The first change I made to the Java class that was provided to me was by adding two new get/set methods in the DecisionNode class; choiceOne and choiceTwo, both being String variables. I have done this so that it stores the text that is required for the buttons that will allow the user to make a choice to progress further in the story.

```
String choiceOne;
String choiceTwo;
```

```
public String getChoiceOne() { return choiceOne; }
public void setChoiceOne(String choiceOne) { this.choiceOne = choiceOne; }
public String getChoiceTwo() { return choiceTwo; }
public void setChoiceTwo(String choiceTwo) { this.choiceTwo = choiceTwo; }
```

The next change I did was with the DecisionMap method in the DecisionMap class. In the original code which was supplied, a Scanner class was used to read the .csv file however I changed it to a BufferedReader class. This is because the Scanner class was unable to find the .csv file even when I specified the file path and location. So, what I did to the original method was that I split up the method by putting the BufferedReader in the onCreate method which is located in the story_info.java class so this would read the file when the activity was loaded to the user. The InputStream would get the file from the correct location, in this case being res/raw/cw_data.csv location. Then the BufferedReader would be used to read the file from the specified location.

```
public DecisionMap() throws FileNotFoundException {
    Scanner inFile = connectDataSet( pathName: "src/cw_data.csv");
    buildUnorderedList(inFile);
    buildOrderedMap();
    // unorderedMap = null;
}
```

```
public DecisionMap(BufferedReader csvFile) throws IOException {
    buildUnorderedList(csvFile);
    buildOrderedMap();
    // unorderedMap = null;
}
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_story_info);

    InputStream inFile = getResources().openRawResource(R.raw.cw_data);
    BufferedReader csvReader = new BufferedReader(new InputStreamReader(inFile));
```

The next change I did was to the navigateMap method in the DecisionMapTest class from the supplied. I decided to get rid of the method and split up the code. I moved the node = perec.entryPoint() to the onCreate method in the story_info.java class, this was done to allow changes to the node variable to make sure that the correct node as used after the user makes their decision. By doing this, I then had to change the parameter for the method, which I renamed to getInfo(). I got rid of Utils parameter as I was no longer using that class and I changed the DecisionMap parameter to DecisionNode as I was no longer using the entire map, just the node required at the current time. Just like in the supplied code, I used an IF statement to make sure that that if, the final node was reached then, it would open the new activity designed just for the ending. The else statement in the original code was also changed as I had to adapt it for an Android app so therefore, I used onClickListener to check if the button has been pressed which would then get the correct node and call the getInfo method to show the correct information to the user.

```
public static void navigateMap(Utils u, DecisionMap perec){
    DecisionNode node = perec.entryPoint();

    while(node != null) {
        u.console(node.getDescription());
        u.console(node.getQuestion());

        if (node.getQuestion().equals("-")) {
            u.pressEnterToContinue();
            node = node.getYesNode();
        }
        else {
            int decision = u.fromConsoleGetInt( prompt: "press 1 for Option 1 or 2 for Option 2");
            switch (decision) {
                case 1:
                    node = node.getYesNode();
                    break;
                case 2:
                    node = node.getNoNode();
                    break;
            }
        }
    }
}
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_story_info);

    InputStream inFile = getResources().openRawResource(R.raw.cw_data);
    BufferedReader csvReader = new BufferedReader(new InputStreamReader(inFile));

    try {
        DecisionMap perec = new DecisionMap(csvReader);
        node = perec.entryPoint();
        getInfo(node);
```

```
public void getInfo(DecisionNode node) {
    if (node.getYesID() == -1 && node.getNoID() == -1){
        Intent buttonClick = new Intent( packageContext: story_info.this, ending_page.class);
        buttonClick.putExtra( name: "pass message", node.getDescription());
        startActivity(buttonClick);
    }
    else {
        TextView storyDesc = (TextView) findViewById(R.id.storyInfo);
        storyDesc.setText(node.getDescription());

        TextView storyQ = (TextView) findViewById(R.id.storyQuestion);
        storyQ.setText(node.getQuestion());

        Button optionOne = (Button) findViewById(R.id.firstChoiceButton);
        optionOne.setText(node.getChoiceOne());

        Button optionTwo = (Button) findViewById(R.id.secondChoiceButton);
        optionTwo.setText(node.getChoiceTwo());
    }
}
```

```
DecisionMap perec = new DecisionMap(csvReader);
node = perec.entryPoint();
getInfo(node);

final Button buttonOne = findViewById(R.id.firstChoiceButton);
buttonOne.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        node = node.getChoiceOneNode();
        getInfo(node);
    }
});

final Button buttonTwo = findViewById(R.id.secondChoiceButton);
buttonTwo.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        node = node.getChoiceTwoNode();
        getInfo(node);
    }
});
```

The next change I made was to the buildUnorderedList() method in the DecisionMap class. The original method used to use the Scanner class as one of the parameters to build the list however due to the change I made from using Scanner to BufferedReader. Each row is read and stored in the row variable. Then a node is built using the buildNode() and stored in the node variable, that is then appended. As this is in a while loop is the process is repeated until the dataset is exhausted.

```
public void buildUnorderedList(Scanner dataSet) {
    dataSet.useDelimiter(",");
    DecisionNode node ;

    do {
        String line = dataSet.nextLine();
        node = buildNode(line);
        append(node);
    } while (dataSet.hasNext());

    dataSet.close();
}
```

```
public void buildUnorderedList(BufferedReader dataSet) throws IOException {
    DecisionNode node;
    String row;

    while ((row = dataSet.readLine()) != null) {
        node = buildNode(row);
        append(node);
    }

    dataSet.close();
}
```

One of the other changes I made was removing the connectDataSet method from the supplied code as this would be used for the Scanner class however, I was no longer using the Scanner class therefore I would no longer have any use for it. I also removed the Utils class from the project as it was mainly designed for a console application whereas mine is designed as an Android application.

The next change that was made to the buildNode method in the DecisionMap class. I added two more lines of code for the choice description which would be used as the text on the button that would appear to the user when confronted with a choice.

```
private DecisionNode buildNode(String line) {
    String[] stringArray = line.split( regex: ",");
    DecisionNode n = new DecisionNode();

    n.setNodeID(valueOf(stringArray[0]));
    n.setYesID(valueOf(stringArray[1]));
    n.setNoID(valueOf(stringArray[2]));

    n.setDescription(stringArray[3]);
    n.setQuestion(stringArray[4]);

    return n;
}
```

```
private DecisionNode buildNode(String line) {
    String[] stringArray = line.split( regex: ",");
    DecisionNode n = new DecisionNode();

    n.setNodeID(valueOf(stringArray[0]));
    n.setYesID(valueOf(stringArray[1]));
    n.setNoID(valueOf(stringArray[2]));

    n.setDescription(stringArray[3]);
    n.setQuestion(stringArray[4]);
    n.setChoiceOne(stringArray[5]);
    n.setChoiceTwo(stringArray[6]);

    return n;
}
```

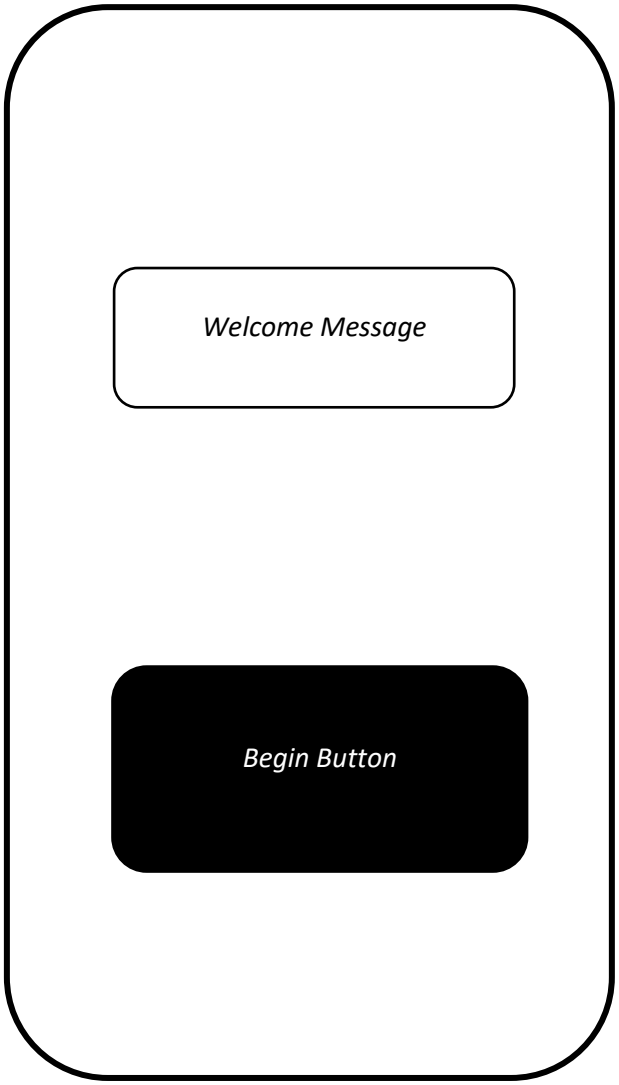
For an exception handler, I have used the try/catch with an IO exception in the onCreate method in story_info class.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_story_info);

    InputStream inFile = getResources().openRawResource(R.raw.cw_data);
    BufferedReader csvReader = new BufferedReader(new InputStreamReader(inFile));

    try {
        DecisionMap perec = new DecisionMap(csvReader);
        node = perec.entryPoint();
        getInfo(node);

        final Button buttonOne = findViewById(R.id.firstChoiceButton);
        buttonOne.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                node = node.getChoiceOneNode();
                getInfo(node);
            }
        });

        final Button buttonTwo = findViewById(R.id.secondChoiceButton);
        buttonTwo.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                node = node.getChoiceTwoNode();
                getInfo(node);
            }
        });

    } catch (IOException e) {
        exceptionToast(getApplicationContext(), e.getMessage());
    }
}
```

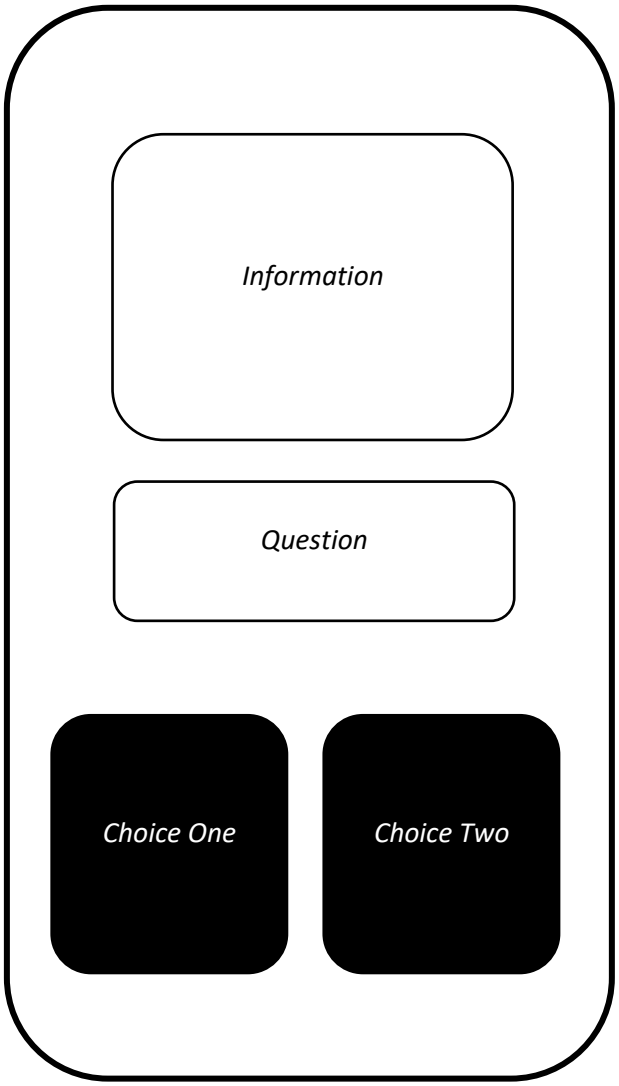
I also created an exceptionToast method which would show a pop up to the user if an error occurred. It would show the context of what was happening and the error message.

```
public static void exceptionToast(Context context, String message) {
    Toast.makeText(context, message, Toast.LENGTH_LONG).show();
}
```

Below shows how the app should look when it starts up. I have used a TextView for a welcome message to welcome the user to the app and a Button used to begin the game. which when pressed would send the user to the correct page in the app to allow them to begin playing the game.



Below shows the page where the game takes place and where the player makes all their decisions. I have two TextViews on the activity, one for the information of what is happening at this current stage of the game and one for the question to user of what their decision will be. Below that I have two buttons each corresponding to a choice; the button on the left corresponds to choice one and the button on the right corresponds to choice two. When the correct button is pressed then all the text would change to represent the current node. This would then repeat till an ending has been reached, then the user would be sent to another page.



This is the final page the user will see. Once the user has made a decision in the previous page (the one discussed in the page of this document), It would be determined by the program whether or not this is the final node. If it is the final node then the user would be sent to this page which would show the final piece of information to the user regarding the ending of the game. Below that I have put in a Button which when pressed, it would send the user back to the beginning of the app (the welcome page that was discussed before) from there the user can begin the game again and hopefully choose a different path and reach a new ending. The user can then repeat this cycle until they have found every ending to the game.