```cpp
//
// Created by Fabian Moik on 18.12.17.
//

#include "aiOwn.h"
#include <random>

AIOwn::AIOwn(){

}

AIOwn::~AIOwn() {

}

AIOwn::AIOwn(const AIOwn& ai) {
    topology_ = ai.topology_;
    inputValues_ = ai.inputValues_;
    resultValues_ = ai.resultValues_;

    name_ = ai.name_;
    net_ = ai.net_;
}

void AIOwn::setTopology (std::vector<unsigned> topo) {
    topology_ = topo;
}

Action AIOwn::doTurn() {

    Action action;
    net_.feedForward(inputValues_);
    net_.getResults(resultValues_);


    // Draw from a discrete distribution to get the action
    std::random_device rd;  //Will be used to obtain a seed for the random number engine
    std::mt19937 gen(rd()); //Standard mersenne_twister_engine seeded with rd()
    std::uniform_real_distribution<> dis(0.0, 1.0);

    double randomSample = dis(gen);
    int actionCommand = -1;

/*
```

```
45      //Choose via distribution
46      if (randomSample < resultValues_[0]) {
47          actionCommand = 0;
48      } else if (randomSample >= resultValues_[0] && randomSample < resultValues_[0] + resultValues_[1]) {
49          actionCommand = 1;
50      } else {
51          actionCommand = 2;
52      }
53      */
54
55      // Choose highest value
56      if (resultValues_[0] > resultValues_[1] && resultValues_[0] > resultValues_[2]) {
57          actionCommand = 0;
58      } else if (resultValues_[1] > resultValues_[0] && resultValues_[1] > resultValues_[2]) {
59          actionCommand = 1;
60      } else {
61          actionCommand = 2;
62      }
63
64      if (actionCommand == 0) {
65          action = Action(A_FOLD);
66      } else if (actionCommand == 1) {
67          // Check outside if A_CALL is valid, if not then A_CHECK
68          action = Action(A_CALL);
69      } else if (actionCommand == 2) {
70          //A raise could either be a small, medium or large raise
71          //TODO change this to a more appropriate method
72          action = Action(A_RAISE);
73          if (resultValues_[2] < 0.4) {
74              action.betType_ = B_SMALL_BET;
75          } else if (resultValues_[2] >= 0.4 && resultValues_[2] < 0.6) {
76              action.betType_ = B_MEDIUM_BET;
77          } else {
78              action.betType_ = B_LARGE_BET;
79          }
80      }
81      //Debug purpose
82      int a = 0;
83      return action;
84  }
85
86  std::string AIOwn::getAIName() {
87      return name_;
88  }
```

```
 89
 90 void AIOwn::setName(std::string name) {
 91     name_ = name;
 92 };
 93
 94 void AIOwn::fillInputValues(std::vector<double> &input) {
 95     // size of input should equal the number of input neurons of the first layer
 96
 97     if (input.size() == topology_[0]) {
 98         inputValues_ = input;
 99     } else {
100         std::cerr << "AI:   too few features provided" << std::endl;
101     }
102 }
103
104 std::vector<double> AIOwn::getResult() {
105     return resultValues_;
106 }
107
108
```

```
 1 //
 2 // Created by Fabian Moik on 18.12.17.
 3 //
 4
 5 #ifndef OWNPOKERSIMULATOR_AIOWN_H
 6 #define OWNPOKERSIMULATOR_AIOWN_H
 7
 8 #include "ai.h"
 9 #include "NN_agent/NeuralNet.h"
10
11 //Naive AI. This AI will do a random action, completely independent of its cards.
12 class AIOwn : public AI {
13 public:
14
15     // Creating a Neural Network with {numNeuron/layer0, numNeuron/layer1, ...}
16     //      -   for now just test it with some basic input value and generate a softmax output
17     //      -   the output represents (fold, check/call, bet/raise)
18     //          -   the raise output could be split into 3 parts (small r, medium r, large r) -> resulting in 5 output N.
19     std::vector<unsigned> topology_ = {16, 8, 3};
20     std::vector<double> inputValues_;
21     std::vector<double> resultValues_;
22
23     std::string name_ = "Default";
24
25     NeuralNet net_ = NeuralNet(topology_);
26
27     Action doTurn() override;
28
29     std::string getAIName() override;
30     void setName(std::string name) override;
31     void setTopology (std::vector<unsigned> topo);
32
33     AIOwn();
34     ~AIOwn();
35     // Copy Constructor
36     AIOwn(const AIOwn& ai);
37
38     void fillInputValues(std::vector<double> &input) override;
39     std::vector<double> getResult();
40
41 };
42
43 #endif //OWNPOKERSIMULATOR_AIOWN_H
44
```