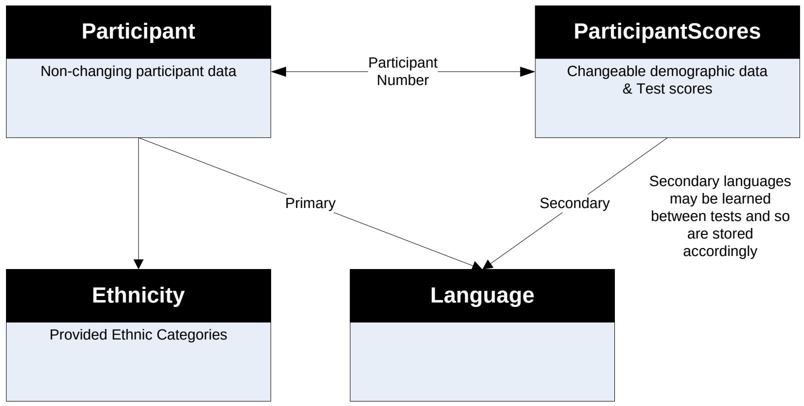
Span Task Test

By: Alexander Tkacs

Overview

This paper documents the development of a custom application that administers “Span Task” memory testing to be used by Dr. DeLuca of the Communication Disorders Department at Southern Connecticut State University. Span Tasks testing is comprised of both visual and audible components. The application records the necessary scores and demographic data needed, provides some on demand summary analyses, and provides for the export of the data to an xlsx file format that can be used by Microsoft Excel.

Implementation


Participant
Non-changing participant data
ParticipantScores
Changeable demographic data & Test scores
Participant Number
Primary
Secondary
Secondary languages may be learned between tests and so are stored accordingly
Ethnicity
Provided Ethnic Categories
Language

(Fig. 1) Database Design

The application’s database both supports certain application controls as well as storing the demographic data and test scores. The database file format is a Microsoft Access ACCDB file. As seen in (fig. 1), the database contains 4 tables. The *ParticipantScores* table has all the data that is not static and may change between administered tests even for the same participant. The *Participant* table has demographic data of the participant that does not change. The *FK_Ethnicity* field in *Participant*, and the *FK_PrimaryLanguage*, and *FK_SecondaryLanguage* in the *Participant* and *ParticipantScores* tables respectively are foreign keys to the *Ethnicity* and *Language* tables, which contain the IDs for their respective ethnicity or language in the table.

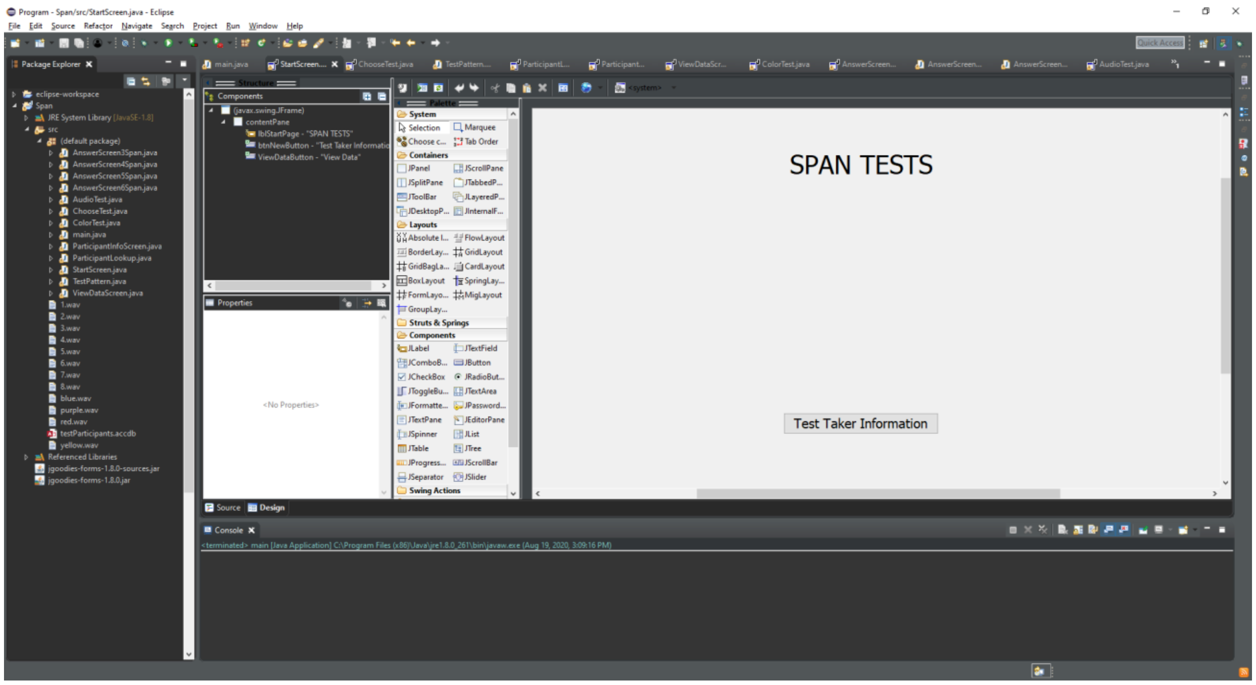
I used Structured Query Language (“SQL”) to pull information from the database. An example of this would be to get the PrimaryLanguage and Ethnicity of an existing user:

```
String query = "SELECT EthnicityValue, LanguageValue FROM Participant INNER JOIN Ethnicity ON FK_Ethnicity = Ethnicity.ID "
        + "INNER JOIN Language ON FK_PrimaryLanguage = Language.ID WHERE ParticipantNumber = "+partNum+";";
ResultSet rs = s.executeQuery(query);
```

(Fig. 2) implementation of SQL statement to get Ethnicity and Primary Language of participant based on the participant number.

To be able to connect to the ACCDB database the open source driver UCanAccess was implemented.

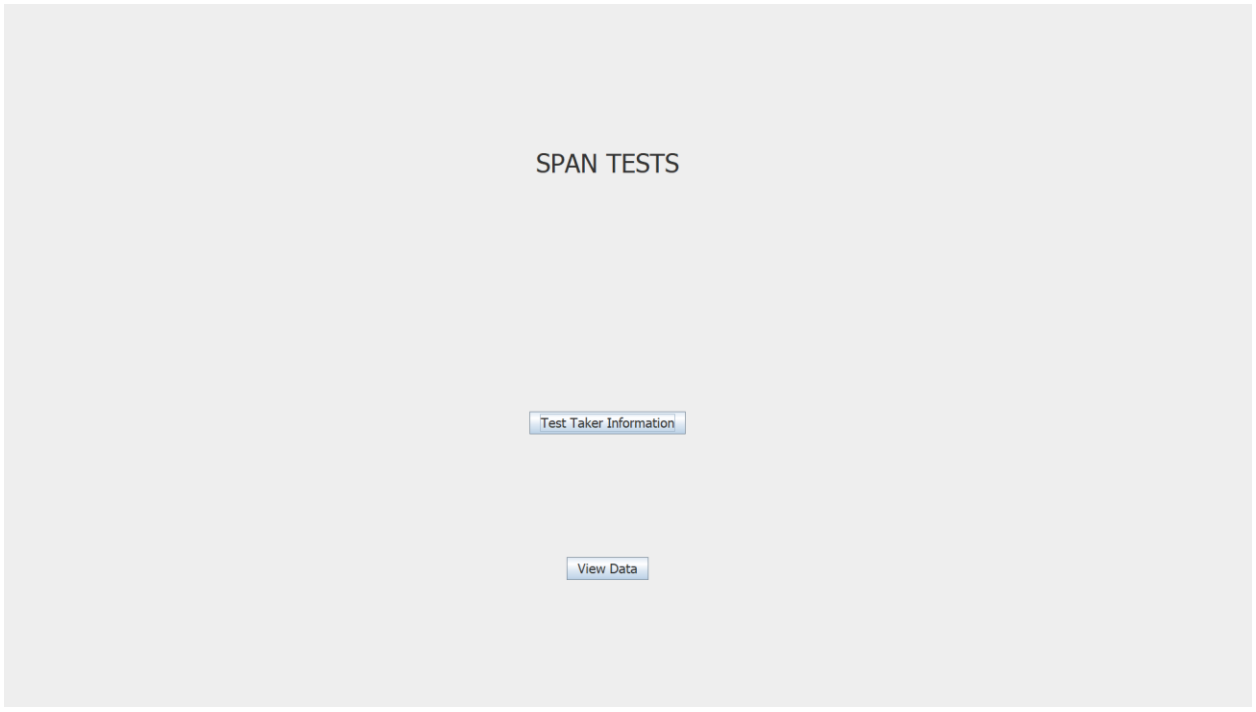
Java Window Builder was used to create the user interface windows.


Program - Span/src/StartScreen.java - Eclipse
Package Explorer
Components
(javax.swing.JFrame)
contentPane
lblStartPage - "SPAN TESTS"
btnNewButton - "Test Taker Informatio
ViewDataButton - "View Data"
Palette
SPAN TESTS
Test Taker Information
Source
Design
Console

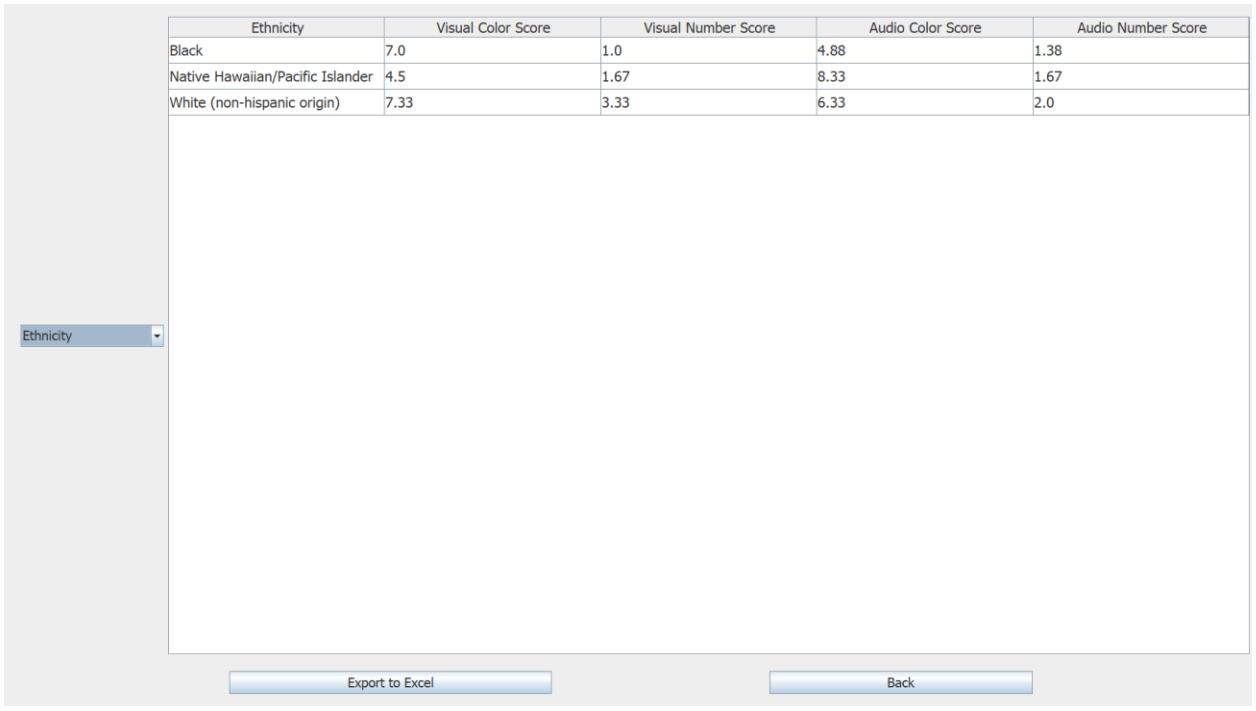
```
<terminated> main [Java Application] C:\Program Files (x86)\Java\jre1.8.0_261\bin\javaw.exe (Aug 19, 2020, 3:09:16 PM)
```

(Fig. 3) Example of Java WindowBuilder

Screenshots

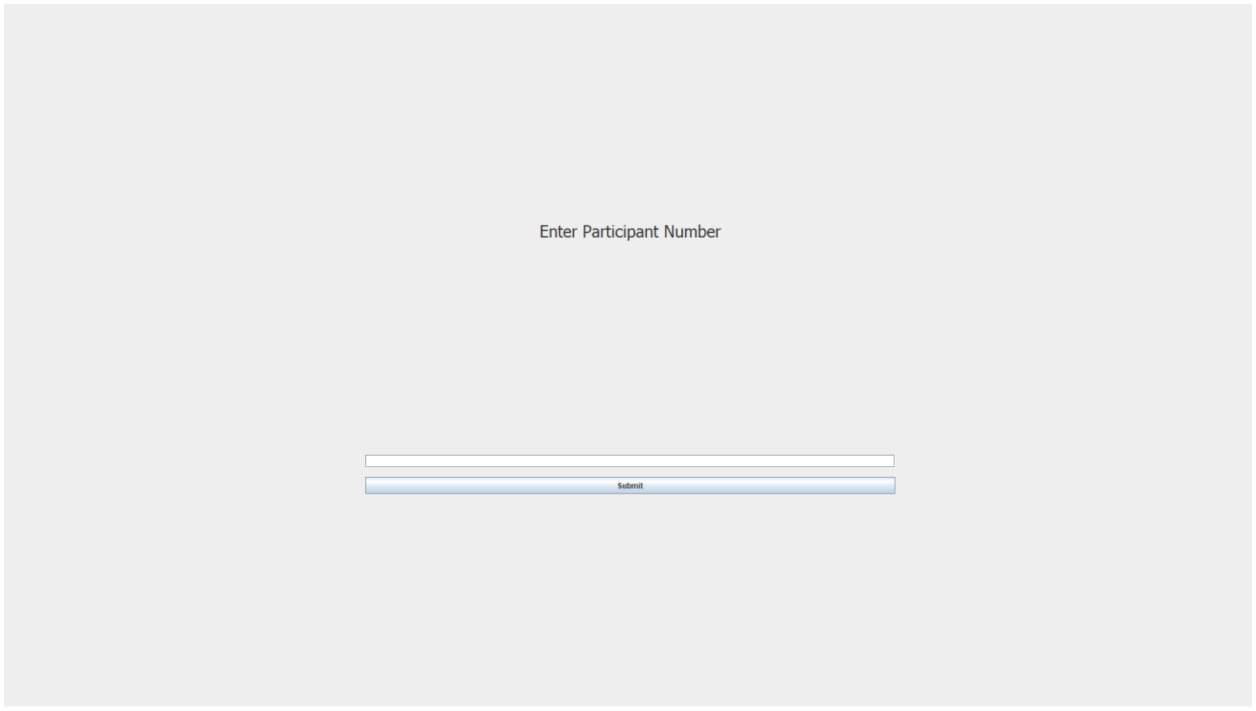


(Fig. 4) Main title screen

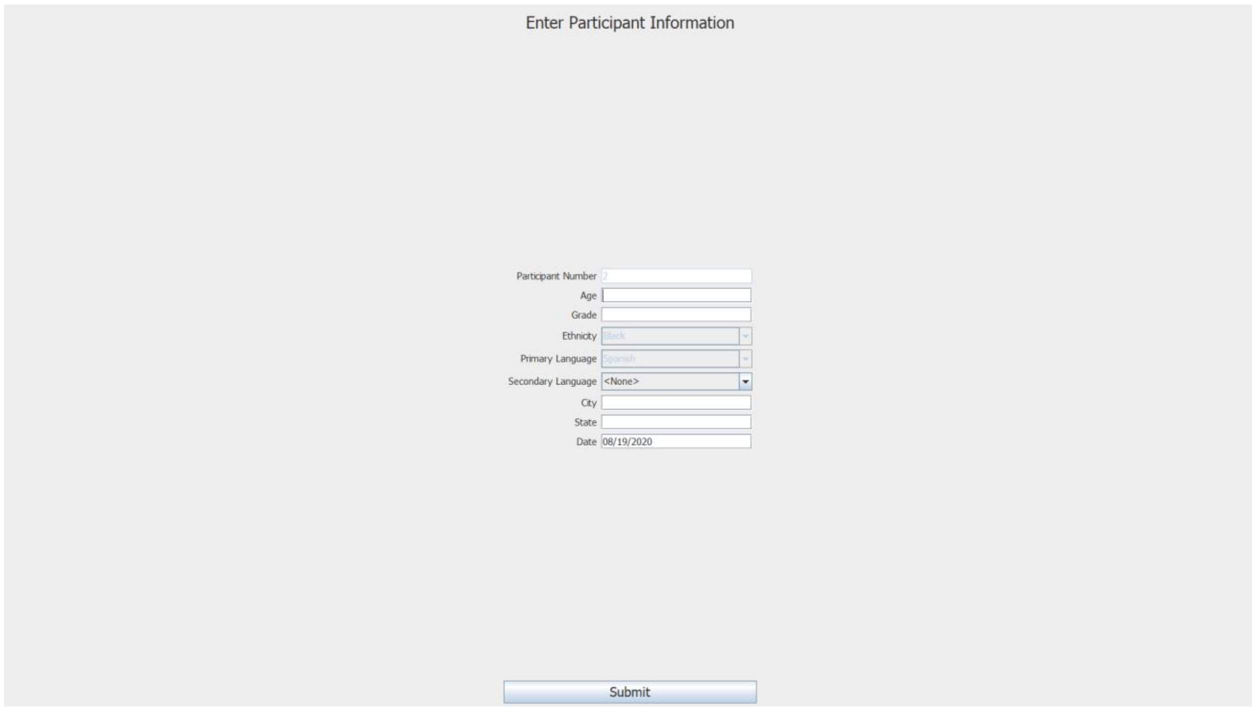


Ethnicity	Visual Color Score	Visual Number Score	Audio Color Score	Audio Number Score
Black	7.0	1.0	4.88	1.38
Native Hawaiian/Pacific Islander	4.5	1.67	8.33	1.67
White (non-hispanic origin)	7.33	3.33	6.33	2.0

(Fig. 5) On demand reporting screen showing the "Ethnicity" Statistics

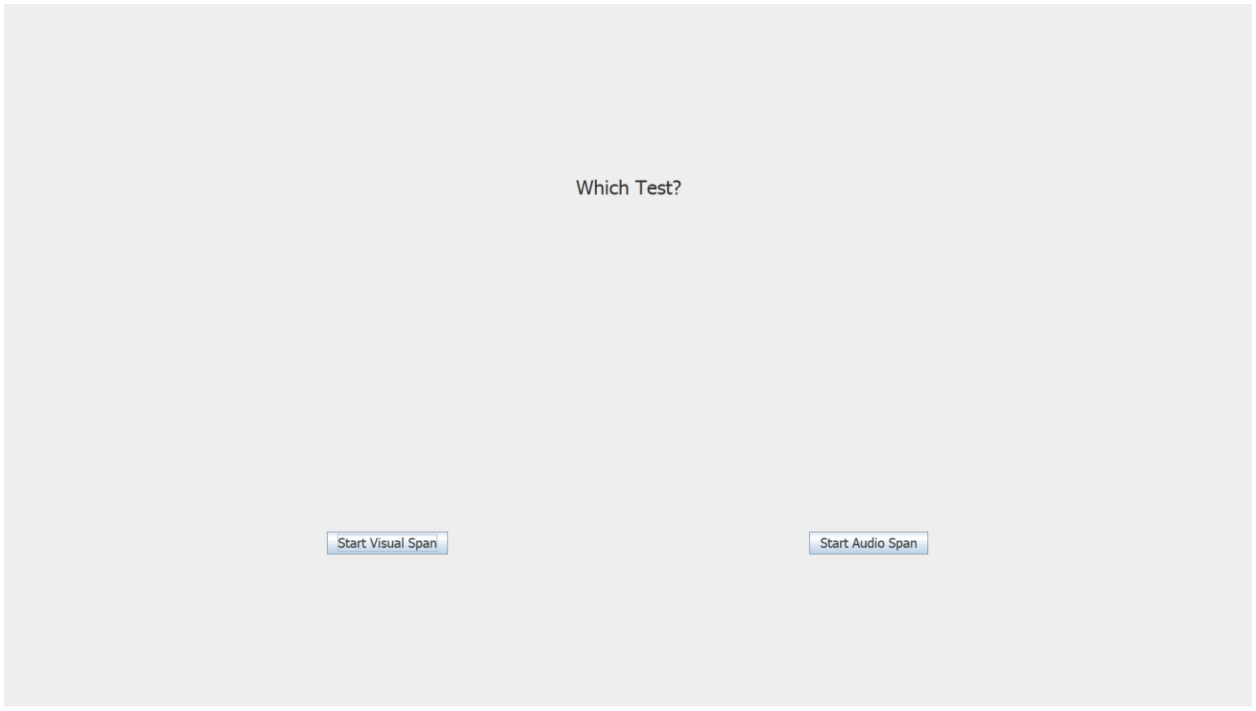


(Fig. 6) Participant lookup screen to either find an existing participant record, or if none are found, create a new one

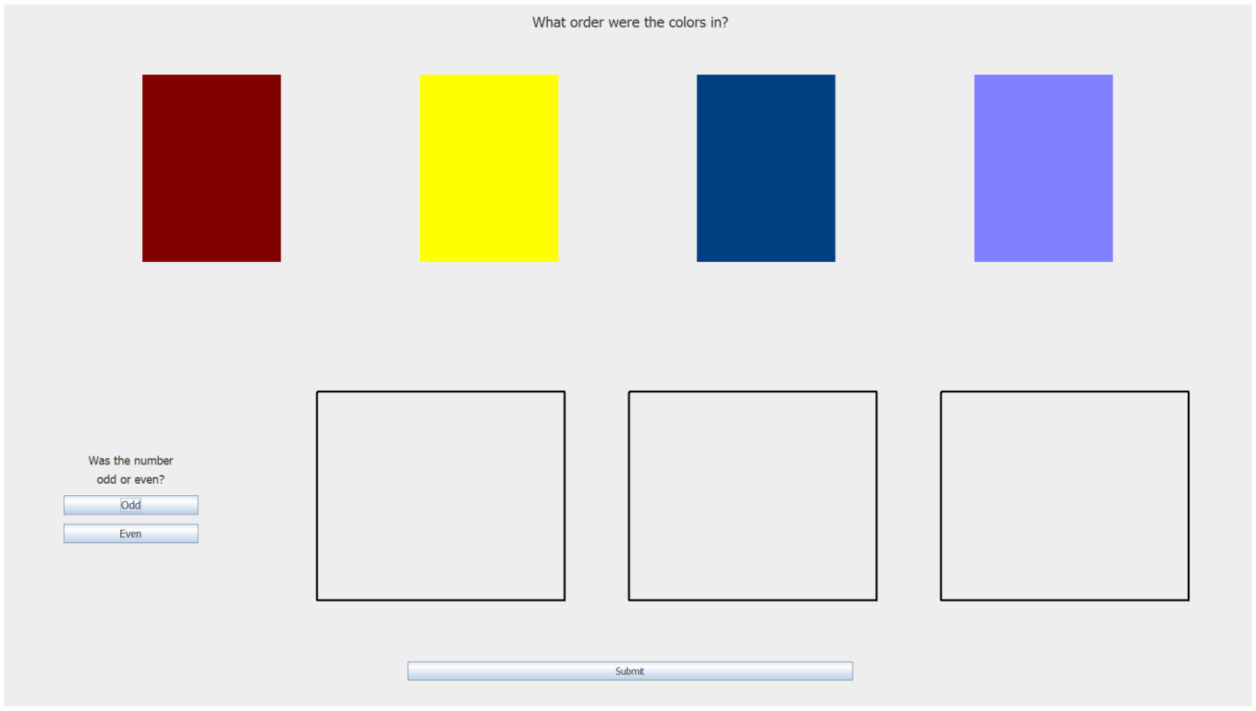

Enter Participant Information

Participant Number	2
Age	
Grade	
Ethnicity	Black
Primary Language	Spanish
Secondary Language	<None>
City	
State	
Date	08/19/2020

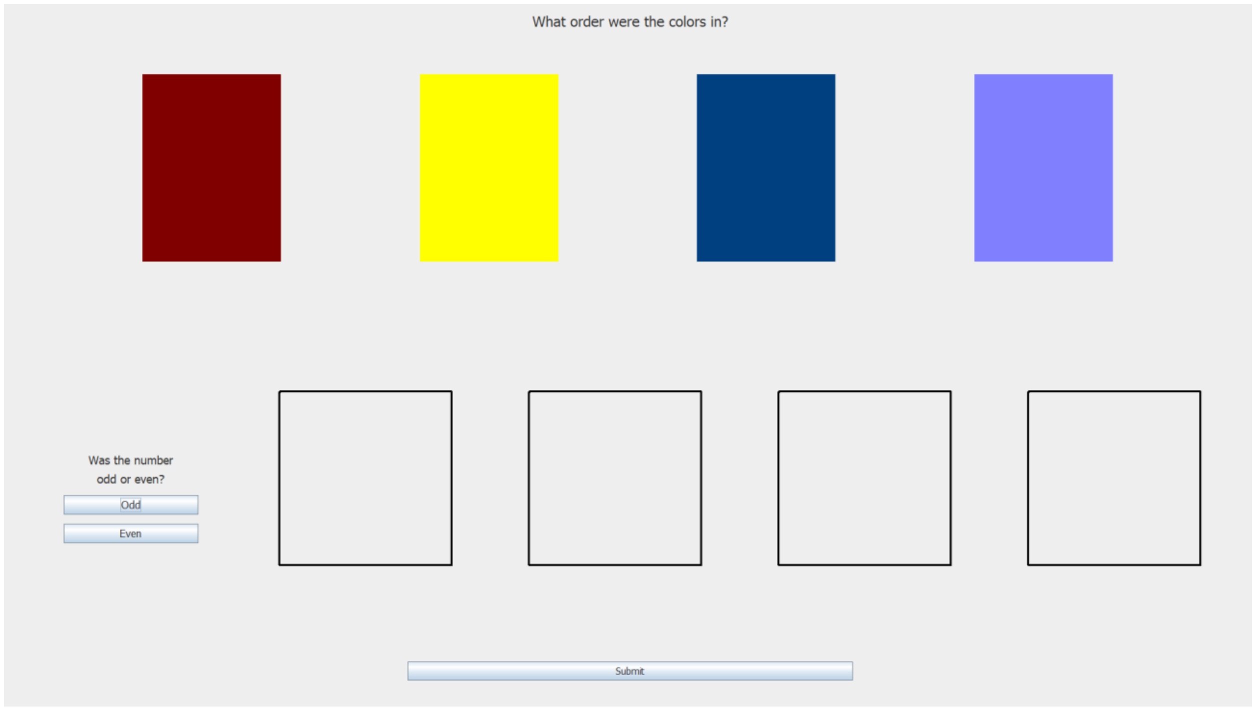
(Fig. 7) Participant demographic info screen



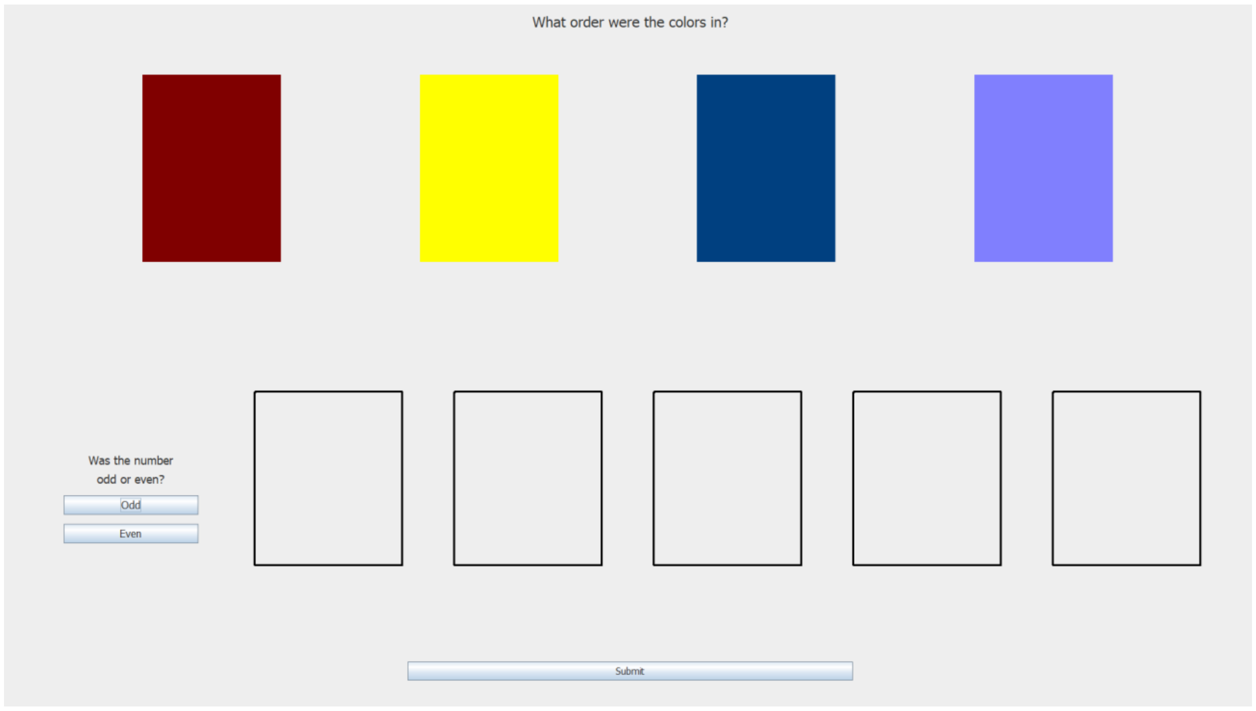
(Fig. 8) Test selection screen



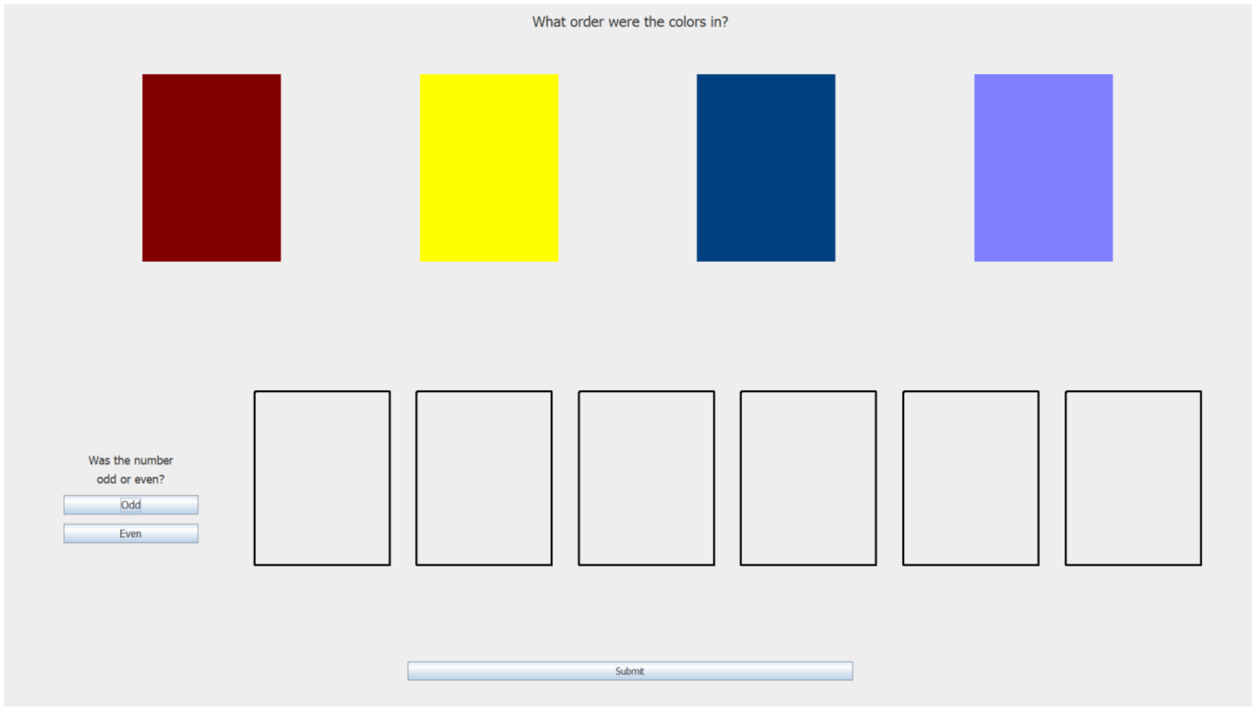
(Fig. 9) 3-span answer screen used to quiz the participant



(Fig. 10) 4-span answer screen



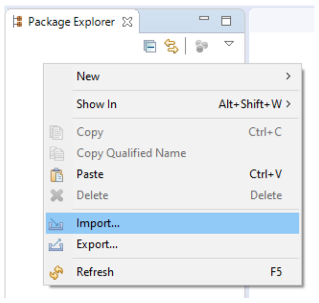
(Fig. 11) 5-span answer screen



(Fig. 12) 6-span answer screen

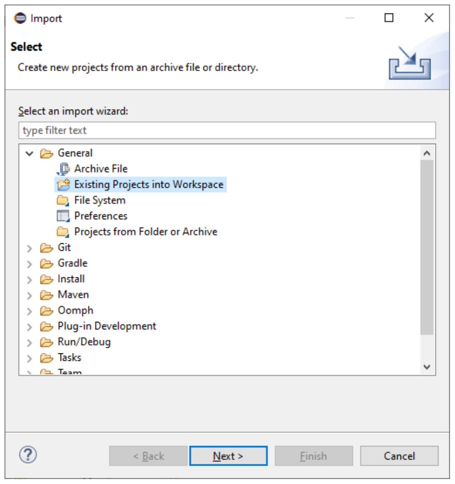
Installation

To install source code, Java's Eclipse Oxygen was used to create, test, and debug the code. To import the project into Eclipse, download the Span zip file from Github (https://github.com/AlexanderTkacs/SpanTask/blob/master/Span.zip). Extract the zip file to a folder. In Eclipse right click in package explorer, and select import.



(Fig. 13) Eclipse import

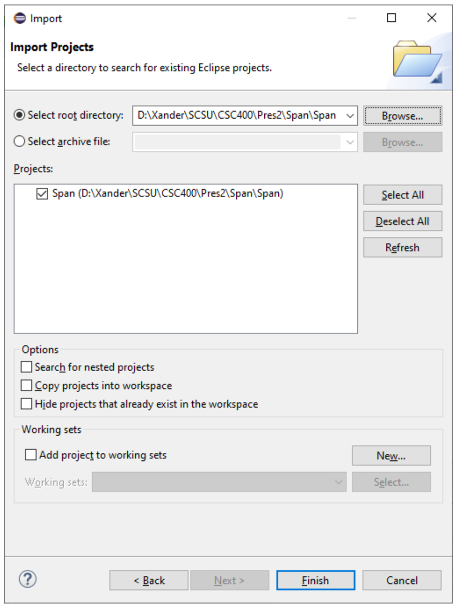
For the import selection, select "General" > "Existing Projects into Workspace"



Create new projects from an archive file or directory.

(Fig. 14) Eclipse Import Existing Projects into Workspace

Browse for the root directory. Select the Span folder that is inside of the Span folder, that is located where the zip file was extracted. Click *finish*.


Import

Import Projects

Select a directory to search for existing Eclipse projects.

(●) Select root directory: D:\Xander\SCSU\CSC400\Pres2\Span\Span

( ) Select archive file:

Projects:

[✓] Span (D:\Xander\SCSU\CSC400\Pres2\Span\Span)

Options

[ ] Search for nested projects

[ ] Copy projects into workspace

[ ] Hide projects that already exist in the workspace

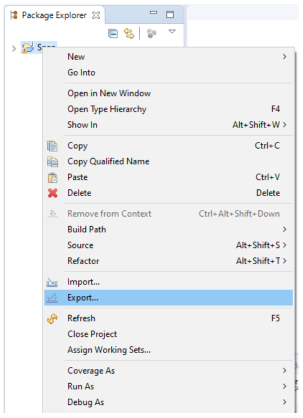
Working sets

[ ] Add project to working sets

Working sets:

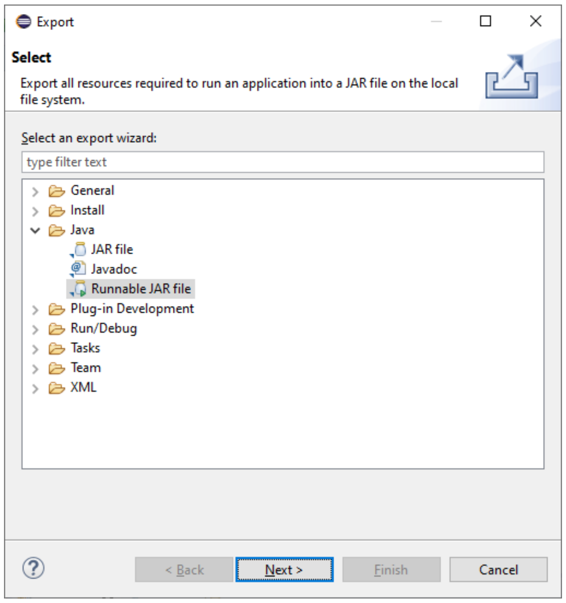
(Fig. 15) Eclipse import Projects

To export the Span project into a runnable *.jar* file, right click on the project, and select export.



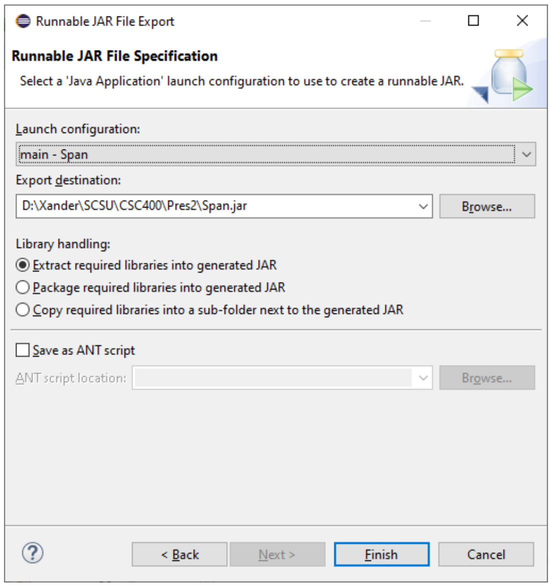
(Fig. 16) Eclipse export project

Select “Java” > “Runnable JAR file”.



(Fig. 17) Eclipse export as Runnable JAR file

Paste the file path for the folder to which it will be saved, and add “Span.jar” to the end. For Launch configuration select “main – Span”, and click finish, and the *jar* file should be there. Copy and paste the ACCDB file from the Span\scr\ folder containing the compiled *.jar*.


Runnable JAR File Export
Runnable JAR File Specification
Select a 'Java Application' launch configuration to use to create a runnable JAR.
Browse...
Browse...
< Back
Next >
Finish
Cancel

Launch configuration: main - Span
Export destination: D:\Xander\SCSU\CSC400\Pres2\Span.jar
Library handling:
(●) Extract required libraries into generated JAR
( ) Package required libraries into generated JAR
( ) Copy required libraries into a sub-folder next to the generated JAR
[ ] Save as ANT script
ANT script location:

(Fig. 18) Eclipse export project destination

As long as these two files are kept together, the application may be run from any location that has Java installed, including a USB flash drive, and does not require installation.

State of Implementation

The participant lookup is fully functional. If the participant number is found in the database, the resulting demographics screen will gray-out and lock the user out of changing static demographic data, such as primary language and ethnicity. If the participant number is not in the database, then the only thing that will be unchangeable will be the participant number at the top. After all the data is input and the submit button is pressed the application will save it to the database.

The visual test works as intended. It begins with the 3-span task, by showing three colors at random from a specified color palette for one second each, and then a number between 1 and 8 inclusively for four seconds before advancing to the test screen where the participant will have to recall the order of the colors, as well as if the number was even or odd. After doing that 3 times total the application will begin the 4-span task test where it will show four colors 1 second each, and then a number between 1 and 8 inclusively for four seconds before advancing to the test screen. After doing that 3 times it will start the 5-span. The application will show five colors for 1 second each and then a number between 1 and 8 inclusively for four seconds and then the test screen 3 times total. After the third answer screen the application will start the 6-span. It will show six colors for 1 second each, and then a number between 1 and 8 inclusively for four seconds 3 times total. After the participant presses the submit button on the final test, the application will check to see if the audio section has been done, and if it has it will return to the start screen. If the audio section has not been done then the application will return to the chose test screen, so that the audio section can be selected when ready.

The audio test fully works as intended. It will verbally say three colors and then say aloud a number between 1 and 8 inclusively before waiting four seconds, then advance to the test screen where the participant must try to recall the order of the colors, as well as if the number was even or odd. After doing that 3 times total the application will begin the 4-span task test where it will announce four colors and then a number between 1 and 8 inclusively before waiting four seconds and then advancing to the test screen. After doing that 3 times it will start the 5-span. The application will announce the names of five colors and then say a number between 1 and 8 inclusively, wait for four seconds and then go to the test screen, a total of 3 times. After the third test screen, the application will start the 6-span. It will say six colors and then a number between 1 and 8 inclusively, 3 times total. After the participant presses the submit button on the final test the application will check to see if the visual section has been completed, and if it has it will return to the start screen. If the visual section has not yet been completed then the application will return to the chose test screen, so that the visual section can be selected when ready.

The ‘answer’ screens of the test are fully implemented; the participant can drag a color from one of the 4 possible colors (red, blue, purple, and yellow) into one of the 3, 4, 5, or 6 answer boxes that are shown based on which span length test they are currently taking.

On the report screen; the ‘export database to Excel’ works fully. Currently only two of the on-demand aggregate demographic reports work showing averages. ‘Ethnicity’ shows the average visual color score, visual number score, audio color score, and audio number score based off each ethnicity that has a score in the database. The ‘age’ selection will show the average visual color score, visual number score, audio color score, and audio number score based off each age that has a score in the database.

Testing and Evaluation

To test the application, I recorded the colors and number with pencil and paper, and checked if the values in the database matched up with what I wrote down should be the values. The project does meet the objectives that I set out to do. It accurately records the answers that the user gives for the span tests, and accurately scores and records them. The application also accurately exports the four-table database to a single 'flattened' table excel spreadsheet. The application is usable at this point.

The next step of testing will be performed by the sponsor of this project, Dr. Deluca.

Lessons Learned and Reflection

One of the biggest lessons that I learned was that Java deprecates a lot of libraries and features. This was a problem because I wanted to use the language's built in functionality such as ODBC/ACCDB database access, only to discover that Java deprecated this functionality. The articles I had read explaining database access via JDBC/ODBC were written only a few years ago, before it was deprecated. After Java changed the open database driver, Microsoft stopped supporting the driver that they had created to connect to ACCDB files. I had to research alternatives and decided upon UCanAccess, an open-source driver that allows Java to connect to ACCDB files.

Another thing that I learned is that Java cannot interact with some files that are bundled in the *jar* file. For example, the *wav* and ACCDB files worked fine in testing in Eclipse, but after exporting to a single *jar*, they did not work. I was able to find a workaround for the *wav* files, but from my research Java cannot change files that are bundled in the *jar*, so the work around would not work for the ACCDB file, because it would be constantly updated. The best work around for the ACCDB file in the *jar* file would be to copy it outside of the *jar*, make changes, and then copy it back in, and delete the copy outside the *jar*. But by the time that I learned about that, I did not think that I had enough time to implement and test it before presenting the application.

The first thing that I learned was how to use Java's "Window Builder" so that I could quickly create windows that would be displayed to the user. This was more complicated than first thought, because there is little documentation on it. Information on how to move between different windows in Java is unexpectedly elusive.

Further work

Before I give this to Dr. Deluca for her to use, I intend to do a few more things. One of the most important things is user input validation. I want the application to not allow for bad data as much as possible. Currently a non-integer can be input as a grade or age, and the application will error because the database will only accept integers as inputs in those fields. I want it to not only not accept the value, but also give a reason why the application could not accept the currently selected input values.

I also want to make it so that it will output an error message if there is no database located in the same folder as the *jar* file. Right now, if it is missing the ACCDB file then it will not save anything, or give any feedback that it does not exist.

After the presentation Dr. Deluca asked me if I could add a gender field for more demographic data. Before she receives the application I want to have the gender field for the participant fully functional and part of the database.

For the reports screen, I want all the reports to function, instead of the two currently functioning (Age, Ethnicity). I also want to show more than just the averages, such as the max and min values in the database for the selected fields.

I also want to make it so that the application will not allow the user to select the same test twice for a participant to take in a row, without exiting the application first, because this will make it so that the data is consistent, and a participant cannot have double the possible maximum score for a test.

I further intend to add a button to the start screen to exit the application, because Dr. Deluca wanted the application to be ‘borderless,’ and have the close button on the top be removed so that children taking the test cannot interrupt it, but the administrator needs a way to exit the application from appropriate screens. The way that I have been exiting the application is by pressing alt+f4. I acknowledge that this not good, and I need a way for the application to close on its own.

Reflection

Some of the main challenges that I had to overcome were creating an Excel file in Java, changing screens, making the answer screen ‘click and drag’ for the colors, and the user interface in general in Java.

Java is relatively light on the number of things that can be done with the user interface. To make the windows I used Java WindowBuilder and found relatively few tutorials for it online. WindowBuilder allows for quick setup of a window in Java, because it is drag and drop, and allows for placement of things such as buttons and dropdown boxes.

For the answer screen to be ‘click and drag,’ I used a *mouseDown* event and checked the coordinates of the mouse and if it was in an area it would set the selected color, and a *mouseUp* event was used to check if the mouse was in one of the answer boxes, and had a selected color, and would then change it appropriately.

I chose Java because it is an object-oriented programing language, and can run anywhere as long as it is installed on the computer, which makes it very accessible. I had a class in high school that was in java, and one in college. During this project though, I learned how much I still needed to know to make a full application in Java. I watched many YouTube tutorials, and read some books so that I would be able to create application. In those books I learned about Java’s competing Swing and AWT libraries for user interface creation.

I also had to learn about SQL, as most of my experience with it was copy/pasting a SELECT query that was already provided in past assignment. One of the SQL books that I used was “SAMS Teach Yourself SQL in 10 Minutes.”

For many of the problems that I encountered I turned to the StackOverflow developers’ web forum. StackOverflow was where I originally learned about the UCanAccess database driver. StackOverflow also helped me understand how to save a date into an ACCDB file using an SQL statement. As well as how to create an Excel XLSX file and put information into it when trying to export the data from the ACCDB file as an excel spreadsheet.

Link to Github Repository:

https://github.com/AlexanderTkacs/SpanTask