Design a Guessing game where the computer tries to determine what type of animal the user is thinking of.

In Main.java, create a new instance of GuessingGame, passing in a root question and the two root answers.
`private DecisionTreeInterface<String>` tree is the reference variable within GuessingGame that will refer to the decision tree.
GuessingGame's constructor already creates a new instance of DecisionTree and sets 'tree' to refer to it.
In GuessingGame.play(), the computer should ask the root question. When the user answers 'yes' or 'no', call either tree.advanceToYes() or tree.advanceToNo() which should move the currentNode variable within the DecisionTree either left or right. Keep asking questions at each level until an answer node is reached (i.e. a leaf). Finally, guess the answer.

- If the user says 'yes', print out "I win"
- If the user says 'no', call the 'learn' method.

The learn() method should ask the user what animal they were thinking of. Then it should ask the user to give a question for which the answer is 'yes' for the animal the user was thinking of and 'no' otherwise. It should then properly set the data
In DecisionTree, the following methods need to be implemented:

```
   public void setCurrentData(String
newData)
   public void setResponses(String
responseForNo,
       String responseForYes)
   public boolean isAnswer()
   public void advanceToNo()
   public void advanceToYes()
```

Remember that the DecisionTree object uses `BinaryNode<String> currentNode` to move through the tree.
After each round, the computer should ask the user if they want to play again. When you play again, the tree should grow each time learn() is called. **DO NOT** stop running the program after each game as we are not backing up the data yet and your tree will reset each time.

With this type of project, you probably want to back your data up. BinaryTree.java currently implements getInorderIterator().

The following return null:

```
public Iterator<T> getPostorderIterator()
public Iterator<T> getLevelOrderIterator()

Implement getPostorderIterator and
getLevelOrderIterator so that they return
instances of two new private inner classes:
private class PostorderIterator
private class LevelOrderIterator
```

After running your game and building up the decision tree, allow the user to print the traversal through your tree using either a PostorderIterator or LevelOrderIterator object. We can use this data to back up our tree.

When the user finishes a game, ask them if they would like to save the tree. Iterate through your data with a LevelOrder Iterator object and save the data to a file. Then, when the user starts the program up again, prepopulate the decision tree with the data from the file.