DESIGN AND INPUT-OUTPUTS

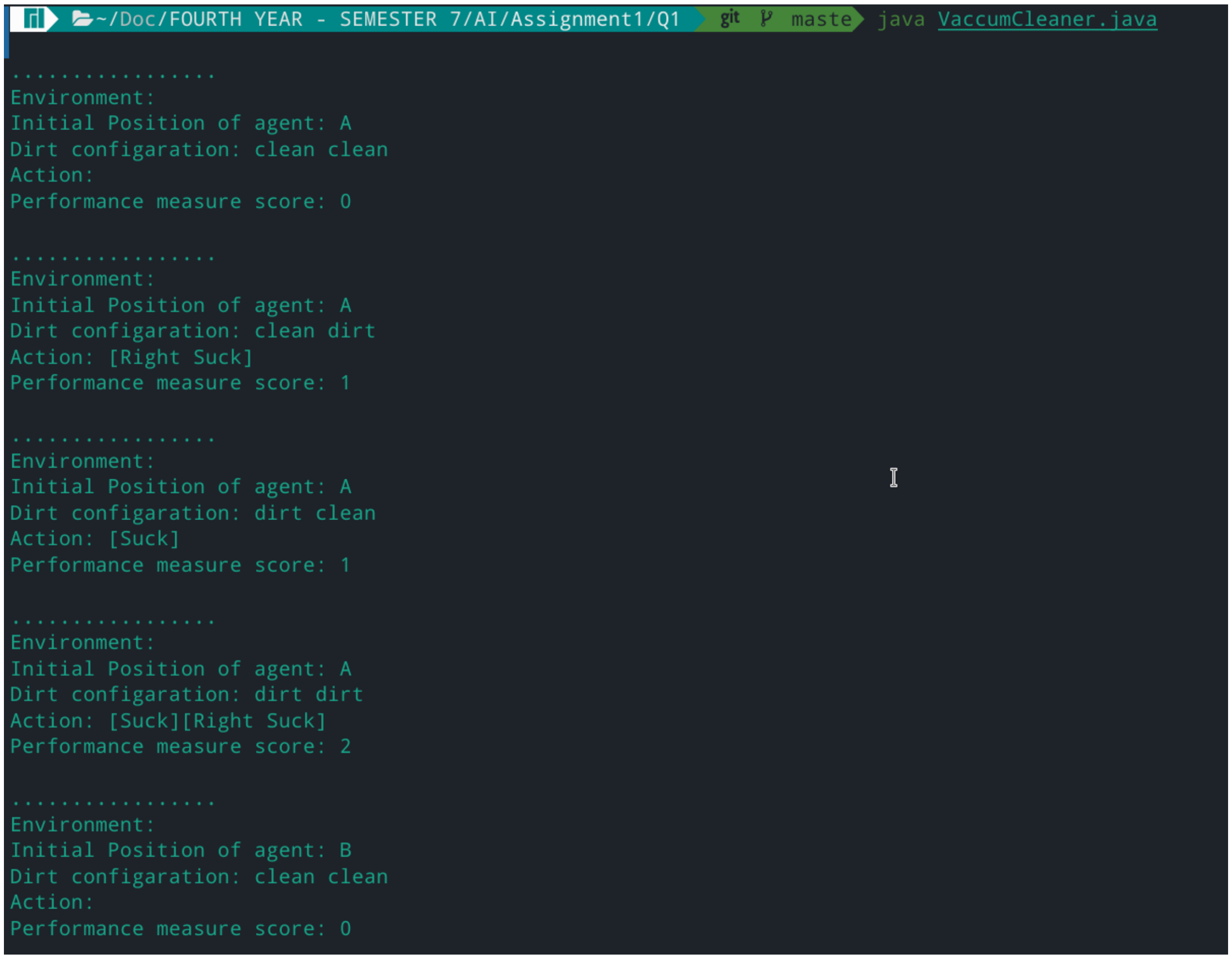
Pati Chandana
B180093CS

Q1 : Vacuum cleaner simple reflex agent

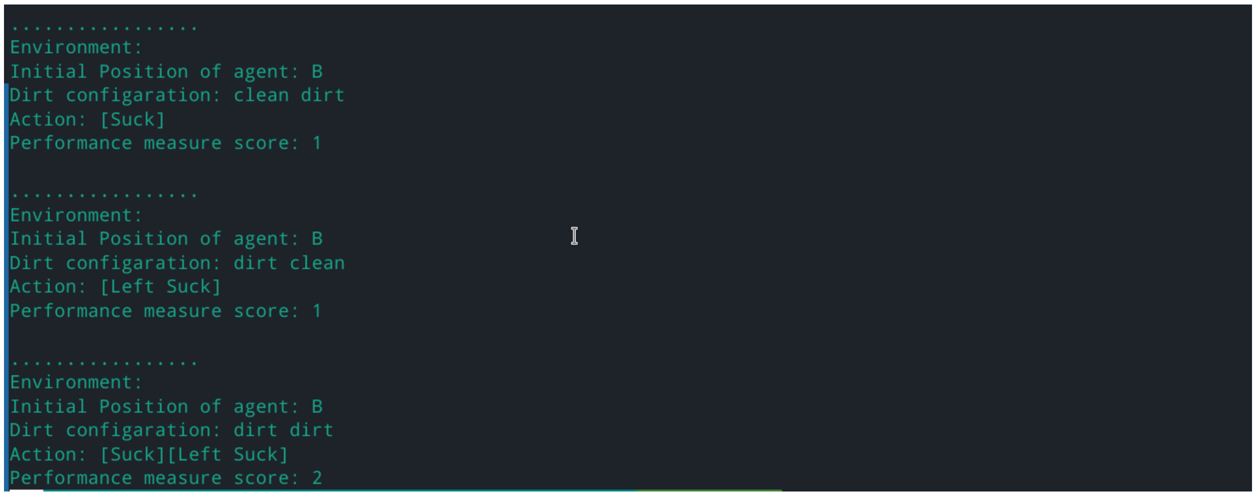
Design :

For the eight configurations based on environment and agent's position, if-else conditions and actions have been coded.

Output :



```
~/Doc/FOURTH YEAR - SEMESTER 7/AI/Assignment1/Q1  git  maste  java VaccumCleaner.java

.................
Environment:
Initial Position of agent: A
Dirt configaration: clean clean
Action:
Performance measure score: 0

.................
Environment:
Initial Position of agent: A
Dirt configaration: clean dirt
Action: [Right Suck]
Performance measure score: 1

.................
Environment:
Initial Position of agent: A
Dirt configaration: dirt clean
Action: [Suck]
Performance measure score: 1

.................
Environment:
Initial Position of agent: A
Dirt configaration: dirt dirt
Action: [Suck][Right Suck]
Performance measure score: 2

.................
Environment:
Initial Position of agent: B
Dirt configaration: clean clean
Action:
Performance measure score: 0
```

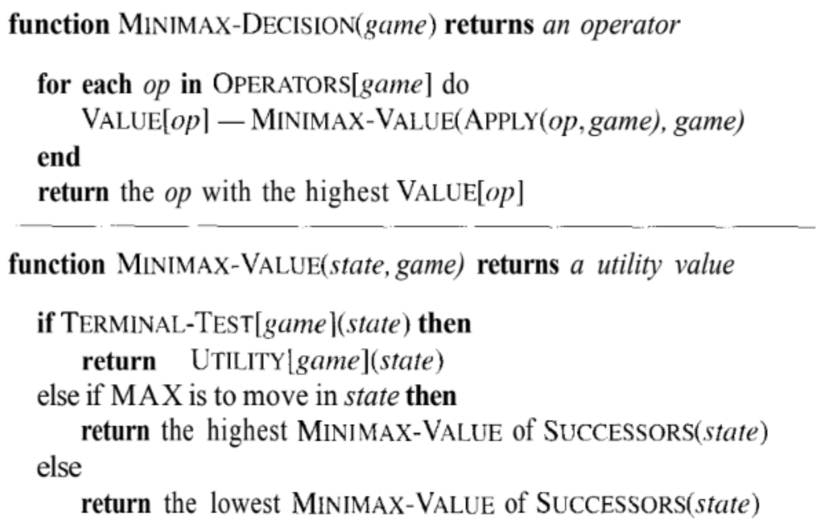


```
.................
Environment:
Initial Position of agent: B
Dirt configaration: clean dirt
Action: [Suck]
Performance measure score: 1

.................
Environment:
Initial Position of agent: B
Dirt configaration: dirt clean
Action: [Left Suck]
Performance measure score: 1

.................
Environment:
Initial Position of agent: B
Dirt configaration: dirt dirt
Action: [Suck][Left Suck]
Performance measure score: 2
```

We could see that it reaches a saturation point way before 1000 timesteps.

Q2 : MiniMax algorithm based game agent for Nim game with two piles of stones.

Design :



```
function MINIMAX-DECISION(game) returns an operator

  for each op in OPERATORS[game] do
      VALUE[op] — MINIMAX-VALUE(APPLY(op, game), game)
  end
  return the op with the highest VALUE[op]
------------------------------------------------------------
function MINIMAX-VALUE(state, game) returns a utility value

  if TERMINAL-TEST[game](state) then
      return   UTILITY[game](state)
  else if MAX is to move in state then
      return the highest MINIMAX-VALUE of SUCCESSORS(state)
  else
      return the lowest MINIMAX-VALUE of SUCCESSORS(state)
```

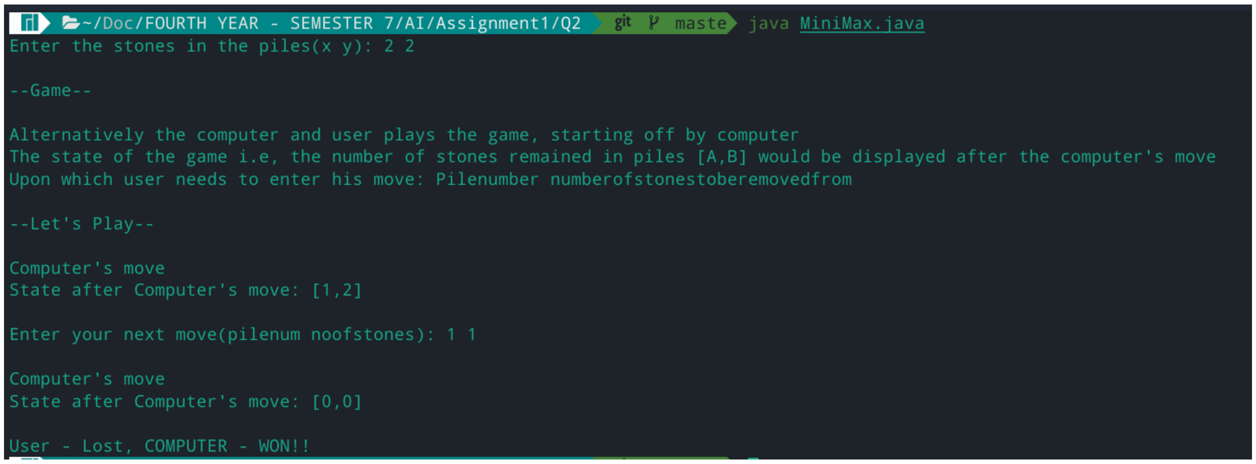
Pseudo code for the algorithm:

Alternatively, the user's input has been taken. Starting with the computer's chance, minimax_decision() function is called.

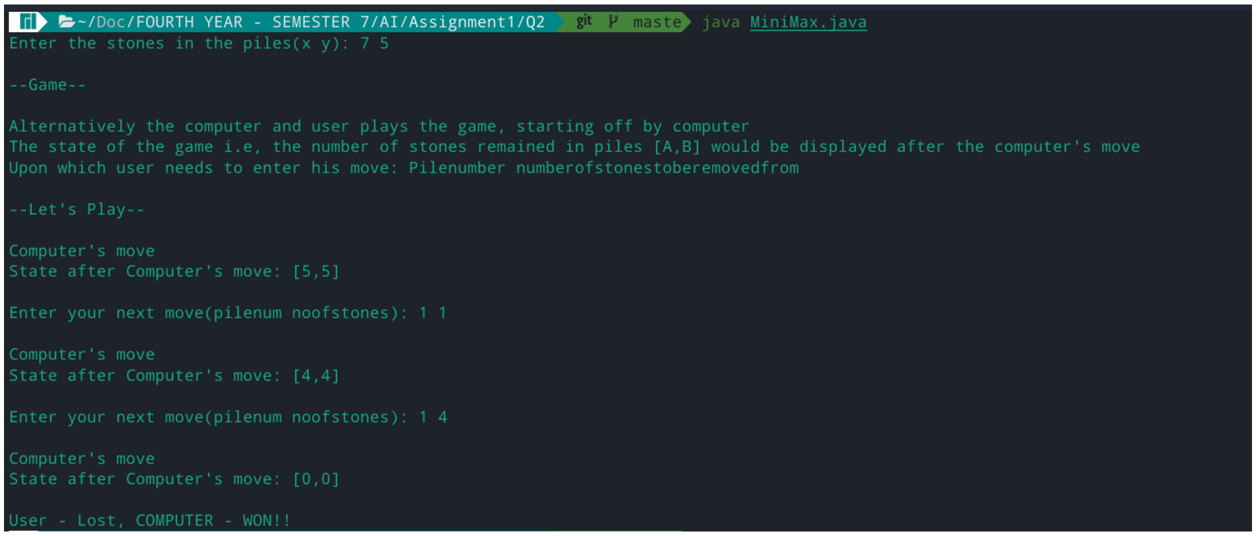
minimax_decision() : Given the current state of the game, the function goes through all the next possible states and calls minimax_value() function to get utility values for every state. After which, the state with the highest utility value is returned.

minimax_value(): Given a state, this function is used to find out the utility value of that state, in a recursive manner.
If the states are the leaf nodes, +1 or -1 are returned according to the player last moved. If the states are intermediate nodes, recursively the function is called to backtrack the utility values from the leaf nodes. If the move's min, then the least utility value among the next possible states is returned and viceversa if the player is max.

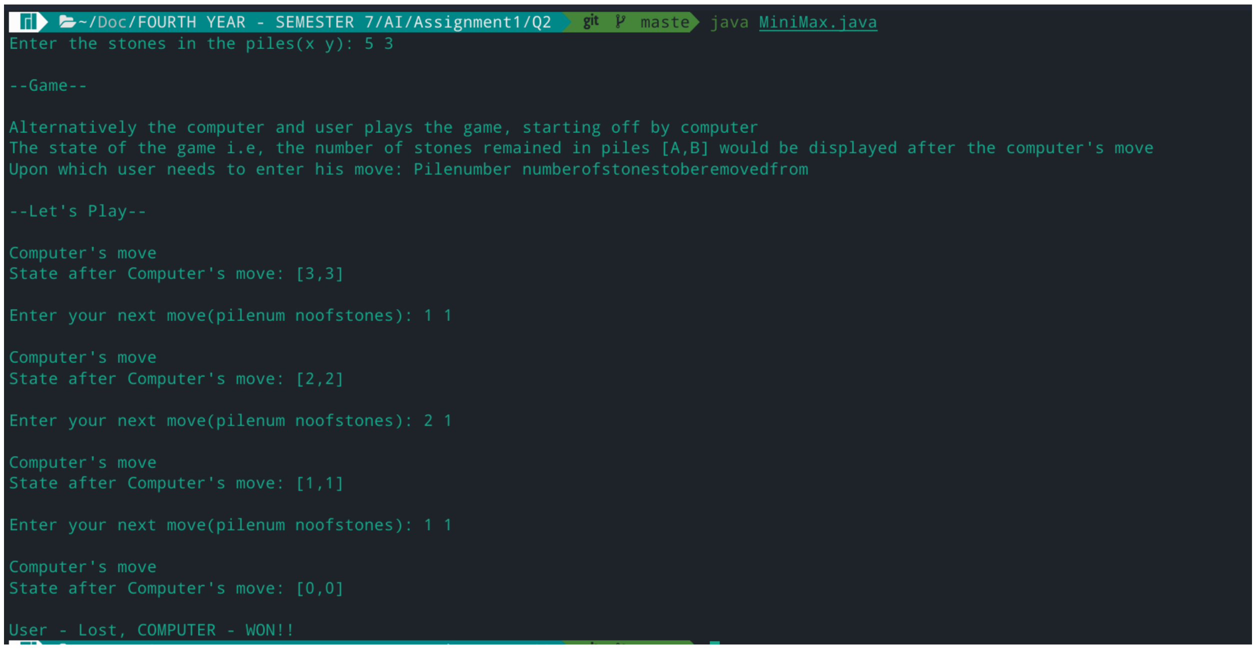
Input Output:



```
~/Doc/FOURTH YEAR - SEMESTER 7/AI/Assignment1/Q2  git  maste  java MiniMax.java
Enter the stones in the piles(x y): 2 2

--Game--

Alternatively the computer and user plays the game, starting off by computer
The state of the game i.e, the number of stones remained in piles [A,B] would be displayed after the computer's move
Upon which user needs to enter his move: Pilenumber numberofstonestoberemovedfrom

--Let's Play--

Computer's move
State after Computer's move: [1,2]

Enter your next move(pilenum noofstones): 1 1

Computer's move
State after Computer's move: [0,0]

User - Lost, COMPUTER - WON!!
```



```
~/Doc/FOURTH YEAR - SEMESTER 7/AI/Assignment1/Q2  git  maste  java MiniMax.java
Enter the stones in the piles(x y): 7 5

--Game--

Alternatively the computer and user plays the game, starting off by computer
The state of the game i.e, the number of stones remained in piles [A,B] would be displayed after the computer's move
Upon which user needs to enter his move: Pilenumber numberofstonestoberemovedfrom

--Let's Play--

Computer's move
State after Computer's move: [5,5]

Enter your next move(pilenum noofstones): 1 1

Computer's move
State after Computer's move: [4,4]

Enter your next move(pilenum noofstones): 1 4

Computer's move
State after Computer's move: [0,0]

User - Lost, COMPUTER - WON!!
```



```
 ~/Doc/FOURTH YEAR - SEMESTER 7/AI/Assignment1/Q2  git  maste  java MiniMax.java
Enter the stones in the piles(x y): 5 3

--Game--

Alternatively the computer and user plays the game, starting off by computer
The state of the game i.e, the number of stones remained in piles [A,B] would be displayed after the computer's move
Upon which user needs to enter his move: Pilenumber numberofstonestoberemovedfrom

--Let's Play--

Computer's move
State after Computer's move: [3,3]

Enter your next move(pilenum noofstones): 1 1

Computer's move
State after Computer's move: [2,2]

Enter your next move(pilenum noofstones): 2 1

Computer's move
State after Computer's move: [1,1]

Enter your next move(pilenum noofstones): 1 1

Computer's move
State after Computer's move: [0,0]

User - Lost, COMPUTER - WON!!
```