Computer Music

Assignment 2

Set Theory Calculator

1 Objective

In this assignment you will implement the backend logic (the music-theory part) for a modern web app that analyzes pitch class sets. The app should take a set of unique mod-twelve integers as input and return helpful insights such as *prime form*, *interval vector*, *transpositional symmetry*, *inversion*, *inversion symmetry*, *12-tone complement*, and *forte number*. We will follow conventions for these inspired by Larry Solomon and John Rahn.

The GUI is prebuilt for you. You are responsible for implementing the backend in either Python or Java, following the specifications of the provided interfaces and abstract classes. You may run your app locally in the browser or deploy it online as a static GitHub page for free.

2 Quick Start

1. Open a terminal at `./frontend/app`. On modern machines you can right-click this folder in the file-system GUI and choose open terminal at folder. Otherwise use `cd` in terminal to navigate there.

2. If you've never used (or never heard of) Node.js and npm before, you will need to install them first:

   - **On Linux:**

     ```
     sudo apt install nodejs
     ```

   - **On Mac:** If you've never used Homebrew, you need to install it:

     ```
     /bin/bash -c "$(curl -fsSL
         https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
     ```

     Then install the packages:

     ```
     brew install node
     ```

   - **On Windows:** Download and install at: `nodejs.org`

3. In your terminal at `./frontend/app`, run `npm install`. This installs dependencies needed for the project.

4. Boot the local server to see the app in the browser:

```
npm run start
```

5. Try analyzing a set. You'll notice the app can't fully function, because the backend isn't implemented. That's where you come in. Pick an option below, either Python or Java, and follow the directions.

3 Python Option

You will need to implement (1) `PrimeFormCalculator.py`, (2) `MoreStatsCalculator.py`, and (3) `CombinationsCalculator.py`. Each of these modules should contain only one class, which extends one of the following respective abstract classes (1) `PrimeForm.py`, (2) `MoreStats.py`, or (3) `Combinations.py`. These abstract classes are found in `./backend-python-project`.

> When picking a class to implement, start with `PrimeFormCalculator.py` and move in the order they are listed above. This will allow you to test the results, integrating the module you've written with the GUI as soon as you're finished.

For instructions and specs, refer to the respective abstract class for the module you're working on. For example, refer to `PrimeForm.py` while you're writing `PrimeFormCalculator.py`. I recommend developing the module within the `./backend-python-project` directory. To get started, extend the abstract class:

```python
from PrimeForm import PrimeForm

class PrimeFormCalculator(PrimeForm):
    # Your implementation here
```

Once you've finished a module, copy it into `./frontend/app/public/pyModules`. Do a hard refresh of your browser (`Ctrl + Shift + R` or `Cmd + Shift + R`). Try analyzing a set. If there are no errors in your backend code, you should see the functionality you've added appear in the GUI.

4 Java Option

4.1 Install Maven

Maven is the build tool that compiles your Java code to JavaScript (via TeaVM).

- **Linux (Debian/Ubuntu):**

```
sudo apt install maven
```

- **Mac (with Homebrew):**

```
brew install maven
```

- **Windows:** Download and install from Maven downloads. Add `bin/mvn` to your PATH.

To check installation:

```
mvn -v
```

This should print your Maven version.

4.2 Navigate to the backend project

Open a terminal in the `./backend-java-project` folder:

```
cd backend-java-project
```

4.3 Implement the classes in order

The abstract behavior is defined in three interfaces in `src/main/java/war`:

- `PrimeForm.java`
- `MoreStats.java`
- `Combinations.java`

For each one, create a corresponding implementation class in the same folder with `Calculator` at the end of the name:

- `PrimeFormCalculator.java` implements `PrimeForm.java`
- `MoreStatsCalculator.java` implements `MoreStats.java`
- `CombinationsCalculator.java` implements `Combinations.java`

For example, start with `PrimeFormCalculator`:

```java
package war;

import java.util.ArrayList;

public class PrimeFormCalculator implements PrimeForm {
    @Override
    public ArrayList<Integer> transToZero(ArrayList<Integer> set) {
        // TODO: implement this
        return null;
    }
    // Etc...
}
```

Work in order:

1. `PrimeFormCalculator` → test it in the app.
2. `MoreStatsCalculator` → test the new functionality.
3. `CombinationsCalculator` → finish by handling combinations and Forte numbers.

4.4 Compile and package

Once you've implemented a class, build the backend with Maven:

```
mvn clean package
```

This does three things:

1. Compiles your Java source.
2. Runs TeaVM to transpile it into JavaScript.
3. Outputs a `classes.js` file in: `frontend/app/public/classes.js`

The frontend automatically loads `classes.js` to handle backend logic.

5 Deploying App

To stop the local server, go to the terminal where you booted it and press `Ctrl + C`.

If you want to share your app with all the world, simply build the deployment and publish it as a GitHub Pages page for free:

1. In a terminal at `./frontend/app` build the app:

   ```
   npm run build
   ```

2. Put the contents of the resulting `./frontend/app/build` folder into a new GitHub repository.
3. Follow GitHub's directions to publish the repo as a website:
   `https://docs.github.com/en/pages/quickstart`
4. Visit the site at `your-username.github.com/repo-name`