**CBL project backlog and research and development process**

**– Ron Peer and Lavi Batzia – “mapPIN”**

Notes for readers:

* The entire development process and all relevant files can be found in this git repository, under the main branch:

<https://github.com/ronpeer/CBL_mapPIN.git>

* This project uses JDBC and Maven. for DB communication to work properly, you must execute the code with the mapPIN directory, being your current directory.
  + In the folder must be the database and pom.xml (used for dependencies).
* The file to execute to properly run the game is Main.java, which is in the mapPIN\src\main\java\mappin\game directory.  
  not to be confused with Main.class inside the mapPIN\target\main\java\mappin\game directory.
* Attached in the submission is also the file populate\_db\_queries.txt, containing all the commands used to populate the database. The DB in the repository is already populated, so there is no need to rerun them.

Advanced learning topics of choice:

1. Databases (Storage and access of data, both during and not during runtime).
2. Git (Version Control)

**Integration of learning concepts:**

**Databases:**

Using the resources listed in the git repository’s readme file, we learned about the usage and management of databases using java. The process of acquiring the necessary information was as follows:

To begin understanding databases, a few generic resources were gathered, elaborating on the basics of DB usage in java. Amongst these sources was also an interactive tutorial, which one can follow to create a basic program and learn to create, populate, edit and query an h2 database.

Once a basic program to test database functionality was created (which is not a part of this project, so will not be submitted), a prototype database was created with the specific tables necessary for the game, and all queries used to populate it were saved in the populate\_db\_queries.txt file.

Then, to properly store and access the information in the database, a deeper understanding of the datatypes used in databases was needed, so more specific research was done on that topic.

After the tables and information were fully constructed, the last step was learning how to use the Resultset type, the type java uses to store database query results.

Lastly, some more research was done on SQL command and query syntax, using w3schools as well as a fun online game.

After all necessary topics were covered (creation of databases and tables, connecting to a database (and allowing multiple connections at once), population of DB tables, editing and removal of entries, SQL querying, database types and management of Resultset objects), a class called DBmanager was written, to contain all database related actions, such as data retrieval and correlation (get city by id, get coordinate by name).

The database itself contains 2 tables:

Cities – 93 total entries, each with a numeric id, name of a city, its latitude and longitude, and its country.

Country\_colors – 93 total entries, each with a numeric id, name of a country, and its color in the backend map broken into RGB values.

**Version Control:**

Using the resource website along with the commands practice website, we learned basic and all necessary commands to be able to collaboratively work on the same project and track changes.

Using branching, we were able to simultaneously work on our tasks, while still updating the other, using push-pull commands to the origin, sharing a base set of code. During the learning process, we explored the previous commits we made due to changes and bugs. We merged into the main branch a few times when full tested code was finished. As part of the development, we frequently switched branches to learn, test, improve and integrate each other’s work.

Commands used:

* git commit (-a -m)
* git add
* git branch
* git status
* git push
* git fetch
* git pull
* git checkout (-f)
* git merge
* git stash
* git log (q to exit)
* git stash pop
* git reset

Backlog Items and their implementation (sorted by development significance):

**Name:** Data storage and access

**How to demo:** Create a database containing all game relevant data. City names, coordinates, country names and colors in the backend map. Then, using a java class, manage all queries and commands to the database. DB itself will be populated before game starts, and will be queried only while the game is active.

**Name:** Render game screen

**How to demo:** Load the 4 elements to the new window – the world map as an image, a timer, the name of the current player, and the name of the location to find. Can be resized when window resolution is changed.

**Name:** Mouse object (implemented in the map screen)

**How to demo:** During the game, a click on the screen will stop the timer and initiate score calculation with the coordinates of the clicking action.

**Name:** Start (mode selection) Screen

**How to demo:** Render a screen with 2 text areas for the players’ names and radio buttons representing the different game modes, as well as a start button. Until the start button is pressed, the players can freely edit the name and pick a mode.

Pressing the start button will transfer you to the main game screen according to the last values.

**Name**: Transfer Turns Between players

**How to demo**: Once a player has made their choice and the score has been calculated, it is shown on the screen and then after a pause a new game starting state is loaded for the next player in line. New player’s name will be reflected on the screen.

**Name:** Timer object (Action Listener)

**How to demo:** Tracks the time passed in turns and pauses. When triggered by a mouse click event, stops the countdown.

**Name:** Image analysis – country mode

**How to demo:** When a click event of the mouse is caught in this game mode, the coordinates of the mouse along with the time interval from the timer will be used to identify on which country the mouse has landed on and calculate the score. If it matches the clue, points will be awarded based on speed of response. If missed, no points will be awarded. The score will be sent to be displayed on the intermediatory screen.

This identification is done using a backend map, in which each country is colored differently. Relating the location chosen on the in-game screen to the relevant location on the backend map and using the color of the location in the backend map, the validity of a guess can be authenticated.

**Name:** Distance calculation – city mode

**How to demo:** When an event of the mouse is caught in this game mode, the coordinates of the mouse along with the time interval from the timer object will be used to calculate distance between the mouse and city of the round and calculate the score. Points will be awarded based on proximity to the clue and speed of response. The score will be sent to be displayed on the intermediatory screen.

Proximity will be calculated using a backend map, which has been made to reflect accurate real life distances, and can be directly correlated to the game map, using a mathematical calculation.

**Name:** Game over screen

**How to demo:** After all rounds are over the last screen loads automatically containing a overall scores for both players.

**Name:** player object

**How to demo:** At the start of the game, each player could choose their name. During each of the players turns in the game, no matter the mode, scores will be added to the list of scores of each player. At the end screen of the game, the overall score of each player will be displayed.