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 | Player.java | Cell.java | Environment.java | Enemy.java | Town.java | Chest.java

```java
//SIMPLE REFLEX AGENT
//Precepts currentPlayer.status , currentCell.getCellType()
//Given the Environment it is in, and it's 'status' returns a valid command
static String reflexAgent(World world){
    String command = "";
    switch (world.currentPlayer.Status){
        case "healthy":
            switch (world.currentCell.getCellType()){
                case "town" :
                    if (world.currentPlayer.gold >= 25){
                        command = "train";
                    }else{
                        command = "move";
                    }
                    break;
                case "env":
                    command = "move";
                    break;
                case "chest":
                    command = "open";
                    break;
                case "enemy":
                    if (world.currentCell.getLevel() >= world.currentPlayer.level){
                        command = "run";
                    }else{
                        command = "attack";
                    }
                    break;
                default:
                    break;
            }
            break;
        //the injured status limits actions, and changes others.
        case "injured":
            switch (world.currentCell.getCellType()){
                case "town" :
                    if (world.currentPlayer.gold >= 10){
                        command = "rest";
                    }else{
                        command = "move";
                    }
                    break;
                case "env":
                    command = "move";
                    break;
                case "chest":
                    command = "open";
                    break;
                case "enemy":
                    command = "run";
                    break;
                default:
                    break;
            }
            break;
        default:
            break;
    }
    return command;
}
```

Source Code World.java

```java
import java.io.IOException;
import java.util.*;

//Contains Main
public class World {

    //keeps track of the number of cells traversed
    int worldCells = 1;
    //the current state of the environment
    //environment history is not saved
    Cell currentCell;
    //current state of the player
    Player currentPlayer = new Player();

    //constructor, first state will always be an environment cell
    public World() { currentCell = new Environment(); }

    //Main
    //Winning is set at player level 60
    public static void main(String args[]) throws IOException {

        //present the user with options to play the game, let the reflex agent play the game,
        //or let the agent play the game several times
        Scanner scanner = new Scanner(System.in);
        System.out.println("Win at level 60, Highest score 55. Let reflex agent play? y/n or test");
        String command = scanner.nextLine();
        switch (command){
            case "y":
                reflexAgent();
                break;
            case "n":
                realPlayer();
                break;
            case "test":
                System.out.println("how many trials?");
                int x = scanner.nextInt();
                int[] arr = new int[x];
                int sum = 0;
                int avg = 0;
                int min = arr[0];
                int max = arr[0];
                int wincount = 0;
                for (int i = 0 ; i < x; i++){
                    arr[i]=reflexAgentTrial();
                }
                for (int y = 0; y < x ; y++){
                    sum = sum +arr[y];
                }
                for (int g = 1; g < x ; g++) {
                    if (min > arr[g]) {
                        min = arr[g];
                    }
                }
                for (int g = 1; g < x ; g++) {
                    if (max < arr[g]) {
                        max = arr[g];
                    }
                }
                for (int g = 0; g < x ; g++) {
                    if (55 == arr[g]) {
                        wincount++;
```

```
                    wincount++;
                }
            }

            avg = sum/x;
            System.out.println("Number of trials: "+x+" Sum: "+sum+" Avg: "+avg+" Max: "+max+" Min: "+min
                    +" Winner Count: "+wincount);
        default: break;
    }

    //waits for input before terminating
    System.in.read();
}

//precondition: valid command
//changes the state of the environment based on command
public void commandIn(String nextCommand) {
    //the currentCell.command(nextCommand) decides if the input is to be accepted
    //if it fails the environment will stay the same and it shows and error
    if (currentCell.command(nextCommand) == true) {
        switch (nextCommand){
            case "move":
                getNextCell();
                break;
            case "run":
                Random rand = new Random();
                int chance = rand.nextInt( bound: 100) +1;
                if (chance >= 30) {getNextCell();}
                else if (currentPlayer.Status == "healthy"){
                    currentPlayer.Status = "injured";
                    System.out.println("you were injured trying to escape");
                    getNextCell();
                } else if(currentPlayer.Status == "injured") {
                    currentPlayer.Status = "dead";
                    System.out.println("you were killed trying to escape");
                }
                break;
            case "attack":
                if (currentCell.attack() == true){
                    Random rand2 = new Random();
                    int gold = rand2.nextInt( bound: 10)+1;
                    System.out.println("Defeated enemy and found "+gold+" gold");
                    currentPlayer.addGold(gold);
                    getNextCell();
                }else{
                    currentPlayer.Status = "injured";
                }
                break;
            case "rest":
                if (currentPlayer.gold>=10) {
                    currentPlayer.Status = "healthy";
                    System.out.println("you spent a night in the inn -10 gold");
                    currentPlayer.gold = currentPlayer.gold - 10;
                }else{
                    System.out.println("you need atleast 10 gold for a room to rest in the inn");
                }
                break;
            case "train":
                if (currentPlayer.gold>=15) {
                    currentPlayer.levelUp();
```

```
                    System.out.println("you hired someone to train you in your skills -15 gold");
                    currentPlayer.gold = currentPlayer.gold - 15;
                }else{
                    System.out.println("you need atleast 15 gold to train your skills");
                }
                break;
            case "open":
                Random rand3 = new Random();
                int chestgold = 10 + rand3.nextInt( bound: 41);
                currentPlayer.addGold(chestgold);
                System.out.println("the chest contained "+chestgold+" gold");
                getNextCell();
                break;
        }
    } else {
        System.out.println("Invalid command");
    }
}

//This method updates the current environment to the next one
//15/30 Environment Cell, 10/30 Enemy Cell, 1/30 Chest Cell, 4/30 Town Cell
//This is ugly, thought more I would add more.
public void getNextCell(){
    worldCells++;
    Random rand = new Random();
    int cell = rand.nextInt( bound: 30) + 1;
    switch (cell){
        case 1:
        case 2:
        case 3:
        case 4:
        case 5:
        case 6:
        case 7:
        case 8:
        case 9:
        case 10:
        case 11:
        case 12:
        case 13:
        case 14:
        case 15:
            currentCell = new Environment();
            break;
        case 16:
        case 17:
        case 18:
        case 19:
        case 20:
        case 21:
        case 22:
        case 23:
        case 24:
        case 25:
            currentCell = new Enemy(currentPlayer);
            break;
        case 26:
            currentCell = new Chest();
            break;
```

```
            default:currentCell = new Town();
        }

    }

    //Prints the current cell desc as well as the command options
    public void getCell() throws IOException {...}

    //Plays the game from user input
    public static void realPlayer(){
        World newWorld = new World();
        Scanner scanner = new Scanner(System.in);
        String command;
        //Game continues until status is "dead", at Level 55, play dies of old age
        try {
            while (newWorld.currentPlayer.Status != "dead") {
                System.out.println();
                System.out.println("Status: " + newWorld.currentPlayer.Status);
                System.out.println("Level: " + newWorld.currentPlayer.level);
                System.out.println("Gold: " + newWorld.currentPlayer.gold);
                newWorld.getCell();
                System.out.println("Input Command");
                command = scanner.nextLine();
                newWorld.commandIn(command);
                if (newWorld.currentPlayer.level >= 60){
                    newWorld.currentPlayer.Status = "dead";
                    System.out.println("you grew old and died");
                }
            }
        } catch (IllegalStateException | NoSuchElementException | IOException e) {
            // System.in has been closed
            System.out.println("System.in was closed; exiting");
        }
        System.out.println("you died (Score: "+ (newWorld.currentPlayer.level - 5) + ")");
    }

    //Plays the game from a reflex agent
    static void reflexAgent(){
        World newWorld = new World();
        Scanner scanner = new Scanner(System.in);
        String command;
        //Game continues until status is "dead", at Level 55, play dies of old age
        try {
            while (newWorld.currentPlayer.Status != "dead") {
                System.out.println();
                System.out.println("Status: " + newWorld.currentPlayer.Status);
                System.out.println("Level: " + newWorld.currentPlayer.level);
                System.out.println("Gold: " + newWorld.currentPlayer.gold);
                newWorld.getCell();
                System.out.println("Input Command");
                command = reflexAgent(newWorld);
                System.out.println(command);
                newWorld.commandIn(command);
                if (newWorld.currentPlayer.level >= 60){
                    newWorld.currentPlayer.Status = "dead";
                    System.out.println("you grew old and died");
                }
            }
        } catch (IllegalStateException | NoSuchElementException | IOException e) {
            // System.in has been closed
```

```
            System.out.println("System.in was closed; exiting");
        }
        System.out.println("you died (Score: " + (newWorld.currentPlayer.level - 5) + ")");
    }

    //returns score from the reflex agent playing the game
    static int reflexAgentTrial(){
        World newWorld = new World();
        Scanner scanner = new Scanner(System.in);
        String command;
        //Game continues until status is "dead", at Level 55, play dies of old age
        try {
            while (newWorld.currentPlayer.Status != "dead") {
                System.out.println();
                System.out.println("Status: " + newWorld.currentPlayer.Status);
                System.out.println("Level: " + newWorld.currentPlayer.level);
                System.out.println("Gold: " + newWorld.currentPlayer.gold);
                newWorld.getCell();
                System.out.println("Input Command");
                command = reflexAgent(newWorld);
                System.out.println(command);
                newWorld.commandIn(command);
                if (newWorld.currentPlayer.level >= 60){
                    newWorld.currentPlayer.Status = "dead";
                    System.out.println("you grew old and died");
                }
            }
        } catch (IllegalStateException | NoSuchElementException | IOException e) {
            // System.in has been closed
            System.out.println("System.in was closed; exiting");
        }
        return (newWorld.currentPlayer.level - 5);
    }

    //SIMPLE REFLEX AGENT
    //Precepts currentPlayer.status , currentCell.getCellType()
    //Given the Environment it is in, and it's 'status' returns a valid command
    static String reflexAgent(World world){
        String command = "";
        switch (world.currentPlayer.Status){
            case "healthy":
                switch (world.currentCell.getCellType()){
                    case "town" :
                        if (world.currentPlayer.gold >= 25){
                            command = "train";
                        }else{
                            command = "move";
                        }
                        break;
                    case "env":
                        command = "move";
                        break;
                    case "chest":
                        command = "open";
                        break;
                    case "enemy":
                        if (world.currentCell.getLevel() >= world.currentPlayer.level){
                            command = "run";
                        }else{
                            command = "attack";
```

```
                    }
                    break;
                default:
                    break;
            }
            break;
        //the injured status limits actions, and changes others.
        case "injured":
            switch (world.currentCell.getCellType()){
                case "town" :
                    if (world.currentPlayer.gold >= 10){
                        command = "rest";
                    }else{
                        command = "move";
                    }
                    break;
                case "env":
                    command = "move";
                    break;
                case "chest":
                    command = "open";
                    break;
                case "enemy":
                    command = "run";
                    break;
                default:
                    break;
            }
            break;
        default:
            break;
    }
    return command;
  }
}

```

Player.java

```
public class Player {

    public int gold = 20;
    public int level = 5;

    //status may be healthy, injured or dead
    public String Status = "healthy";

    //increases the player level by 1
    public void levelUp() { level++; }

    //adds income to player gold
    public void addGold(int income) { gold = gold+income; }

}

```

Cell.java

```java
//Cells represent the state the player is in
public abstract class Cell {

    public int level;

    //returns true if the player can win an attack
    abstract boolean attack();

    //returns the commands the user can input depending on
    //the player status
    abstract void getOptions(String playerStatus);

    //returns the name of the cell type
    abstract String getCellType();

    //prints the cell type
    abstract void printCellType();

    //returns true if the command is valid
    abstract boolean command(String nextCommand);

    //returns a non 0 if the current state has a level.
    public abstract int getLevel();
}

```

Environment.java

```java
import java.util.*;
//enviornment has only the option to move, the Env String is merely for immersion
public class Environment extends Cell {
    private static String Env;
    private int EnvType;

    //constructor, variable Env has 1 random possibilities, Cave, Forest, Plains, Tundra
    public Environment() {

        Random rand = new Random();
        int upperbound = 4;
        EnvType = rand.nextInt(upperbound);

        switch (EnvType) {
            case 0: Env = "Cave";
                break;
            case 1: Env = "Forest";
                break;
            case 2: Env = "Plains";
                break;
            case 3: Env = "Tundra";
                break;
            default:
                throw new IllegalStateException("Unexpected value: " + EnvType);
        }
    }

    //always false for this state
    @Override
    boolean attack() {
        return false;
    }

    //returns the commands the user can input
    @Override
    void getOptions(String playerStatus) { System.out.println("Move"); }

    //returns the name of the cell type
    @Override
    String getCellType() {
        return "env";
    }

    //prints the name of the cell type
    @Override
    void printCellType() {
        System.out.println(Env);
    }

    //returns to if the command is a valid command for this cell
    @Override
    boolean command(String nextCommand) {
        boolean state;
        switch (nextCommand) {
            case "move": state = true;
                break;
            default:
                state = false;
        }

        return state;
```

```
        return state;
    }

    //returns level, 0 if this cell has no level
    @Override
    public int getLevel() {
        return 0;
    }


}
```

Enemy.java

```java
import java.util.Random;

public class Enemy extends Cell {
    Random rand = new Random();
    public int level;
    Player currentPlayer;
    boolean attackBool;

    //constructor
    //int level falls into the range of half to one level over current player level
    //attackBool is true if the player can succesfully kill the enemy
    //attackBool is true if the player level is higher than the enemy, at the same level
    //it is 1/2 chance to be true, if the player level is lower it is false.
    public Enemy (Player player){
        currentPlayer = player;
        level = (player.level/2)+(rand.nextInt( bound: player.level/2)+2);
        if (player.level > level){
            attackBool = true;
        }else if (player.level == level){
            int chance = rand.nextInt( bound: 2);
            if (chance == 1){
                attackBool = true;
            }else{
                attackBool = false;
            }
        }
    }

    //prints the valid commands for this cell type
    @Override
    void getOptions(String playerStatus) {
        switch (playerStatus) {
            case "healthy": System.out.println("Attack Run");
                break;
            case "injured": System.out.println("Run");
        }
    }

    //returns the name of the cell
    @Override
    String getCellType() {
        return "enemy";
    }

    //prints the name of the cell, for an enemy cell it also prints the level in the name
    @Override
    void printCellType() {
        System.out.println("Enemy Level: "+level);
    }

    //returns true if nextCommand is a valid command
    @Override
    boolean command(String nextCommand) {
        boolean state;
        switch (nextCommand) {
            case "run": state = true;
                break;
            case "attack": if (currentPlayer.Status == "injured") {
                state = false;
            }else{
                state = true;
            }
            default: state = false;
        }
        return state;
    }
}
```

```
                state = true;
            }
            break;
            default:
                state = false;
        }

        return state;
    }

    //returns the level of the cell
    public int getLevel() {
        return level;
    }

    //returns true id the player can win an attack
    public boolean attack() { return attackBool; }
}
```

Town.java

```
public class Town extends Cell{
    //Always returns false
    @Override
    boolean attack() { return false; }

    //Prints the valid command options
    @Override
    void getOptions(String playerStatus) { System.out.println("train rest move"); }

    //returns the type of cell
    @Override
    String getCellType() { return "town"; }

    //prints the type of cell
    @Override
    void printCellType() { System.out.println("Town"); }

    //returns true if nextCommand is a valid command
    @Override
    boolean command(String nextCommand) {
        switch (nextCommand) {
            case "rest":
                return true;
            case "train":
                return true;
            case "move":
                return true;
            default:
                return false;
        }
    }

    //returns level, 0 if there is none
    @Override
    public int getLevel() { return 0; }
}
```

Chest.java

```java
public class Chest extends Cell{

    //always returns false
    @Override
    boolean attack() {
        return false;
    }

    //returns the valid commands
    @Override
    void getOptions(String playerStatus) {
        System.out.println("open");
    }

    //returns the type of cell
    @Override
    String getCellType() {
        return "chest";
    }

    //prints the type of cell
    @Override
    void printCellType() {
        System.out.println("chest");
    }

    //returns true if nextCommand is a valid command
    @Override
    boolean command(String nextCommand) {
        switch (nextCommand) {
            case "open":
                return true;
            default: return false;
        }
    }

    //returns level, zero if there is none
    @Override
    public int getLevel() {
        return 0;
    }
}

```