Scuola universitaria professionale della Svizzera italiana
Dipartimento tecnologie innovative

SUPSI

Connect Four

Studente/i

Mattia Cainarca
Luca Fantò
Antonio Marroffino

Corso di laurea	Modulo / Codice Progetto	Anno
Bachelor Ingegneria Informatica	Software Engineering and Development I	2023 - 2024

Committente	Data
Giancarlo Corti	04/06/2024

STUDENTSUPSI

Index

Context and Motivation

Project goals

The project focuses on **improving** the **skills** and **soft skills** involved in developing a software project in teams:

- Configuration management (source code versioning, product versioning, software dependency, build and distribution of the final software product)
- Requirements elicitation and management
- Software design and development
- Internationalization
- User interaction

Requirements Elicitation and Analysis

User requirements

Presents a **Connect Four** grid **interface**.

Displays contextual **feedback to** guide **players**.

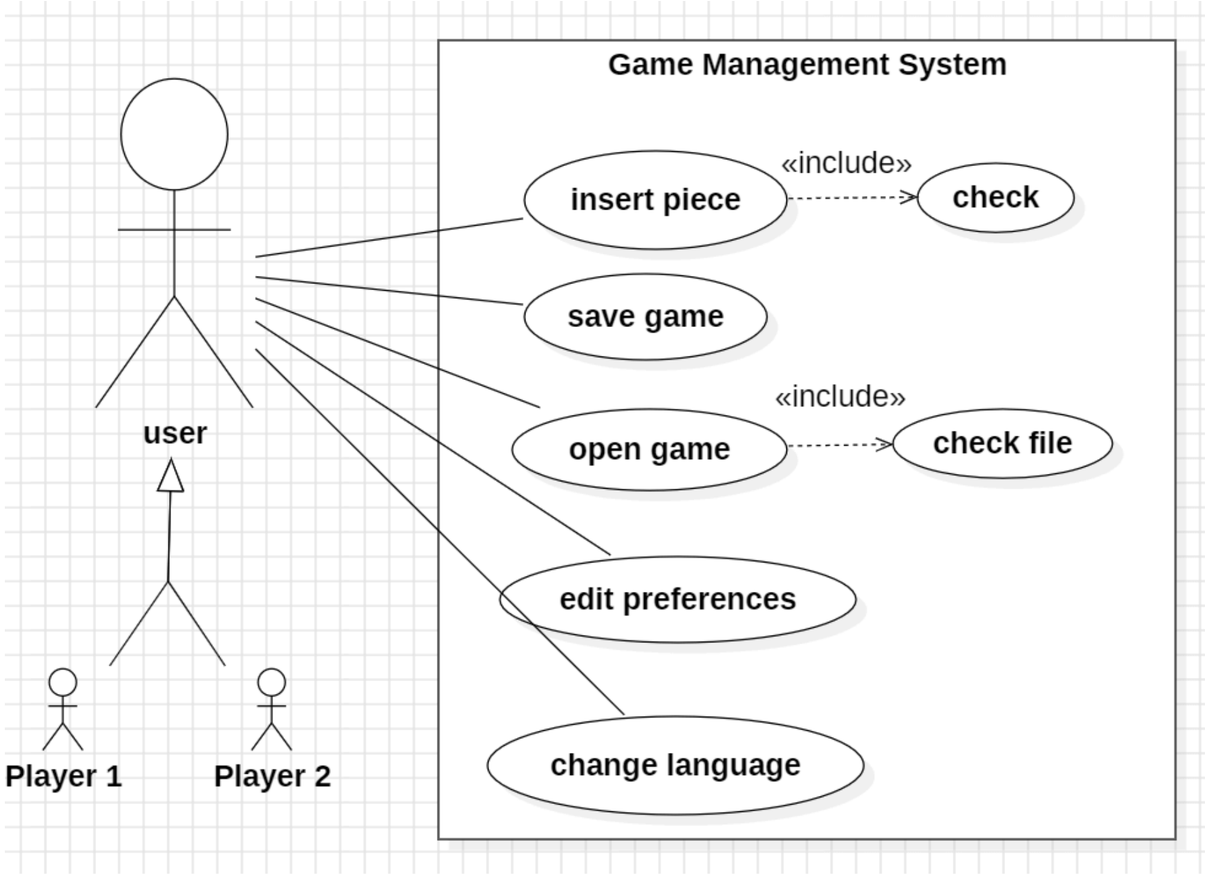
Allow **saving** and **loading** game progress.

Supports **different language** in the UI.

Standalone application with a graphical user interface.

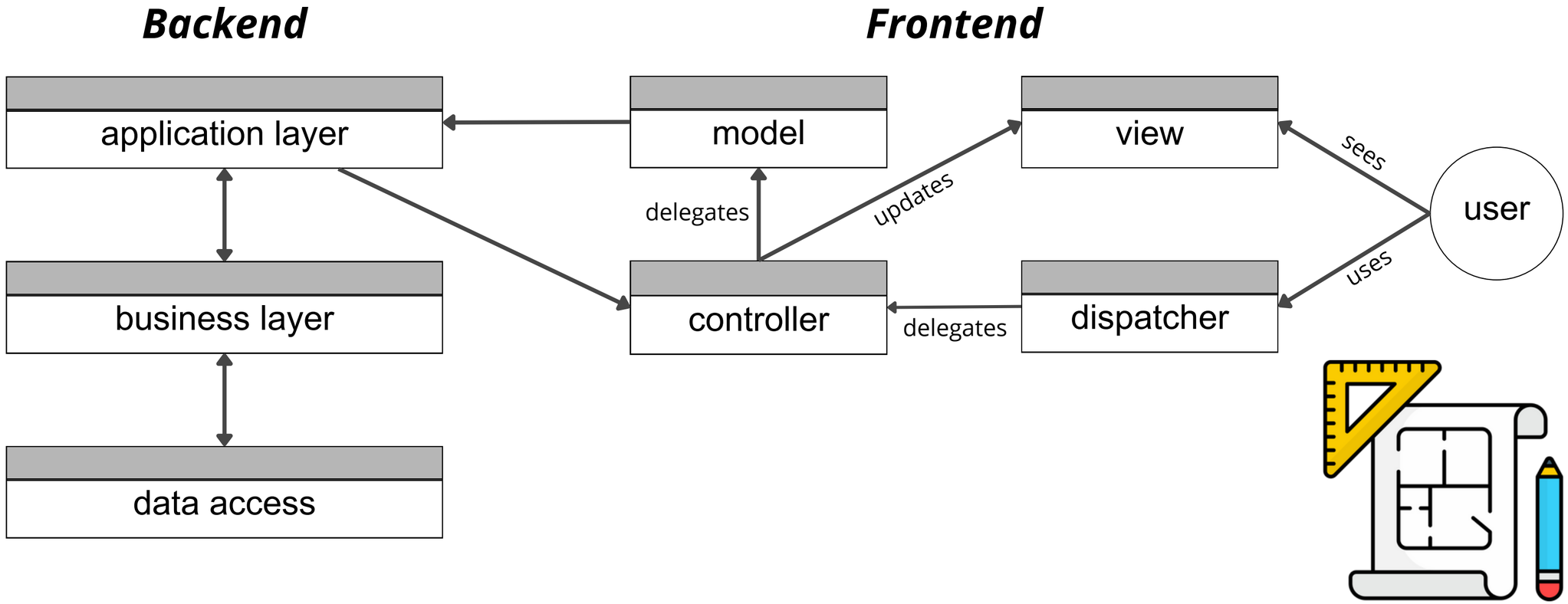
Problem

Use case diagram about game management system



Approach

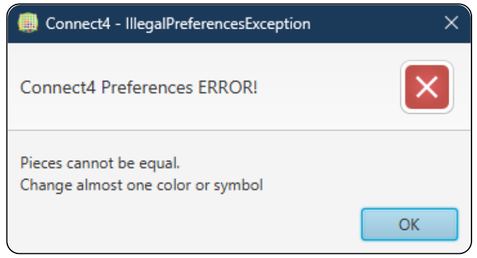
Architectural design: Layering - MVC

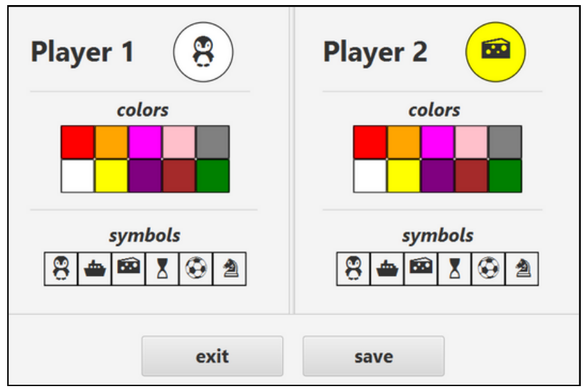


Approach

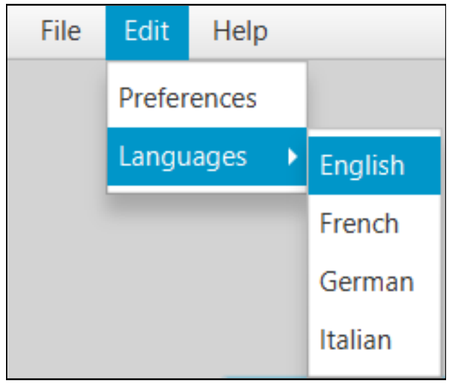
Preferences and Languages

Colours and **symbols** are **selectable** using an user interface.
Save button update preferences in real time.
If **preferences** are **equal** an **error** is throw.



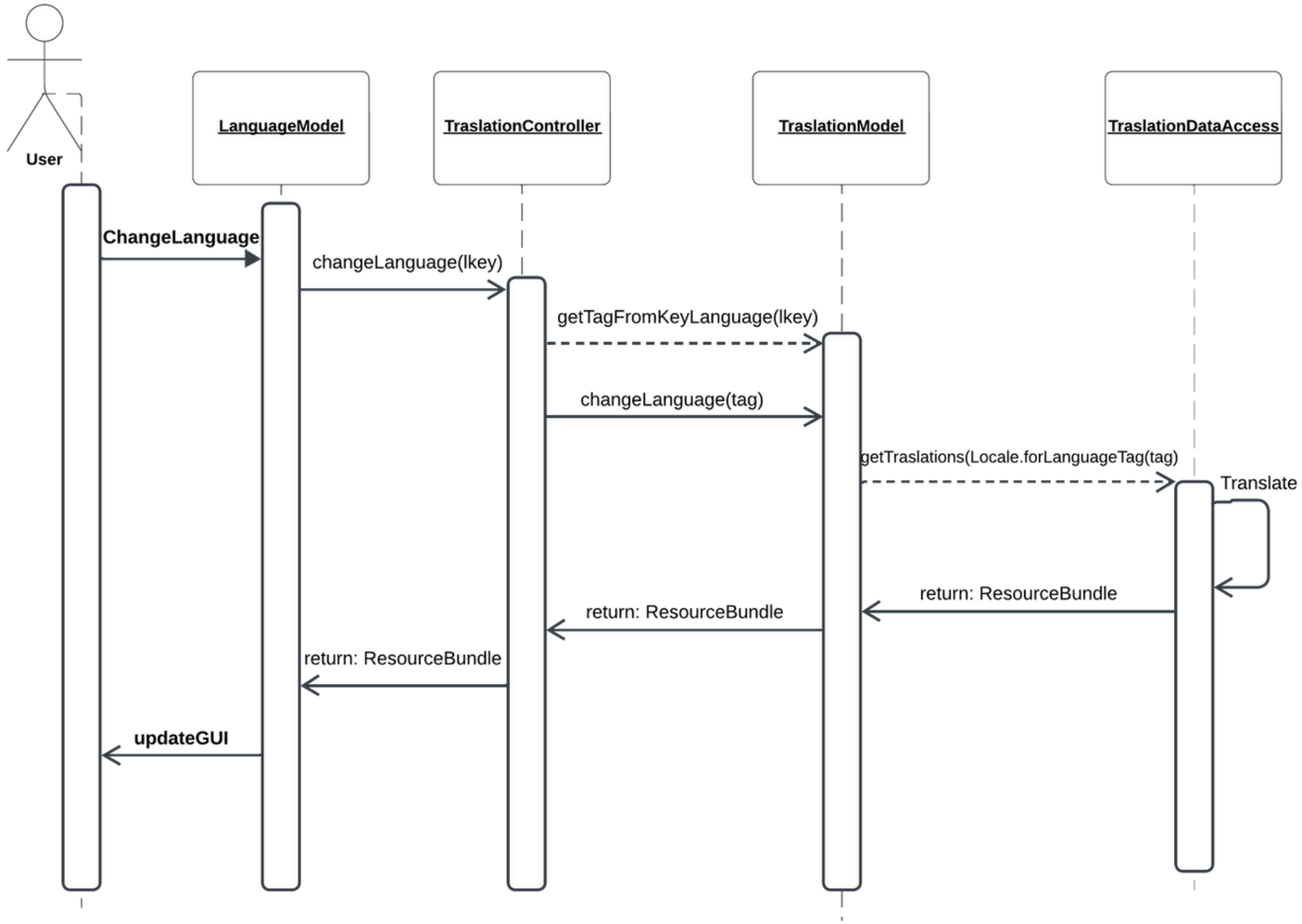


Languages can be changed during entire state of the game.
Real time updating of the language.



Approach

Languages

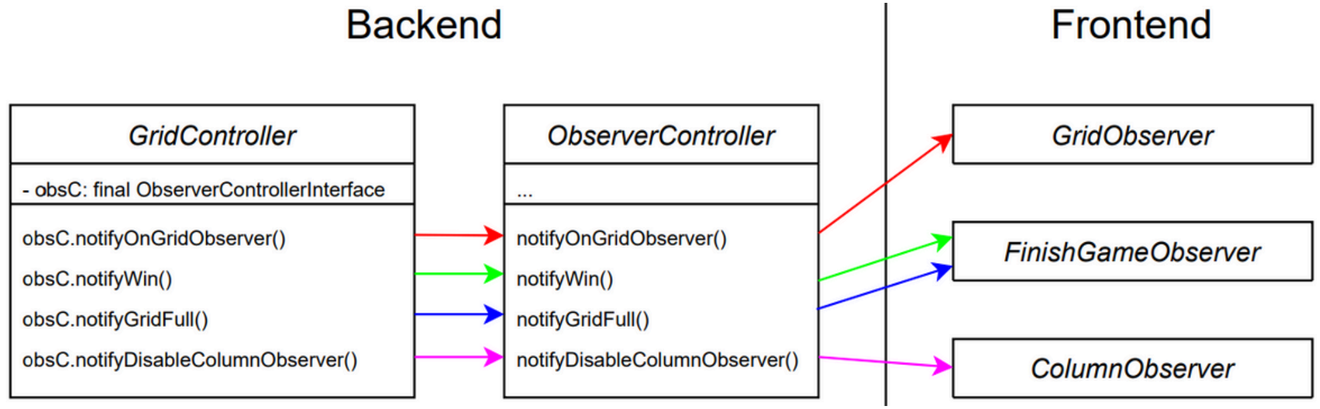


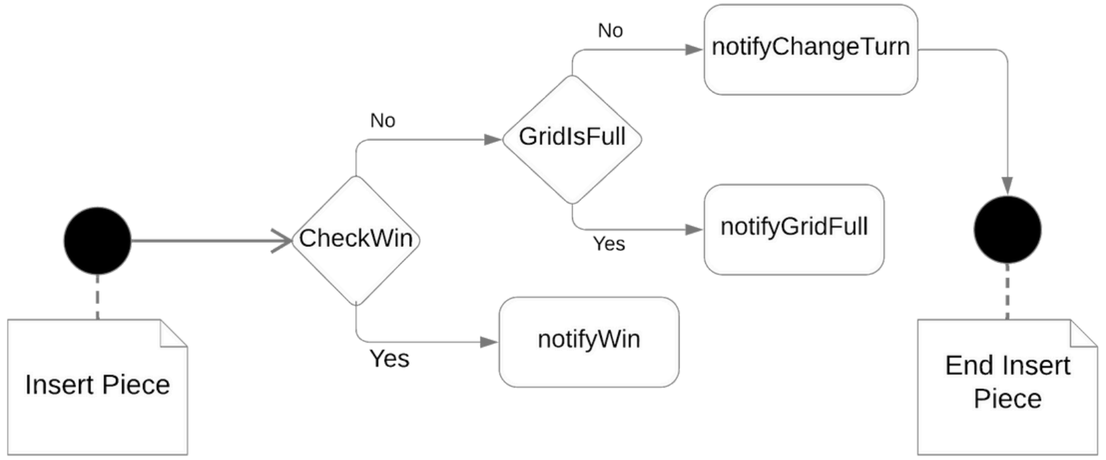
Approach

Observer Pattern

Provides the ability to establish a **dependency** between objects, allowing them to be **notified** and **updated automatically** when the **state of an object changes**.

Promotes loose **coupling** between the observing and observed objects, enhancing **modularity** and **maintainability** in **software design**.





Approach

Game's error management

To **ensure** the game **runs correctly**, we have implemented an **error management system** based on exceptions. When a critical error occurs in the backend, an **exception is thrown** and its handling is **delegated** to the **frontend**.

Backend

```
@Override
public void setNewPreferences(List<Piece> pieces)
        throws IllegalPreferencesException {
    preferencesModel.checkPiecesAreDifferent(pieces);
    preferencesModel.setCurrentPieces(pieces);
    gridModel.setNewPlayerPreferences(pieces);
    observerController.notifyAllGridUpdate(gridModel.getGrid());
}
```

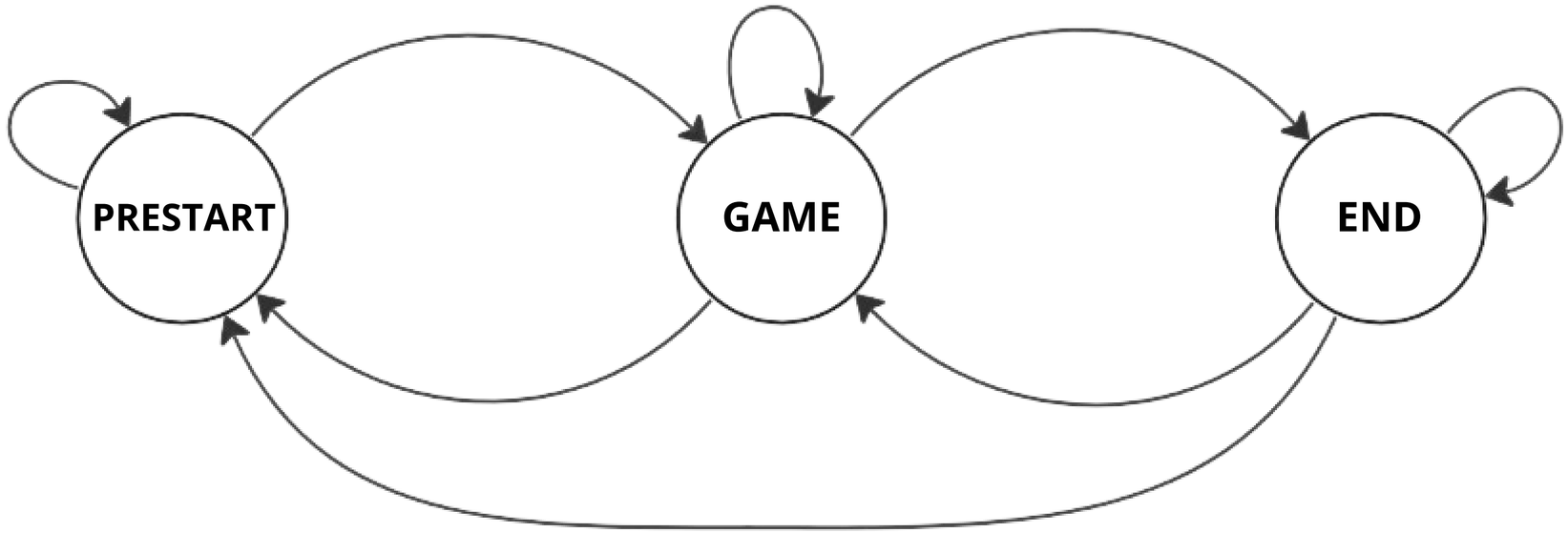
Frontend

```
@Override
public void setNewPreferences(List<Piece> pieces) {
    try {
      preferencesController.setNewPreferences(pieces);
      preferencesDispatcher.exit();
    } catch (IllegalPreferencesException e) {
      errorView.showPopUpError(e.getClass().getSimpleName(),
              e.getMessage());
    }
}
```

Approach

Game status

A **change** of state **during** the **execution** of the game **entails the adherence to certain conditions** of certain classes "(*List<UpdateStatusInterface>*)".



Approach

Saving - Loading game

During the game, the user has the **option to save** the progress within a **json file**.

The software allows to **save** the game into a **new file or to overwrite** the progress on an existing file, if already present.

Saving operations are only **available during** the **game status** and are **notified** to the user via the **infobar**.

The user can **upload** a **previously saved game** at any time. Are accepted **only valid json** file with the correct structure.

```
{
    "grid": [
        … [
            … {
                "col": 4,
                "row": 5,
                "fill": false,
                "player": "Player1"
            }, …
        ] …
    ],
    "turn": false
}
```

Results

Achieved results in line with established requirements

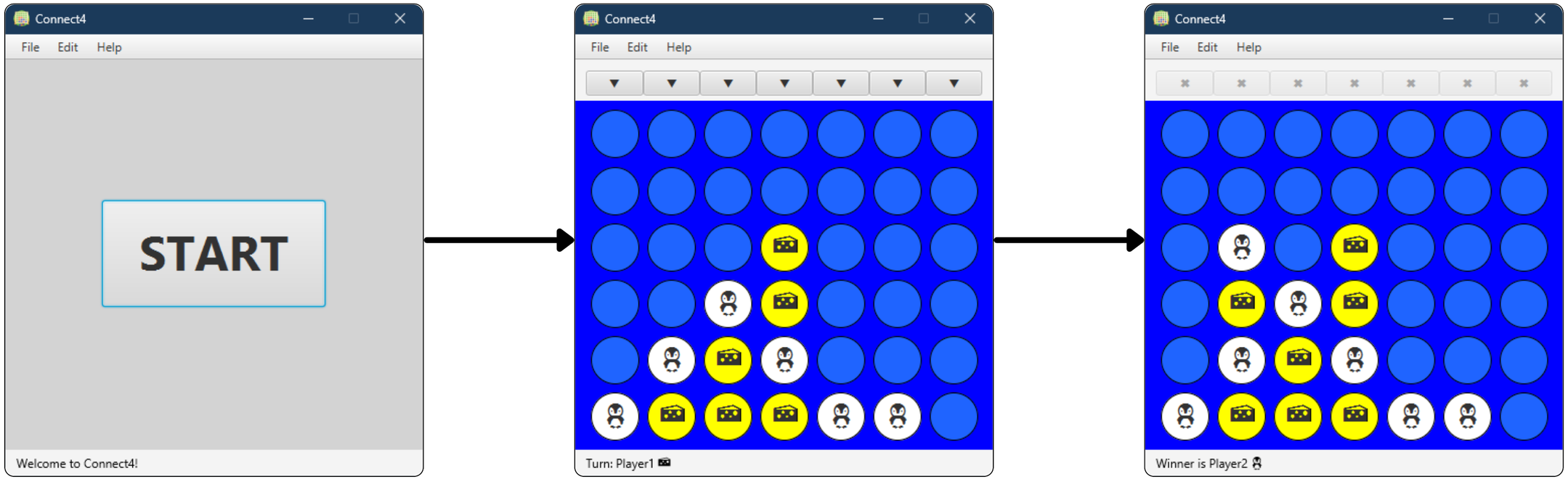
The final software product is in **accordance** with the **initial requirements**, is **compatible with all operating systems** and **supports different languages** that can be change during execution.

The game **operates correctly** in according to the classic rules of Connect Four, **allows customization** of the two players, and **guides** them based on **the actions** they perform.

Program, also **supports game saving** and subsequent **loading** of matches to continue.

Result

Static program demo



Conclusion

Possible optimizations and improvements

To **notify** a **change** itself, we used the **Observer mechanism**. However, the current implementation **does not** properly **comply** with the **Pattern Observer** and SOLID principles, in particular the **"Open-Closed Principle"** (OCP): open for extensions and closed for changes.

This **lack of** adherence to the **OCP** means that our **code** is more **difficult to maintain and to extend**.

THANK YOU FOR YOUR ATTENTION