**APP Project**

“AI” Game Using Binary Tree in JAVA

Team Members:

Shivam Bansal [RA2211003010622]

Swachchatoya Samnta [RA2211003010620]

Kumar Mehul [RA2211003101624]

**Input File:  
akifood.txt**

Food Guessing Akinator

Guess a food cuisine at the start of the game and answer some questions for it. This akinator bot will try and guess the cuisine.

50 Is it Indian? (Y/N)

25 Is it form Southen India? (Y/N)

12 Is it Coastal? (Y/N)

6 Is Coconut a prominent Ingridient? (Y/N)

3 Its Kerala Cuisine.

7 Its Tamil Cuisine.

23 Does it include Tomato Rice? (Y/N)

20 Its Karnatak Cuisine.

24 Its Andra Cuisine.

40 Is it from Eastern India? (Y/N)

30 Does it include Fish? (Y/N)

26 Its Bengali Cuisine.

35 Its Uttrakhand Cuisine.

48 Does it use Basen as a prominent Ingridient? (Y/N)

45 Its Rajasthani Cuisine.

49 Its Maharastran Cuisine.

100 Does it have an Asian Origin? (Y/N)

60 Is it Stir Fried? (Y/N)

55 Is it Traditional Food? (Y/N)

51 Its Chinese Cuisine.

59 Its Korean Cuisine.

65 Is it Curry Based? (Y/N)

62 Its Thai Cuisine.

70 Its Japanese Cuisine.

120 Is it Grilled? (Y/N)

110 Is it Open Flame Grilled? (Y/N)

105 Its South African Cuisine.

115 Its Turkish Cuisine.

130 Is it Fried? (Y/N)

125 Its Amarican Cuisine.

135 Its Italian Cuisine.

**Source Code:**

# Define a class for a tree node

class TreeNode:

def \_\_init\_\_(self, data, question):

self.data = data

self.question = question

self.left = None

self.right = None

# Define the Quiz class

class Quiz:

def \_\_init\_\_(self):

self.root = None

# Method to insert a new node into the tree

def insert(self, data, question):

self.root = self.\_insert\_rec(self.root, data, question)

# Recursive helper method for inserting a new node

def \_insert\_rec(self, root, data, question):

if root is None:

return TreeNode(data, question)

if data < root.data:

root.left = self.\_insert\_rec(root.left, data, question)

elif data > root.data:

root.right = self.\_insert\_rec(root.right, data, question)

return root

# Method to query the tree and play the quiz

def query\_tree(self):

current\_node = self.root

while current\_node.left or current\_node.right:

self.print\_question(current\_node)

user\_input = self.get\_user\_input().strip().lower()

if (user\_input == "y" or user\_input == "yes") and current\_node.left:

current\_node = current\_node.left

elif (user\_input == "n" or user\_input == "no") and current\_node.right:

current\_node = current\_node.right

else:

print("Invalid input. Please enter 'Y' or 'N'.")

return f"Final Answer: {current\_node.question}"

# Method to read input from a file and build the tree

def read\_input\_from\_file(self, file\_name):

with open(file\_name, 'r') as file:

lines = file.readlines()

for line in lines[2:]:

data, question = line.strip().split(' ', 1)

self.insert(int(data), question)

# Method to display tree traversal options

def display\_tree(self):

while True:

print("Tree Traversal Options:")

print("1 - In-order Traversal")

print("2 - Pre-order Traversal")

print("3 - Post-order Traversal")

print("4 - Return to Main Menu")

choice = self.get\_user\_input()

traversal\_functions = {

"1": self.in\_order\_traversal,

"2": self.pre\_order\_traversal,

"3": self.post\_order\_traversal,

}

if choice in traversal\_functions:

print(f"{traversal\_functions[choice].\_\_name\_\_} Traversal:")

traversal\_functions[choice](self.root)

elif choice == "4":

return

else:

print("Invalid choice")

# Method for in-order tree traversal

def in\_order\_traversal(self, node):

if node is None:

return

self.in\_order\_traversal(node.left)

self.print\_node\_data(node)

self.in\_order\_traversal(node.right)

# Method for pre-order tree traversal

def pre\_order\_traversal(self, node):

if node is None:

return

self.print\_node\_data(node)

self.pre\_order\_traversal(node.left)

self.pre\_order\_traversal(node.right)

# Method for post-order tree traversal

def post\_order\_traversal(self, node):

if node is None:

return

self.post\_order\_traversal(node.left)

self.post\_order\_traversal(node.right)

self.print\_node\_data(node)

# Method to print a question from a tree node

def print\_question(self, node):

print(node.question)

# Method to print data from a tree node

def print\_node\_data(self, node):

print(node.data, node.question)

# Method to get user input

def get\_user\_input(self):

return input("Enter your choice: ").strip().lower()

# Main execution block

if \_\_name\_\_ == "\_\_main\_\_":

# Create an instance of the Quiz class

bt = Quiz()

file\_name = "akifood.txt"

bt.read\_input\_from\_file(file\_name)

# Main menu loop

while True:

print("Menu:")

print("P - Play Quiz")

print("L - Load Game File")

print("D - Display Tree")

print("H - Help")

print("X - Quit")

choice = bt.get\_user\_input()

if choice == "p":

final\_answer = bt.query\_tree()

print(final\_answer)

elif choice == "l":

print("Enter file path or name")

file\_name = bt.get\_user\_input().strip()

bt.read\_input\_from\_file(file\_name)

elif choice == "d":

bt.display\_tree()

elif choice == "h":

with open(file\_name, 'r') as file:

print(file.readline().strip())

print(file.readline().strip())

elif choice == "x":

break

else:

print("Invalid choice")

**Screenshots :**

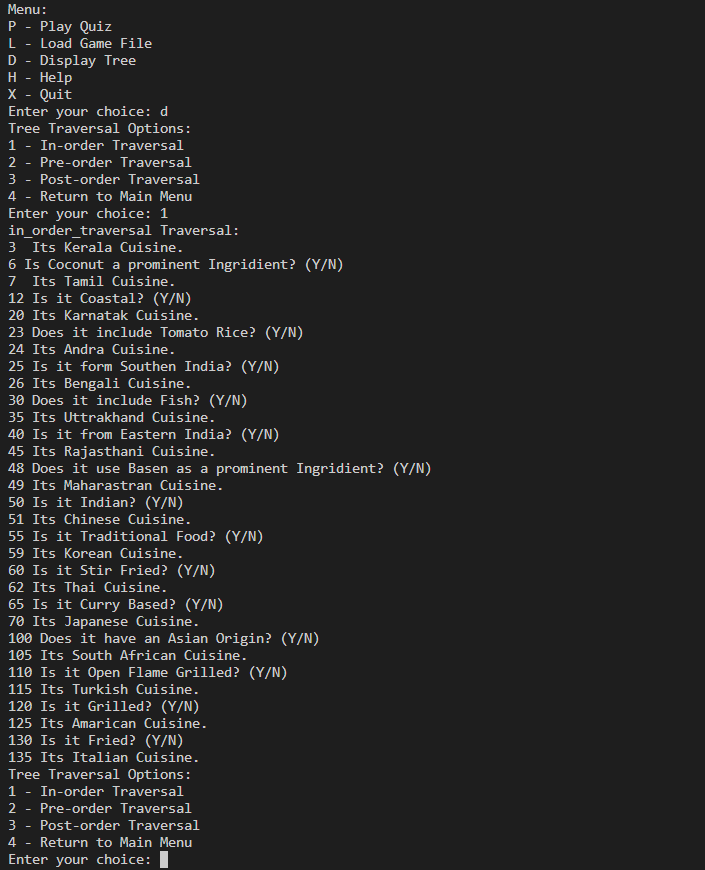


Fig 1. In-order Traversal

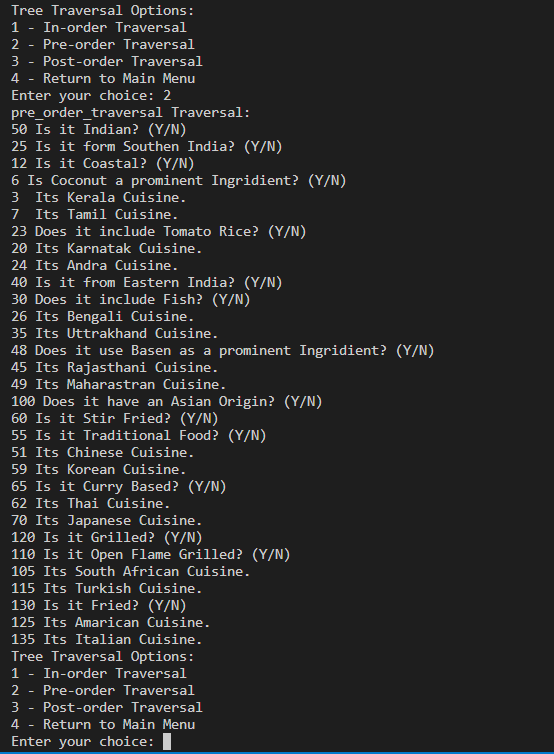


Fig 2. Pre-order Traversal

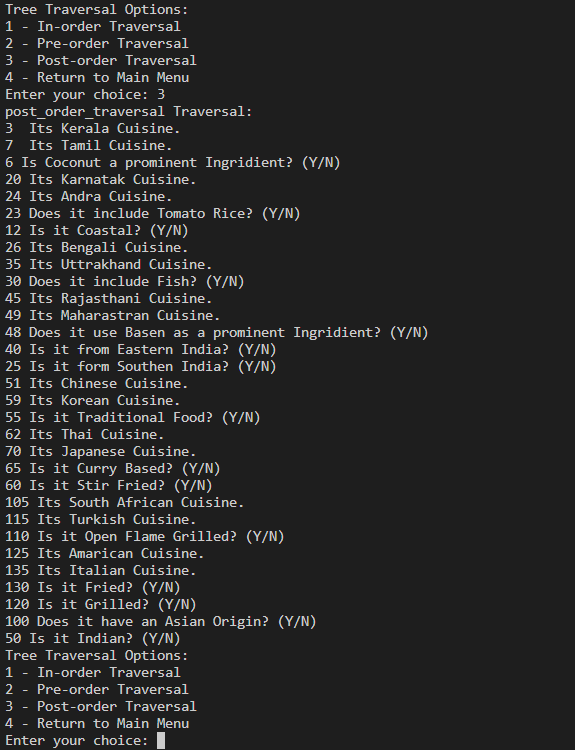


Fig 3. Post-order Traversal

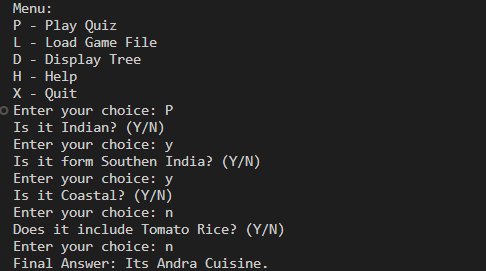


Fig 4. Animal Guessing Game