Khalid Alkhulayfi
Kaa2@pdx.edu

Introduction:

In this project I have been asked to develop an application that help sighted and visually impaired to learn the braille language in an interesting way. The app should be made for android operating system. In addition, it should work with android talkback feature. This means that a game should be a good way to make the user entertain and learn the braille language in the same time. Therefore, I have decided to have hangman game that will make the user guess the word in order to save the human life. The application will also have a practice screen where the user can practice on using the virtual keyboard. In addition, the user can be tested by going to the test screen where the application will keep track of letter that the user is weak with, and ask him to enter that letter with other randomly chosen letters. All the files for this project are on GitHub (https://github.com/engkhalid/HangmanBraille.git)

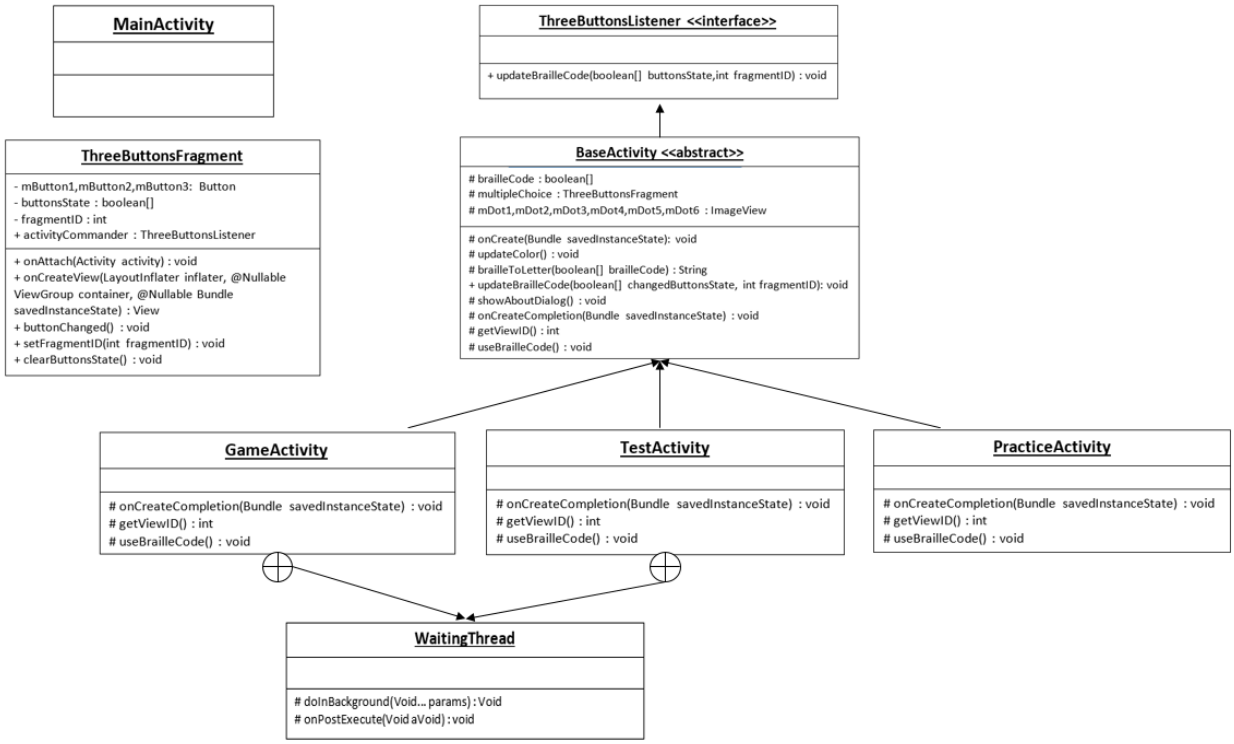
Software structure:

Since I decided to have a virtual keyboard for this application, it was necessary to think about the way that I should design the keyboard. My goal was to create a virtual keyboard that should be the same in all the activities and the user should not feel that he have to learn how to use more than one keyboard. Therefore, I decide to have six buttons (three buttons on each side) where the user can user his both hands to click the buttons. This means that I should have a block of code that I can use in every activity which leads me to fragments. I decided to have two separate block of three buttons. Thus, I designed the fragment with only three buttons and use two of it in every activity.

Another aspect of designing the software, was to think about hosting the virtual keyboard in every activity. Because I have the same layout of the virtual keyboard in every activity, I and every activity should create the fragment and implement the listener for them. In addition, having all the methods that deals with the input in every activity that have the virtual keyboard. I was important to think about a way to optimize that code and have an abstract class that has all code that can be used by more than one activity. The Game activity, Practice activity, and Test activity are sharing the same code in the Base activity which is the abstract class.

The way that the fragment gets it initial values is by using the get arguments. That are set by the hosting activity before the creation of the fragments in a bundle. On the other hand, the fragment uses an interface as a listener to communicate with the host activity. The following UML diagram shows how all the classes are related to each other. In addition, there is a JAVADoc file for this application that explains all the methods.

Khalid Alkhulayfi
Kaa2@pdx.edu



Base Activity:

In this activity are the code that is shared by the Game activity, Practice activity, and Test activity. This make fixing problems and changing the design a lot easier than going to each class and manipulating the code that is related to the fragment.

This class should take care of creating the fragments, setting its arguments, calling the completion of the creation code for every activity. In addition, it implements that ThreeButtonsListener interface that force this activity to have a method that take the input form the two fragments and distinguish each one of them. After that, it should use the changed code to update the color of the corresponding button. It also allows the subclasses to have a method that is called to use the new code.

Main Activity:

This activity is the first activity the user should see when he starts the application. It has three buttons that will take the user to Game activity, Practice activity, and Test activity. It also has its own menu option that shows the about dialog to the user. The following picture shows the activity layout.

Game Activity:

This activity has the hangman game. It uses the virtual keyboard which make it a subclass of the base activity. In addition to that is uses the space between the two fragments to show the picture of the victim, the mystery word, the used letters, and the entered letter.

Khalid Alkhulayfi
Kaa2@pdx.edu

The way that this activity words is that it shows the number of letters that the mystery word have. Then the user should start entering letters that he thinks it is part of the word. Every time the user enter a valid code a waiting thread is created to wait for the user to enter all the code for an amount of time. While the thread is running the UI thread will be free to respond to the user input. When the thread finish sleeping the UI thread will show a dialog that conform the current user input. This will give the user a fear amount of time to enter the full code of the letter he want to enter. After that, the chosen letter will be added to the used letters and check if it is one of the mystery word letters or not. If it is one of the mystery word letters, it will add it to the mystery word. However, if it is not one of the mystery word letters, it will change the picture of the victim. In both cases, it will show a message to the user.

When the user gets all the letters right it will show another dialog that will ask the user to go to the next question or quit the game. If the picture is the last one in the app it will show that the user fail to figure the word and it will show the same dialog.

When the orientation of the device is changed the app save all the needed variables in the onSaveInstanceState bundle. This will prevent the app from break down. In addition, it saves the flags that are raised when any dialog is shown to show it back when the application is recreated.

This activity has its own menu that include change delay and about dialog. It allows the user to change the delay time to respond to the user input.

Practice Activity:

This activity was created to allow the user to have a way to practice on entering the braille code. This means that the activity is going to use the virtual keyboard which means it has to be a subclass of the base activity that will take care of drawing the all the keyboard elements in its layout.

This activity uses the space between the two fragments to show the entered letter. It also, has its own menu that allows the user to see the about dialog.

Test Activity:

This activity was created to allow the user to test himself by showing a random letter and asking the user to enter its braille code. This means that the activity is going to use the virtual keyboard which means it has to be a subclass of the base activity that will take care of drawing the all the keyboard elements in its layout.

It uses the space between the two fragments to show report button, question number, the letter, and clear button. The user can click the report button to see all the letters scores that is saved in the application private files. On the other hand, the clear button can zero all the scores for all the letters.

Khalid Alkhulayfi
Kaa2@pdx.edu

The activity shows the letter that needs be entered by the user and wait until there is a valid data. When the valid code is entered it will use the same waiting thread mechanism that was used in the game activity. Then it will evaluate the answer and show the result to the used by a toast. After that it will go the next letter.

The smart thing about this activity is that it asks the user to enter the letter that has the minimum score often. This appearance of the letter with the minimum score can changed by changing the repetition number. This can be achieved by using the activity menu to open the dialog that will allow the user to choose how often does he wants to see that letter.

The activity uses the SharedPreferences to save the data. It loads the data when the activity is created and save it when the activity is paused. Saving the data when opening and closing the activity will make the application more efficient and save a lot of CPU cycles.

Three Buttons Fragment:

This fragment is creating part of the virtual keyboard that allows the user to enter the braille code. This means that any activity that wants to have a keyboard should have two of this fragment that are positioned next to each other.

The way that this fragment works is that it provide the user with three buttons. Therefore, having two of it will provide the user with six buttons that can be used to enter the braille code. As a result of that, the hosting fragment has to have a way to know which of the two fragments sent the recent update of the braille code. Thus, that can be achieved by having an ID for every fragment that was assigned to the fragment. Moreover, when the fragment uses the implemented function by the host from the three buttons listener, it will pass that fragment ID with the new update of the braille code.

The fragment is capable of have two different types of buttons. When the talkback is off the fragment uses the touch buttons that allow the user to touch and hold the button. On the other hand, when the talkback is on the fragment will get that information from the host activity and will create a click buttons that will keep the button hold when it is clicked.

Three Buttons Listener:

This interface class is used to force the hosting activity that implements it to have updateBrailleCode method. That will be used by the fragment to send the update of the three buttons it listen to. It provide the fragment with an easy way to get an instance of it that is casted to be a ThreeButtonsListener.

Waiting Thread Class:

This class is an inner class that is a subclass of the AsyncTask class. This means that this class has a doInBackground method that allow the UI thread to create a new thread that will do

Khalid Alkhulayfi
Kaa2@pdx.edu

something while the UI thread is responding to the user input. This make sure that we are not blocking the UI thread.

In this application I am using this class to make the application wait for the user to enter the whole code before the UI thread consider the input. When the waiting thread is created and executed it will sleep to some time (the default time is 0.5 second in Game activity and 1 second in the Test activity) before it calls the UI thread to call the next function.

Menu XML:

Since I am having different options for every activity, I had to create different menu for each one of the activities. This allows me to give the right options to the right activity. In addition, it make the application look more professional by providing a way that the user can do to change the app behavior.

When talkback is activated:

The application is working perfectly without the talkback and it responding to the user input by having the touch button that is going to be on as long as the user is holding them. However, when the talkback is on, the user has to navigate through the component on the screen and double click the button that he wants to use. This means that the user will not be able to touch and hold more than one button at a time. Therefore, I have used a click button that will make the user able to click the button and go to another one and click and so on.

To make the game work perfectly with the talkback the picture is provided with a description that will be read be the talkback with the user move his finger above it. In addition, the toast messages are change to be an alert messages that will force the talkback to read it and show it in sequence. Moreover, the user can change the delay time to be able to have enough time to enter the code.

Changing the delay time:

Using the virtual keyboard mean that different users will enter the braille code in different speeds. It will be great if the user is capable of changing the delay time. This option can be achieved by having a dialog window with a edit text that allow the user to enter the time that he think will take to input the braille code. This option is working great with the talkback because the user need more time to enter the code. However, the user who does not use the talkback can use less time to input the braille code.

Changing the repetition number:

In the test activity, this also can be achieved by having dialog that has a list of options of how frequent the application should show the letter with the minimum score. This feature make the user able to test them self with a random letter or with the letter that they are the most weak with.