Xiaofan Wu & Yuanzhen Pan
CS 230 Final Project Phase 2
11/30/15

Background

Linguistic research has shown that there are multiple benefits of learning language at a young age. Our LingoTree Game App aims at providing a fun way for kids to learn different languages.

1. User's Manual

In the main interface, we will have a welcome message and main buttons—About, start, and diagnose.

Once the user clicks the About button, we will proceed to another panel, explaining the objectives of this app and the instruction to use this app. Additionally, the user can choose to learn about who created the app.

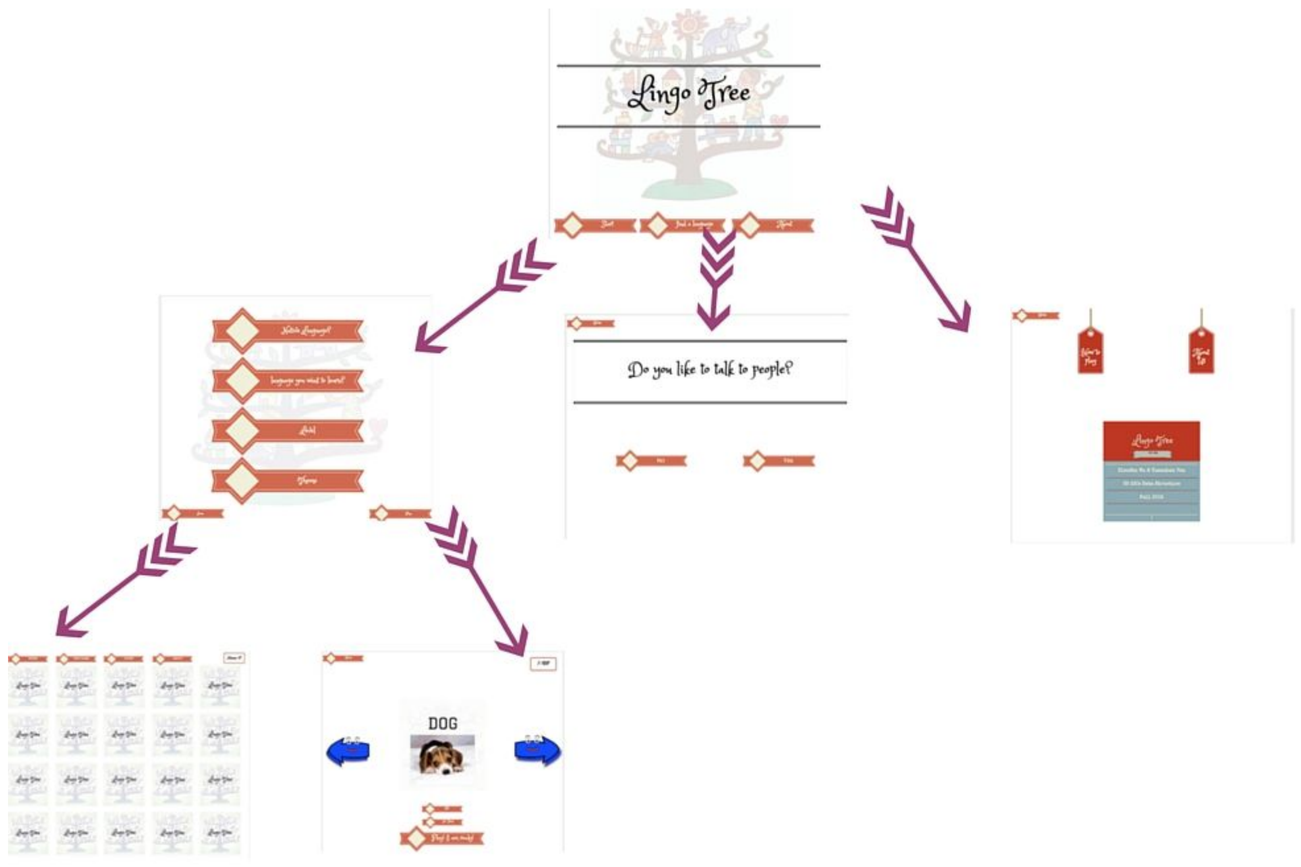
If the user clicks the Start button, our users will proceed to setting panel where they can choose their native language. (At the early stage, we are assuming that our users are native English speakers). The language they want to learn, level of difficulty and the theme of the vocabulary. In the same panel, there will be two buttons at the botton, Learn and Play. If the user clicks Learn, she will have proceed to the learn panel. Otherwise, Play panel.

Learn Panel: The users will be presented with a deck of vocabulary cards. Each card will have the vocab in their native language and then they can flip the card to see the vocab in the language they want to learn.

In the Play Panel, we will have our match card game. There will be a group of cards where users can click on to reveal the content—a word in the native in language or a word in the foreign language. The maximum number of cards that a user can click open is 2. If the meanings of the words clicked open in the 2 cards do not match, then the cards will be covered again. If they do, they will disappear. There will be button options for the user to go back to study the vocabulary first and return to About and the main menu.

In the diagnose panel, the user will answer some yes or no questions and then will get a recommended language that they should learn.

See the picture below for the general flow:



2. Technical report:

<u>ADTs</u>:

a) We will use a queue to store vocabulary. A queue is first in first out. We dequeue a card first when the user is studying it and then we enqueue the card back again.This is maintain the sequence of the cards in the queue so that when the user reaches the last card of the original queue, she is able to learn again.

b) Hashtable to store vocabulary for match card game. The key will be the user's native language, its value is the other language.

c) LinkedBinaryTree (Decision tree) to help the user to determine what language they should learn, the level of difficulty they should go for in this game. We decided to use LinkedBinaryTree because it makes the most sense when the answer is yes or no for every question.

<u>Classes will be used:</u>

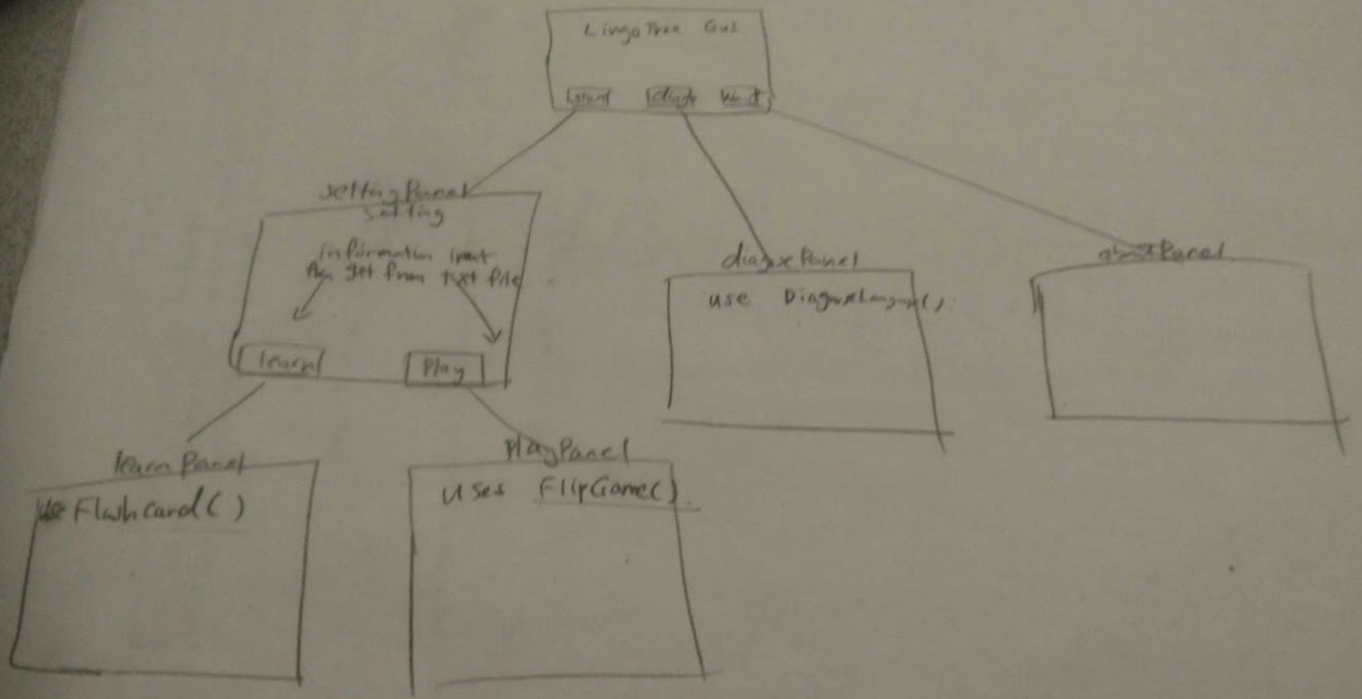
Panel Classes:
- LingoTreeGUI
- AboutPanel
- SettingPanel
- DiagnosePanel
- PlayPanel
- LearnPanel

Algorithm classes:
- DiagnoseLanguage: Program use to help user decide which language to learn
- FlashCard: Program to implement the flashcard method in the Learn Panel
- FlipGame: Main program to play the flip game. Implemented via hashtable.

Text files:

language flashcards will be in text files format and will be imported in. This way, we can have flexibilities in our program.

Overall Structure Picture:

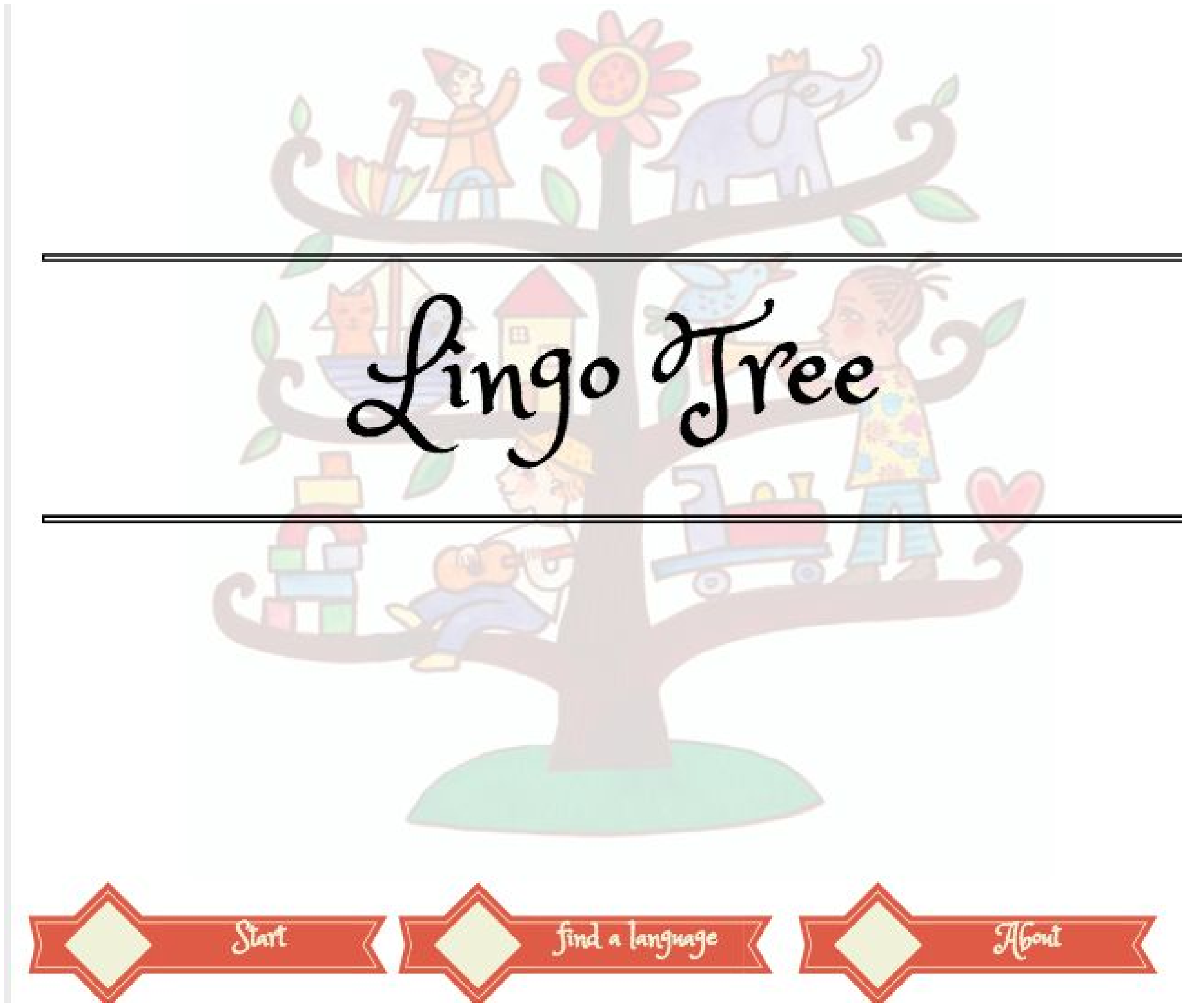

LingoTree Gui
Learn
Diag
[illegible]
settingPanel
Setting
information input
then get from text file
learn
Play
diagxPanel
use Diagx[illegible]()
[illegible]Panel
learnPanel
use Flashcard( )
PlayPanel
uses FlipGame()

<u>Class details:</u>

<u>GUI Classes:</u>

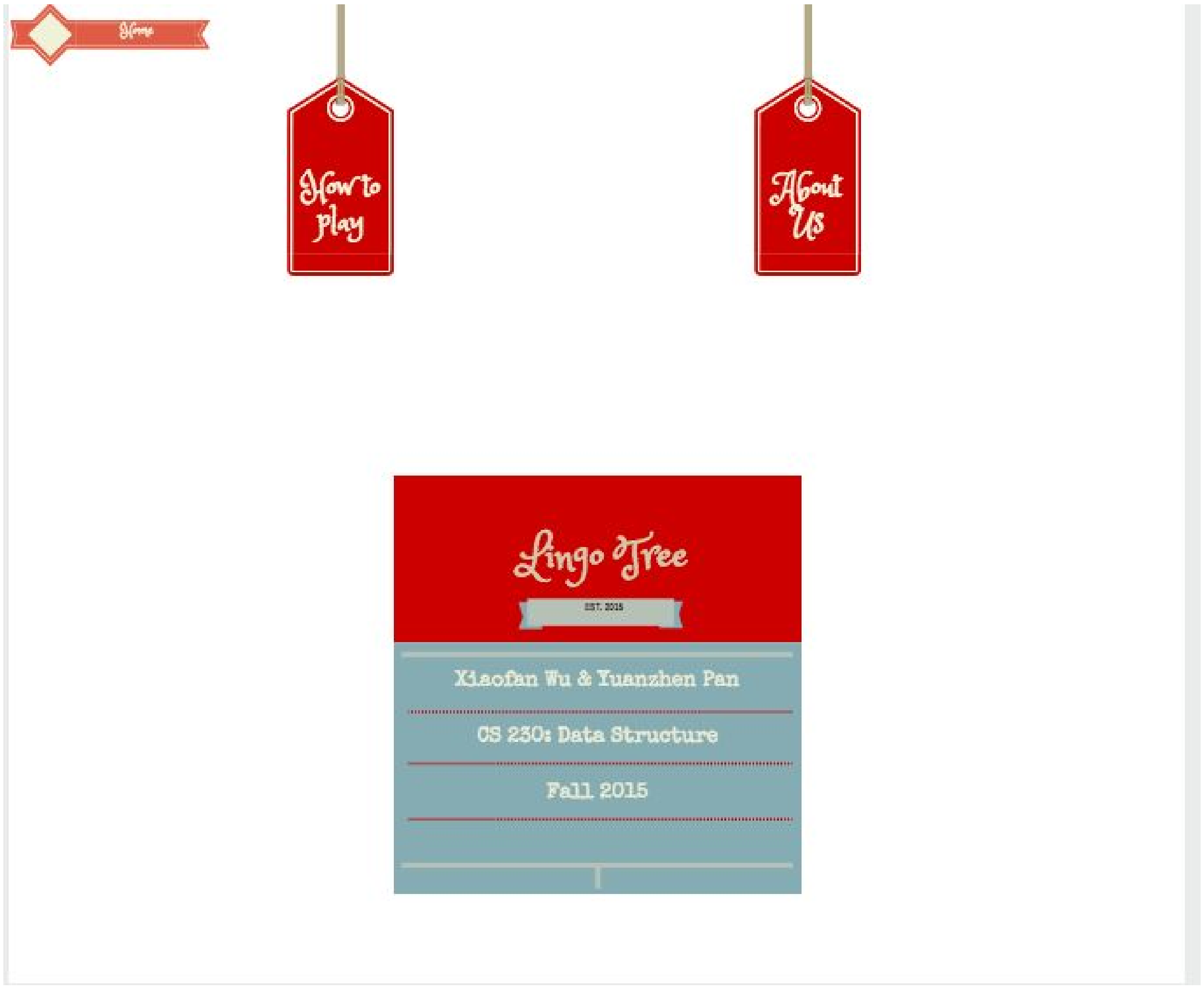
LingoTreeGUI

1. Main method that invokes different classes and create the interface
2. Create buttons that allow user to choose either play or about
    a. Will have button listeners to switch the frame to about or setting.
3. Import background picture
4. Link to three buttons (Start, Diagnose,About).



AboutPanel

1. Two buttons(About, HowToPlay).
2. Button Listenter: when the About button is clicked, text on the aim of this game and the creators of this game will be shown. When HowToPlay is clicked, instruction on this game will be shown at the same location.



Xiaofan Wu & Yuanzhen Pan

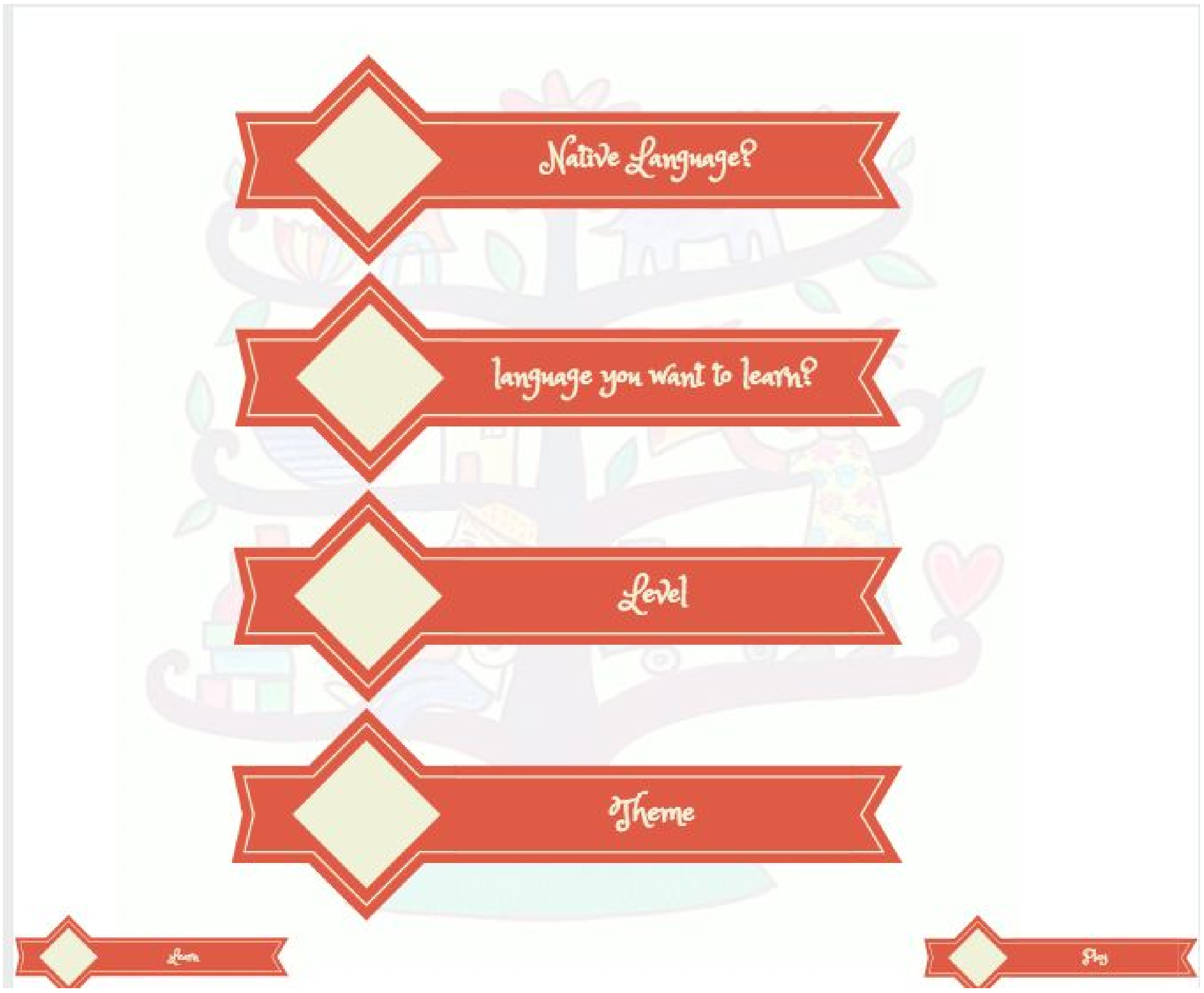
CS 230: Data Structure

Fall 2015

SettingPanel

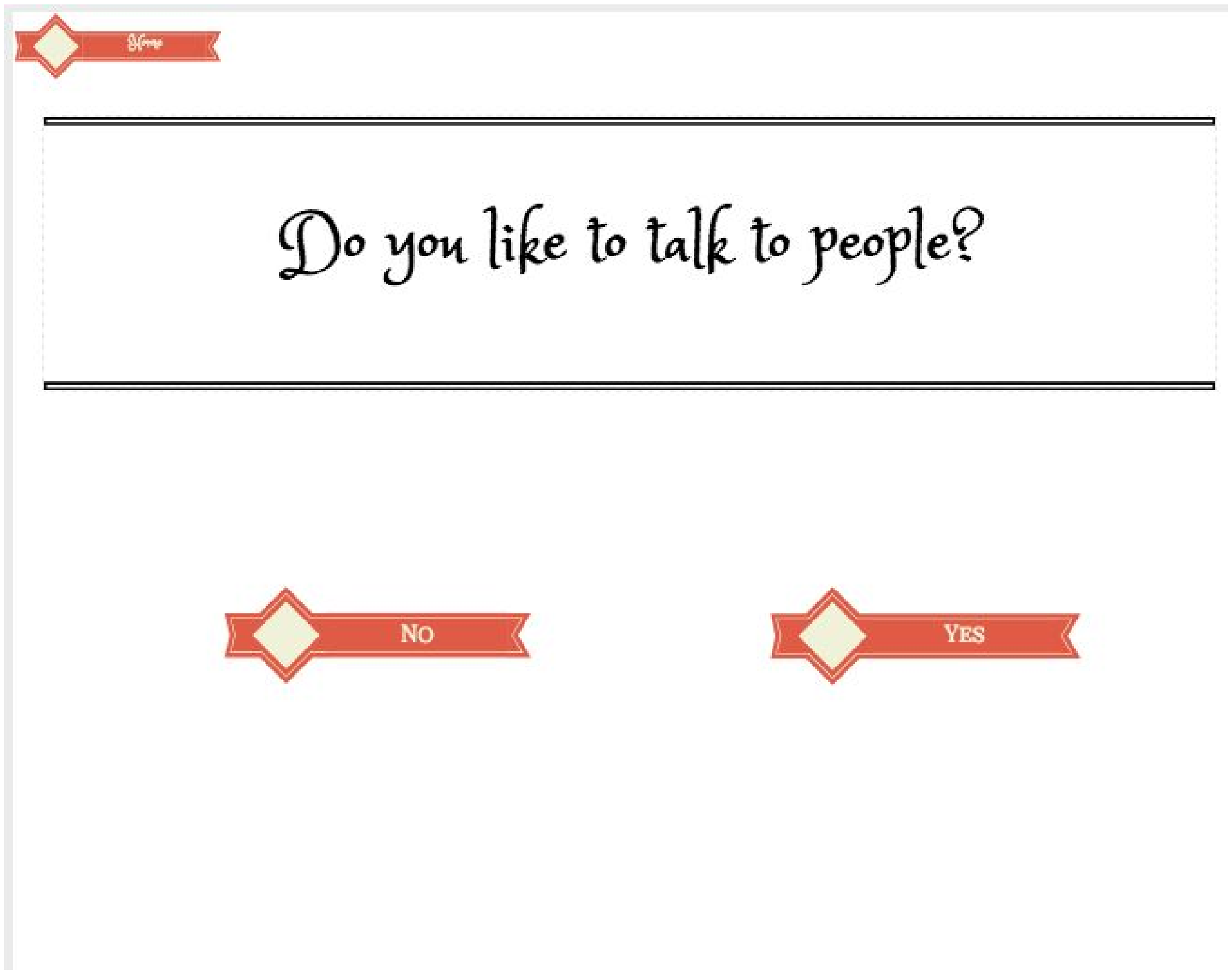
1. 4 dropdown menus
   a. native language(English, Hebrew, Chinese, Spanish, but for now we are assuming English as the native language),
   b. language you want to learn(English, Chinese, Hebrew, Spanish),
   c. level of difficulty(easy or hard)
   d. and theme(Numbers, Animals, Fruits).
2. Two buttons at the bottom to choose:
   a. Play: Will go to the PlayPanel to play the game. All the information inputted from the drop down menus will call different flashard files. This flashcard file will then be imported in FlashCard.java file.
   b. Learn: Will go to LearnPanel to learn the vocabulary. Similar to play, the flashcards will be created via importing stored text files.

3. We will set English as first language, Chinese as language you want to learn, Numbers as theme if the user does not choose. If the user clicks Play, invoke game class. If the user clicks Learn, invoke Learn class.



DiagnosePanel

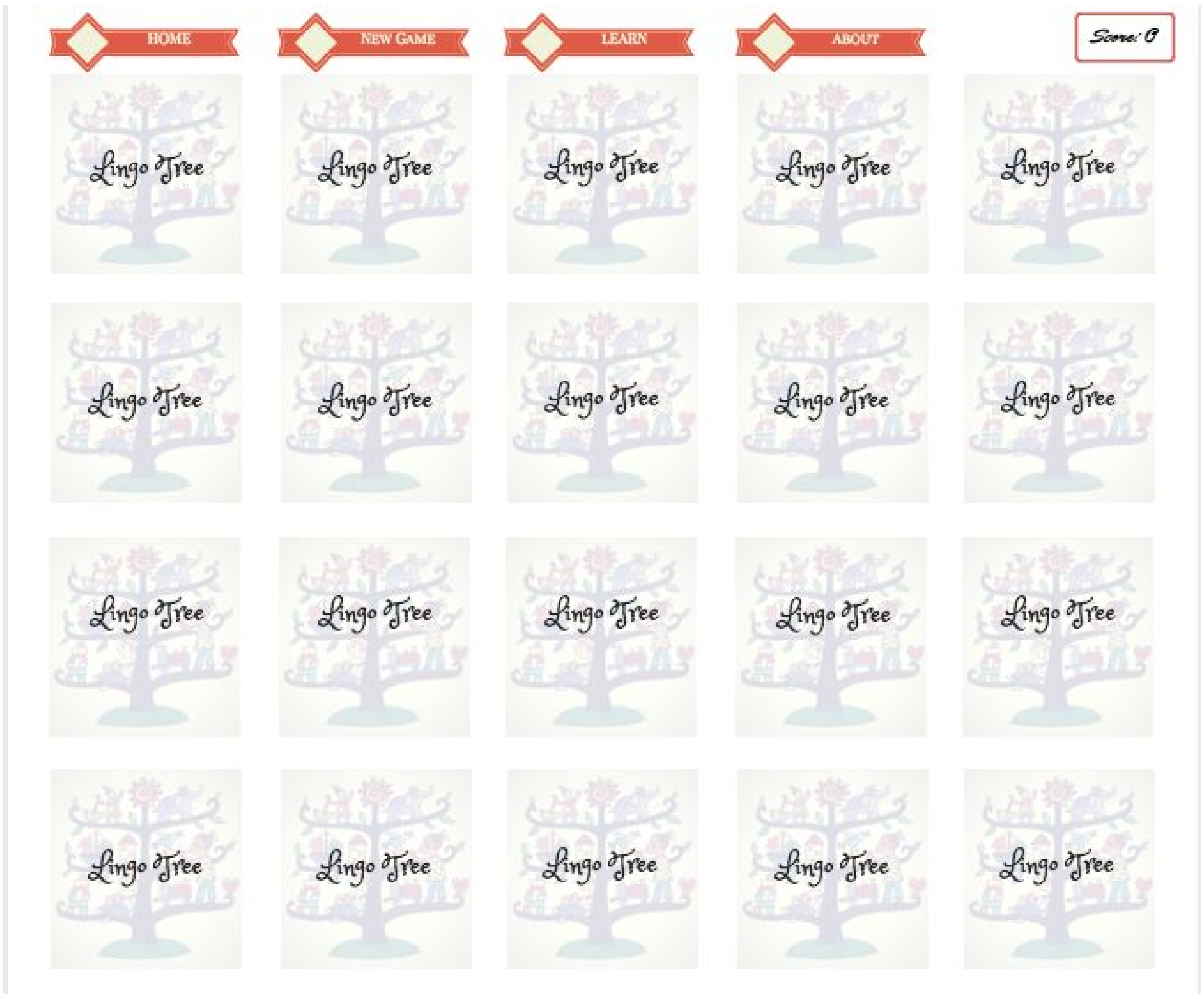
1. Uses DiagnoseLanguage class to implement the Decision tree.
2. Use Decision trees to help user to decide what languages they want to learn.
3. Will use buttons to select yes or no, instead of scanner.
4. will need to dynamically change the content of JPanel once the user clicks yes or no.



Do you like to talk to people?

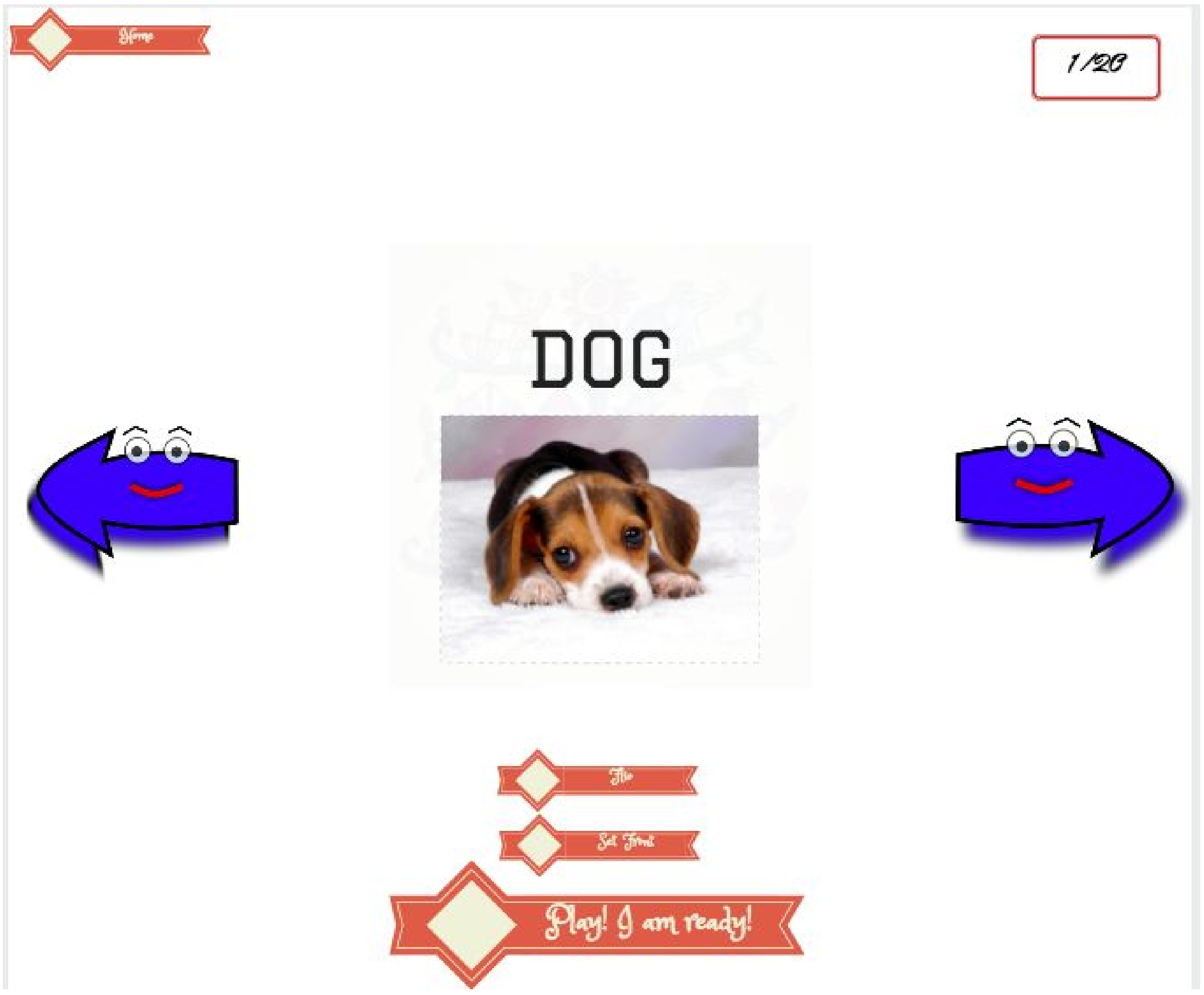
PlayPanel:

1. Uses FlipGame class to implement cards.
2. Create visual representations for each cards in the class.
3. Menu buttons at the top for user to return to the main pages.



LearnPanel:

1. Use FlashCard class to implement the flashcards.
2. Add buttons for next button, previous button, and menu buttons up top.



Other Classes:

FlipGame:

1. Use Hashtable to link two cards together
2. Key is the user's native language, its value is the other language.
3. When user clicks on two cards, the computer will check if the two cards match and then make the cards disappear.
4. This class will have the following methods
   a. Constructor will take in txt filename determined by user's input, a file name example will be german_english_easy_animal.txt. The components of the file name are determined in the setting Panel.
   b. check() method to check if user answered correctly.
   c. takeOut() method to take out cards if check() returns true.
   d. flipBack() to cover the 2 cards if they don't match
   e. score() to keep score

DiagnoseLanguage:

1. Use LinkedBinaryTree() to help user decide what languages to learn.
2. Import the questions from text files, so that it is easy to change later.
3. yesOrNo() method that listens to the user's selection and returns true if the Yes button is clicked, returns false for no.

FlashCard:

1. Use Queue to recycle cards once we are done with them to the back of the queue.
2. nextCard() method for the user to dequeue the first card and enqueue the card again to the queue. This is so that when the user reaches the last card of the original queue, she is able to learn again.
3. Implement shuffle method to shuffle the cards randomly.
4. Implement addToMemory method to keep another queue of cards that need to be remembered again if there is time.