Combining forward search with condition action allows the code to run for 10000 trails in a short duration.

To RUN CODE:
1. Delete all the .class files using rm *.class
2. To compile use javac WorldApplication.java
3. Run using java WorldApplication -a false -n false -t 10000

NOTE: Shooting on getting stench is not an optimal move. Since the position of the wumpus is not narrowed down the first time. The depth of -1 ensures that moving to a safe location is the best action and this also narrows down the position of the wumpus. So shooting and the left move and forward move is decided by the forward search.

CONDITION-ACTION-RULES:
There are two cases in which condition action is obvious choice.
1. If glitter if perceived then grabbing the gold is the best move.
2. If the forward state is not visited and it is safe to visit then moving forward is the best possible move. This move improves the performance of the model to a huge extend.

UTILITY:
This function changes how the agent works. I have used Depth Limited Search with a depth of 10. The depth is decided so that any unvisited square can be explored from

any given location. For each depth with corresponds to a move. I have given a reward of -1. And for shoot the reward is -10. Since the reward is -1 for each move, the agent will not visit any visited squares as visiting previously visited square will not give any reward.
Collocation with a wumpus gives a reward of -1000.
For every unvisited square the reward is not fixed. Since the price of the gold is +1000 the value of visiting unvisited square is:
Utility of unvisited square = 1000 / No. of unvisited squares

The formula used to count the pit is :
pitCount = pitCount * 2 + 1.
This makes sure that when the breeze is perceived from a different location. The chances of pit being in the particular square increases.

The total probability of the pit is calculated by:
probability of pit = 100 / (count of cluster of pit) * count of current pit.

To get value of being in the pit, we use
utility of pit = probability of pit * -1000

LOGIC USED:

The utility of NO_OP is given as 0. We try to take an action that gives the argmax of the utility.

The location of the pit is narrowed down using the manhattan distance. If the manhattan distance is greater than 2 then we have discovered 2 clusters of pit. So we can stop marking new pits after perceiving the breeze. This increases our state of knowledge about the world.

The Class called as IntPair is used to store the x and y coordinates of the pits. We use a hashset to save the coordinates of the possible pits locations and remove the coordinates once we are sure that the pit is not present there. At the start of each timestamp we calculate the manhattan distance between all the locations of the pits. If for any pit the closest pit is with a distance > 2 then we have found two distinct clusters. Overriding the equal method in this class is used to compare two objects.

The State Class is used to store the future and current nodes of the search space. It uses arrowUsed to keep a track of the arrow. It has a probability grid to calculate the probability of the pits. Its has one function to calculate the value of the safe square. The formula is given in the Utility section. It has another function to calculate the risk of entering a pit using probabilities. This is one of the main class of this project.

The Search Class is used to explore the tree. It takes the start position of the state as a input and generates a pruned tree to maximize the efficiency. The depth of the tree used in this is 10. This is to make sure that the arrow is not wasted on luck and the space is explored before

shooting the arrow.

Every function has comments on it to explain the logic used.