
326             case "chest":
327                 command = "open";
328                 break;
329             case "enemy":
330                 command = "run";
331                 break;
332             default:
333                 break;
334         }
335         break;
336     default:
337         break;
338 }
339 return command;
340 }
341 }
342

```

Player.java

```

1
2 public class Player {
3
4     public int gold = 20;
5     public int level = 5;
6
7     //status may be healthy, injured or dead
8     public String status = "healthy";
9
10    //increases the player level by 1
11    public void levelUp() { level++; }
12
13
14
15    //adds income to player gold
16    public void addGold(int income) { gold = gold+income; }
17
18
19 }
20
21

```


Cell.java

```
1 //Cells represent the state the player is in
2 public abstract class Cell {
3
4     public int level;
5
6     //returns true if the player can win an attack
7     abstract boolean attack();
8
9     //returns the commands the user can input depending on
10    //the player status
11    abstract void getOptions(String playerStatus);
12
13    //returns the name of the cell type
14    abstract String getCellType();
15
16    //prints the cell type
17    abstract void printCellType();
18
19    //returns true if the command is valid
20    abstract boolean command(String nextCommand);
21
22    //returns a non 0 if the current state has a level.
23    public abstract int getLevel();
24 }
25
```

Environment.java

```
1  import java.util.*;
2  //environment has only the option to move, the Env String is merely for immersion
3  public class Environment extends Cell {
4      private static String Env;
5      private int EnvType;
6
7      //constructor, variable Env has 1 random possibilities, Cave, Forest, Plains, Tundra
8      public Environment() {
9
10         Random rand = new Random();
11         int upperbound = 4;
12         EnvType = rand.nextInt(upperbound);
13
14         switch (EnvType) {
15             case 0: Env = "Cave";
16                 break;
17             case 1: Env = "Forest";
18                 break;
19             case 2: Env = "Plains";
20                 break;
21             case 3: Env = "Tundra";
22                 break;
23             default:
24                 throw new IllegalStateException("Unexpected value: " + EnvType);
25         }
26     }
27
28     //always false for this state
29     @Override
30     boolean attack() {
31         return false;
32     }
33
34     //returns the commands the user can input
35     @Override
36     void getOptions(String playerStatus) { System.out.println("Move"); }
37
38     //returns the name of the cell type
39     @Override
40     String getCellType() {
41         return "env";
42     }
43
44     //prints the name of the cell type
45     @Override
46     void printCellType() {
47         System.out.println(Env);
48     }
49
50     //returns to if the command is a valid command for this cell
51     @Override
52     boolean command(String nextCommand) {
53         boolean state;
54         switch (nextCommand) {
55             case "move": state = true;
56                 break;
57             default:
58                 state = false;
59         }
60     }
61
62     return state;
```

```
63         return state;
64     }
65
66     //returns level, 0 if this cell has no level
67     @Override
68     public int getLevel() {
69         return 0;
70     }
71
72
73 }
74
```

Enemy.java

```
1  import java.util.Random;
2
3  public class Enemy extends Cell {
4      Random rand = new Random();
5      public int level;
6      Player currentPlayer;
7      boolean attackBool;
8
9      //constructor
10     //int level falls into the range of half to one level over current player level
11     //attackBool is true if the player can successfully kill the enemy
12     //attackBool is true if the player level is higher than the enemy, at the same level
13     //it is 1/2 chance to be true, if the player level is lower it is false.
14     @ public Enemy (Player player){
15         currentPlayer = player;
16         level = (player.level/2)+(rand.nextInt( bound: player.level/2)+2);
17         if (player.level > level){
18             attackBool = true;
19         }else if (player.level == level){
20             int chance = rand.nextInt( bound: 2);
21             if (chance == 1){
22                 attackBool = true;
23             }else{
24                 attackBool = false;
25             }
26         }
27     }
28
29     //prints the valid commands for this cell type
30     @Override
31     void getOptions(String playerStatus) {
32         switch (playerStatus) {
33             case "healthy": System.out.println("Attack Run");
34             break;
35             case "injured": System.out.println("Run");
36         }
37     }
38
39     //returns the name of the cell
40     @Override
41     String getCellType() {
42         return "enemy";
43     }
44
45     //prints the name of the cell, for an enemy cell it also prints the level in the name
46     @Override
47     void printCellType() {
48         System.out.println("Enemy Level: "+level);
49     }
50
51     //returns true if nextCommand is a valid command
52     @Override
53     boolean command(String nextCommand) {
54         boolean state;
55         switch (nextCommand) {
56             case "run": state = true;
57             break;
58             case "attack": if (currentPlayer.Status == "injured") {
59                 state = false;
60             }else{
61                 state = true;
62             }
63         }
64     }
```

```

61         state = true;
62     }
63     break;
64     default:
65         state = false;
66     }
67
68     return state;
69 }
70
71 //returns the level of the cell
72 public int getLevel() {
73     return level;
74 }
75
76 //returns true id the player can win an attack
77 public boolean attack() { return attackBool; }
78 }
79
80
81

```

Town.java

```

1  public class Town extends Cell{
2      //Always returns false
3      @Override
4      boolean attack() { return false; }
5
6
7      //Prints the valid command options
8      @Override
9      void getOptions(String playerStatus) { System.out.println("train rest move"); }
10
11
12      //returns the type of cell
13      @Override
14      String getCellType() { return "town"; }
15
16
17      //prints the type of cell
18      @Override
19      void printCellType() { System.out.println("Town"); }
20
21
22      //returns true if nextCommand is a valid command
23      @Override
24      boolean command(String nextCommand) {
25          switch (nextCommand) {
26              case "rest":
27                  return true;
28              case "train":
29                  return true;
30              case "move":
31                  return true;
32              default:
33                  return false;
34          }
35      }
36
37
38      //returns level, 0 if there is none
39      @Override
40      public int getLevel() { return 0; }
41
42
43
44
45
46
47

```

Chest.java

```
1  public class Chest extends Cell{
2
3      //always returns false
4      @Override
5  boolean attack() {
6      return false;
7  }
8
9      //returns the valid commands
10     @Override
11  void getOptions(String playerStatus) {
12      System.out.println("open");
13  }
14
15     //returns the type of cell
16     @Override
17  String getCellType() {
18      return "chest";
19  }
20
21     //prints the type of cell
22     @Override
23  void printCellType() {
24      System.out.println("chest");
25  }
26
27     //returns true if nextCommand is a valid command
28     @Override
29  boolean command(String nextCommand) {
30      switch (nextCommand) {
31          case "open":
32              return true;
33          default: return false;
34      }
35  }
36
37     //returns level, zero if there is none
38     @Override
39  public int getLevel() {
40      return 0;
41  }
42  }
43
```