

## Homework 8

Ramzy Izza Wardhana - 21/472698/PA/20322

Make a knowledge tree that can be used to guess what things the user is thinking. Initially, the tree only knows an object, namely "Computer", and as it interacts with the user through yes/no questions, the tree will gather knowledge about the characteristics of other objects.

### BinaryTreeNode.java

```
package assignment;

public class BinaryTreeNode {
    String label;
    BinaryTreeNode left;
    BinaryTreeNode right;

    BinaryTreeNode(String label, BinaryTreeNode left, BinaryTreeNode right) {
        this.label = label;
        this.left = left;
        this.right = right;
    }

    public String toString() {
        return "[" + label + ", " + left + ", " + right + "]";
    }
}
```

### ShowBinaryTree.java

```
package assignment;

public class ShowBinaryTree {
    public static void prettyPrint(BinaryTreeNode root) {
        int treeHeight = getHeight(root);
        int totalWidth = (int) Math.pow(2, treeHeight) - 1;
        String[][] treeArray = new String[treeHeight][totalWidth];

        fillTreeArray(treeArray, root, 0, 0, totalWidth - 1);

        for (String[] row : treeArray) {
            for (String cell : row) {
                if (cell == null) {
                    System.out.print(" ");
                } else {
                    System.out.print(cell);
                }
            }
        }
    }
}
```

```

        }
    }
    System.out.println();
}

}

public static int getHeight(BinaryTreeNode node) {
    if (node == null) {
        return 0;
    }

    return Math.max(getHeight(node.left), getHeight(node.right)) + 1;
}

public static void fillTreeArray(String[][] treeArray, BinaryTreeNode node,
int level, int start, int end) {
    if (node == null) {
        return;
    }

    int mid = (start + end) / 2;
    treeArray[level][mid] = node.label;

    fillTreeArray(treeArray, node.left, level + 1, start, mid - 1);
    fillTreeArray(treeArray, node.right, level + 1, mid + 1, end);
}
}

```

## TreeKnowledge.java

```

package assignment;

import java.util.Scanner;

public class TreeKnowledge {
    Scanner sc;
    BinaryTreeNode root, newNode, newParent, parent;

    TreeKnowledge(Scanner sc) {
        this.sc = sc;
        root = new BinaryTreeNode("Computer", null, null);
    }

    public boolean YorN() {
        while (true) {

```

```

        String s = sc.next();
        if (s.startsWith("y"))
            return true;
        else if (s.startsWith("n"))
            return false;
    }
}

public void run() {
    while (true) {
        BinaryTreeNode x = root;
        System.out.println("Imagine an object!");
        while (true) {
            if (x.left == null && x.right == null) {
                System.out.print("Is the object you are thinking is " +
x.label + " (y/n)? ");
                if (YorN()) {
                    System.out.println("Then your object should be " +
x.label);

                    break;
                }
            }
            else {
                System.out.print("What is your object? ");
                String ans = sc.next();
                System.out.print("Enter the question if the answer is YES
it is " + ans + " and if the answer is NO it is a " + x.label + ": ");
                String question = sc.next();
                newNode = new BinaryTreeNode(ans, null, null);
                newParent = new BinaryTreeNode(question, newNode, x);
                if (x == root)
                    root = newParent;
                else {
                    BinaryTreeNode parent = getParent(root, x);
                    if (parent != null) {
                        if (parent.left == x)
                            parent.left = newParent;
                        else
                            parent.right = newParent;
                    }
                }
            }
            break;
        }
    }
    else {

```

```

        System.out.print(x.label + " (y/n)? ");
        if (YorN())
            x = x.left;
        else
            x = x.right;
    }
}
System.out.print("Continue (y/n)? ");
if (!YorN()){
    System.out.println("Result of the knowledge tree:");
    ShowBinaryTree.prettyPrint(root);
    break;
}
}
}

public BinaryTreeNode getParent(BinaryTreeNode current, BinaryTreeNode child)
{
    if (current == null || current.left == child || current.right == child)
        return current;
    parent = getParent(current.left, child);
    if (parent == null)
        parent = getParent(current.right, child);
    return parent;
}

public static void main(String[] args) {
    TreeKnowledge tk = new TreeKnowledge(new Scanner(System.in));
    tk.run();
}
}

```

### Explanation on Tree Knowledge:

1. First, the main logic of the program is defined in the run() method in which the program starts with a while loop and the stop condition is input from the user (y/n)

```

public void run() {
    while (true) {

```

2. At the beginning of the while loop, x is set as the variable that stores the knowledge tree. Initially, the x = root is "Computer", which is the first knowledge that the program would ask the user

```

    BinaryTreeNode x = root;
    System.out.println("Imagine an object!");

```

3. Next, we introduce a new while loop inside our current loop. This new while loop will help us navigate through nodes according to the decision made by the user input.

```
while (true) {
```

4. As for the first condition to check (x.left == null && x.right == null) act as the method to check whether our current visited node is a leaf or not, i.e. with no child at all.

```
if (x.left == null && x.right == null) {
```

5. If suppose that this condition is satisfied, then we reached the end of the node and ask for the information contained in that node using x.label to the user through input.

```
System.out.print("Is the object you are thinking is " + x.label + " (y/n)?");
```

6. In this if statement, there 2 conditions, particularly:

- a. When the user said yes = The program will confirm that the object they guessed is true and break the while loop. This will trigger the final state, where the program will ask the user's prompt whether they want to continue or not. If they want to continue then restart into the first state of the program, i.e. start from the root node. Otherwise, the program will break the outer loop and would be terminated.

```
if (YorN()) {  
    System.out.println("Then your object should be " + x.label);  
    break;  
}
```

- b. When the user said no = Implies that the answer the user wanted is not available on the knowledge tree. Therefore, the program will ask for the answer along with the relevant question. This answer and question will then be used as the main component of creating a new node for updating the new tree knowledge

```
else {  
    System.out.print("What is your object? ");  
    String ans = sc.next();  
    System.out.print("Enter the question if the answer is YES it is " + ans + " and if the answer is NO it is a " + x.label + ": ");  
    String question = sc.next();
```

7. In order to update the tree knowledge, we first need to declare two new nodes, where the first one is called newNode in which the root node is the answer itself and contains no child (leaf), while the second one is newParent, which is the parent of the newNode. To do this, we can utilize the previously written constructor:

```
newNode = new BinaryTreeNode(ans, null, null);  
newParent = new BinaryTreeNode(question, newNode, x);
```

8. After that, we check if the current node *x* is the same as the root node. If yes, then re-construct the tree with the previously created parent above. This usually only occurs during the first construction of the knowledge tree only. After the knowledge tree has at least one root and two children, the condition will be different, which will be explained next.

```
if (x == root)
    root = newParent;
```

9. In the next case, since the depth of the tree is more than one, we need to find the parent of our current visited node in order to successfully update the correct tree. This can be done by using the `getParent()` method. The logic behind `getParent()` method is to recursively traverse starting from the root, until our current visited node. This method will return the parent of our node and if no parent is found, then return null.

```
public BinaryTreeNode getParent(BinaryTreeNode current, BinaryTreeNode
child) {
    if (current == null || current.left == child || current.right ==
child)
        return current;
    parent = getParent(current.left, child);
    if (parent == null)
        parent = getParent(current.right, child);
    return parent;
}
```

10. Now, after obtaining the parent of our node, we can manipulate the knowledge tree with this condition:
- We replace the parent's left child with the new node (containing the question and answer from the user) if `parent.left` matches with *x*
  - Otherwise, we replace the parent's right child with the new node, also containing the same value of question and answer from the user.

This approach is done to update the knowledge tree correctly and can automatically decide on how the node (left or right child) is replaced with the new value from the user.

11. The last part of the inner while loop is the traversal process. This process is decided by the user itself whether they will traverse to the left-hand side or right-hand side of the knowledge tree. The decision is based on the question available and made on the knowledge tree (*x.label*). Side note: As far as I know after taking a machine learning class, in the decision tree, we usually go left if the condition is true and go right if the condition is false, and so does the knowledge tree implemented in this assignment.

```
else {
    System.out.print(x.label + " (y/n)? ");
    if (YorN())
```

```

        x = x.left;
    else
        x = x.right;
}

```

12. Finally to close the outer loop, at the end of each prompt from the user, the program will ask whether the user wants to continue or not. If yes, then restart from the first state, and if no, then the program will continue to output the final result of the knowledge tree. With the help of ShowBinaryClass's method called prettyPrint(), we can get a neat and beautiful result of the constructed knowledge tree.

### Explanation of ShowBinaryTree:

Note: I rework this part of ShowBinaryTree class since I want to improvise it so that the result of the tree is in top-to-bottom format instead.

1. For our initial step, The prettyPrint() method takes the root node of the binary tree as input. In this case, we want to declare the method in a static way.

```
public static void prettyPrint(BinaryTreeNode root)
```

2. Next, we want to calculate the height of the binary tree by recursively traversing the left and right subtrees and finding the maximum height between them by using getHeight() method. This method will return the height which represents the number of levels in the tree. The base case of this recursive method is when the next node we visit is null i.e. we reached the end of the node aka leaf node, aka no child left.

```
int treeHeight = getHeight(root);
```

```

public static int getHeight(BinaryTreeNode node) {
    if (node == null) {
        return 0;
    }

    return Math.max(getHeight(node.left), getHeight(node.right)) + 1;
}

```

3. Back to the previous method, prettyPrint() will then calculate the total width of the tree by using the formula  $\text{totalWidth} = 2^{\text{treeHeight}} - 1$ . The total width represents the maximum number of nodes in a level. Since we know that the knowledge tree is a binary tree, therefore the total width can be calculated with  $2^n - 1$ . Where n is the depth of the tree (height).

```
int totalWidth = (int) Math.pow(2, treeHeight) - 1;
```

4. Because we want to construct a tree that projected to the x and y axis, therefore we will require a 2D data structure to store our node. In this case, we choose a 2D array for the simplicity purpose of implementation. By creating a 2D array called treeArray

with dimensions `treeHeight` (rows) and `totalWidth` (columns). This array will be used to store the labels of the nodes in their corresponding positions.

```
String[][] treeArray = new String[treeHeight][totalWidth];
```

Note that the datatype is `String` since the value of each node contains the information from user input with `String` datatype.

5. In order to fill in the array with the user's input value, the `fillTreeArray()` method is called to populate the `treeArray`. It takes the `treeArray`, the root node, the current level, the start index, and the end index as parameters.

```
fillTreeArray(treeArray, root, 0, 0, totalWidth - 1);
```

```
public static void fillTreeArray(String[][] treeArray, BinaryTreeNode node,
int level, int start, int end) {
    if (node == null) {
        return;
    }

    int mid = (start + end) / 2;
    treeArray[level][mid] = node.label;

    fillTreeArray(treeArray, node.left, level + 1, start, mid - 1);
    fillTreeArray(treeArray, node.right, level + 1, mid + 1, end);
}
```

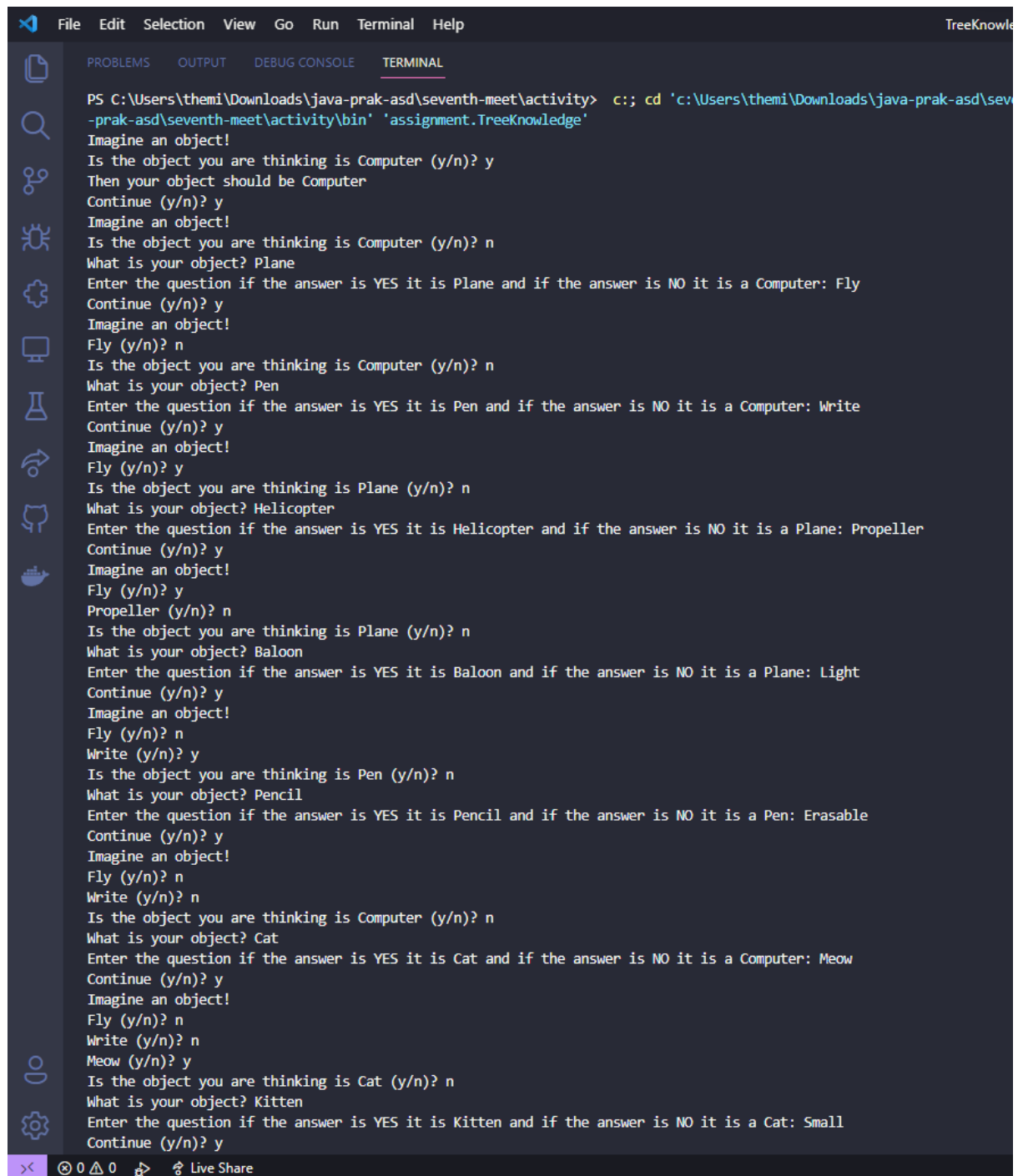
6. The `fillTreeArray()` method is defined above. The method has 5 parameters, namely the array that we want to store, the starting root node, the level of the tree, the start, and the end. This method works in a recursive way where the base case is if the current node is null, it returns and moves to the next iteration.
7. Then, we want to calculate the middle index by finding the average of the start and end indices. This index is used to represent the position of the current node in the current level of the `treeArray`.
8. Right after that, the label of the current node is placed in the `treeArray` at the previously calculated middle index for the current level.
9. For the recursive section, the `fillTreeArray()` is called recursively for the left child of the current node, incrementing the level by 1 and updating the indices to cover the left half of the current level.
10. On the other hand, `fillTreeArray()` is also called recursively for the right child of the current node, in a similar way, we increment it by 1 and then update the indices in order to cover the right-hand side of the level we currently have.
11. On the last part of the code snippet, once the `treeArray` is populated, the method iterates through each row of the array with respect to x-axis and y-axis and prints each cell. If the array contains null, it prints a space; otherwise, it prints the label of the node.



```
for (String[] row : treeArray) {  
    for (String cell : row) {  
        if (cell == null) {  
            System.out.print(" ");  
        } else {  
            System.out.print(cell);  
        }  
    }  
    System.out.println();  
}
```

12. Finally, the resulting binary tree is displayed on the console in a pretty and neat format.

## Program Output 1



```
PS C:\Users\themil\Downloads\java-prak-asd\seventh-meet\activity> c;; cd 'c:\Users\themil\Downloads\java-prak-asd\seventh-meet\activity\bin' 'assignment.TreeKnowledge'
Imagine an object!
Is the object you are thinking is Computer (y/n)? y
Then your object should be Computer
Continue (y/n)? y
Imagine an object!
Is the object you are thinking is Computer (y/n)? n
What is your object? Plane
Enter the question if the answer is YES it is Plane and if the answer is NO it is a Computer: Fly
Continue (y/n)? y
Imagine an object!
Fly (y/n)? n
Is the object you are thinking is Computer (y/n)? n
What is your object? Pen
Enter the question if the answer is YES it is Pen and if the answer is NO it is a Computer: Write
Continue (y/n)? y
Imagine an object!
Fly (y/n)? y
Is the object you are thinking is Plane (y/n)? n
What is your object? Helicopter
Enter the question if the answer is YES it is Helicopter and if the answer is NO it is a Plane: Propeller
Continue (y/n)? y
Imagine an object!
Fly (y/n)? y
Propeller (y/n)? n
Is the object you are thinking is Plane (y/n)? n
What is your object? Balloon
Enter the question if the answer is YES it is Balloon and if the answer is NO it is a Plane: Light
Continue (y/n)? y
Imagine an object!
Fly (y/n)? n
Write (y/n)? y
Is the object you are thinking is Pen (y/n)? n
What is your object? Pencil
Enter the question if the answer is YES it is Pencil and if the answer is NO it is a Pen: Erasable
Continue (y/n)? y
Imagine an object!
Fly (y/n)? n
Write (y/n)? n
Is the object you are thinking is Computer (y/n)? n
What is your object? Cat
Enter the question if the answer is YES it is Cat and if the answer is NO it is a Computer: Meow
Continue (y/n)? y
Imagine an object!
Fly (y/n)? n
Write (y/n)? n
Meow (y/n)? y
Is the object you are thinking is Cat (y/n)? n
What is your object? Kitten
Enter the question if the answer is YES it is Kitten and if the answer is NO it is a Cat: Small
Continue (y/n)? y
```

On the knowledge tree illustrated above, I input 11 new objects namely: Helicopter, Balloon, Bird, UFO, Plane, Pencil, Pen, Kitten, Cat, Keyboard, and Computer.

## Program Output 2 (Continued)

```

File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Is the object you are thinking is Pen (y/n)? n
What is your object? Pencil
Enter the question if the answer is YES it is Pencil and if the answer is NO it is a Pen: Erasable
Continue (y/n)? y
Imagine an object!
Fly (y/n)? n
Write (y/n)? n
Is the object you are thinking is Computer (y/n)? n
What is your object? Cat
Enter the question if the answer is YES it is Cat and if the answer is NO it is a Computer: Meow
Continue (y/n)? y
Imagine an object!
Fly (y/n)? n
Write (y/n)? n
Meow (y/n)? y
Is the object you are thinking is Cat (y/n)? n
What is your object? Kitten
Enter the question if the answer is YES it is Kitten and if the answer is NO it is a Cat: Small
Continue (y/n)? y
Imagine an object!
Fly (y/n)? y
Propeller (y/n)? n
Light (y/n)? n
Is the object you are thinking is Plane (y/n)? n
What is your object? Bird
Enter the question if the answer is YES it is Bird and if the answer is NO it is a Plane: Tweet
Continue (y/n)? y
Imagine an object!
Fly (y/n)? y
Propeller (y/n)? n
Light (y/n)? n
Tweet (y/n)? n
Is the object you are thinking is Plane (y/n)? n
What is your object? Ufo
Enter the question if the answer is YES it is Ufo and if the answer is NO it is a Plane: Alien
Continue (y/n)? y
Imagine an object!
Fly (y/n)? n
Write (y/n)? n
Meow (y/n)? n
Is the object you are thinking is Computer (y/n)? n
What is your object? Keyboard
Enter the question if the answer is YES it is Keyboard and if the answer is NO it is a Computer: Type
Continue (y/n)? n
Result of the knowledge tree:

          Fly
        /   \
  Propeller   Write
 /    \    /   \
Helicopter Baloon Tweet Pencil Pen Small Meow
          /   \    /   \
        Bird Alien Kitten Cat Keyboard Type
          /   \
        Ufo Plane Computer
  
```

For readability purposes, I have drawn the line using Paint to help visualize the tree. Note that traversing to the right = no, and traversing to the left = yes.

