APRO2 Etap II - Calico

Tymoteusz Malec
Jakub Kusznier
Maksymilian Lepak
Anna Paziewska
Filip Bandrowski

19 maja 2023

Spis treści

1 Wstęp

Etap II projektu przedmiotu APRO2 zakłada stworzenie "Backend'u"-logiki aplikacji. W tym etapie mamy za zadanie najpierw podejść krytycznie do "Design proposal" stworzonego przez poprzednią grupę i zaimplementować kod na podstawie, przez nas zmodyfikowanej, propozycji. Następnie, stworzyć odpowiednie testy jednostkowe oraz dokumentację w postaci Javadoc oraz dokumentu pisemnego.

2 Krytyczna analiza etapu I

Tworząc rozwiązanie projektu, doszliśmy do wniosku, że zapis planszy w postaci tablicy jednowymiarowej jest niewygodny i nieintuicyjny, więc postanowiliśmy zastosować dwuwymiarową tablicę o wymiarach 7x7. Kolejną zmianą jest zastosowanie typu ENUM zamiast zapisywania danych w zmiennej typu String.

Dodatkowo, wcześniejsza wersja używała klasy Cat do reprezentowania i planszy kotów, i żetonów kotów. Zmieniliśmy to w ten sposób, że oprócz klasy Button, która wcześniej reprezentowała konkretnie żetony guzików, dodaliśmy ColorButton(żetony guzików) i CatButton(żetony kotów). Klasa Button teraz reprezentuje po prostu żetony, klasa CatBoard-plansze kotów, a Cat jest enumem deklarującym możliwe rodzaje kotów.

Dokonaliśmy też zmiany struktury Scored-z Klasy na Interfejs.

3 Struktura projektu

3.1 Klasy i struktury

Board: Klasa, która jest odpowiedzialna za generowanie planszy, za sparwdzanie pól sąsiadujących z konkretnym polem, za stawianie żetonów i kafelków na odpowiednie pola i sprawdzanie kształtów powstałych na planszy z kafelków o konkretnych kolorach czy wzorach.

Button: Klasa reprezentująca żetony inne niż kafelki.

Calico: Klasa łącząca tryby gry, rdzeń całego kodu.

Cat: ENUM, który deklaruje możliwe rodzaje żetonów kotów. Bierze pod uwagę takie cechy jak: imię, kropki, preferowany kształt, wielkość preferowanego kształtu.

CatBoard: Klasa, która reprezentuje plansze kotów z ulubionymi wzorami.

CatButton: Klasa reprezentująca żetony kotów.

Client: Klasa, która łączy się z serwerem z innych maszyn.

Color: ENUM, który deklaruje możliwe opcje kolorów na kafelkach.

ColorButton: Klasa reprezentująca żetony guzików.

Field: Klasa reprezentująca pole na planszy 7x7.

Game: Klasa reprezentująca rozgrywkę, odpowiedzialna za takie funkcje jak wykonywanie ruchu, załadowanie i zapisanie gry, sprawdzanie tury.

Player: Klasa reprezentująca gracza. Wykonuje takie funkcje jak np. przydzielenie kafelek projektów.

ProjectTile: Klasa reprezentująca kafelek projektów.

ProjectTileType: ENUM, który deklaruje możliwe rodzaje kefelków projektów.

RegularTile: Klasa reprezentująca zwyczajny kafelek. Zawiera funkcje losowania kafelków, potrzebną do rozgrywki

Scored: Interfejs, który opisuje za co zostały zdobyte punkty.

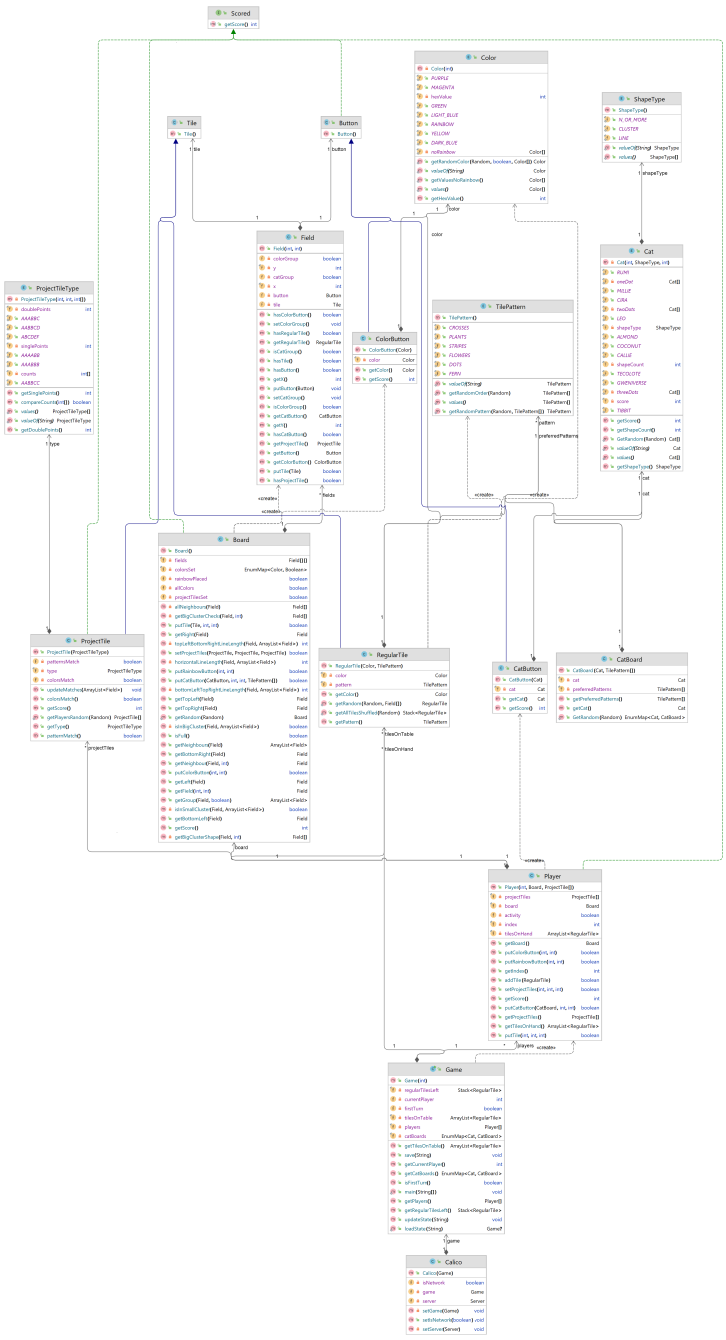
Server:Klasa reprezentująca serwer, do którego będą się łączyć gracze, aby mieć możliwość rozgrywki przez sieć

ShapeType: ENUM, który deklaruje możliwe kształty układów kafelków. Rozróżniane są trzy typy "kształtów" preferowanych przez koty: grupy n płytek o tym samym wzorze, linie oraz grono.

Tile: Klasa reprezentująca kafelki.

TilePattern: ENUM, który deklaruje możliwe opcje wzorów na płytkach.

3.2 Diagramy UML



Rysunek 1: Diagram UML

3.3 Testowanie aplikacji

Aplikacje testujemy za pomocą testów jednostkowych.

Za pomocą serii testów jednostkowych, testujemy czy prawdiłowo są zlicznae punkty za układanie kafelków wokół kafelku projektu. Ten test pozwolił nam na sprawdzenie czy tworzenie planszy, kładzenie kafelków, kładzenie kafelków projektu, przypisywanie cech do kafelków i kafelków projektów (cechy takie jak kolor, wzór czy rodzaj kafelka projektu), zliczanie punktów, poprawnie działa.

Testami jednostokowymi sprawdzamy też poprawność działania kodu odpowiedzialnego za kładzenie żetonów kotów na ich preferowanych wzorach i kształtach. Ten test sprawdza czy kod poprawnie kładzie żeton kota, przypisuje cechy do kafelków, sprawdza preferencje konkretnego kota i rozpoznaje kształty układane na planszy.

3.4 Połączenie sieciowe

Socket Programming w Javie- klasa stworzona do komunikacji pomiedzy Client'em, a Server'em. Socket jest powiązany z numerem portu, dzięki czemu warstwa TCP może zidentyfikować aplikację, do której dane mają zostać wysłane.

3.5 Sterowanie

Sterowanie grą jest możliwe na tym etapie za pomocą wysłynia konkretnych wiadomości, które są sparowane z konkretnymi ruchami w grze. Te wiadomości gracz może wysyłać za pomocą funkcji Client do serwera.

4 Wnioski

Jednym z bardziej przejawających się problemów była ilość zasad w grze Calico. Gra ta, w porównaniu do np. Warcab, jest dosyć skomplikowana, więc część czasu zajęło pojęcie wszystkich istotnych, ale drobnych szczegółów; gracz ma do wykorzystania, aż 6 różnych rodzajów żetonów, których kładzenie jest ściśle określone w zasadach.

Dodatkowo, problemem było również napisanie kodu, który umożliwiłby rozgrywke przez sieć. Jednym z powodów był fakt, że gra umożliwiwa rozgrywkę od 2 do 4 graczom. Problem wynikał w dużej mierze z braku doświadczenia pracy z siecią z poziomu kodu programu. Wykład z rozwiązań sieciowych odbył się późno zwarzywszy na ramy czasowe, które zostały przewidziane na wykonanie drugiego etapu projektu.

Pracowanie nad tym etapem projektu pozwoliło na na rozszerzenie wiedzy w zakresie tworzenia połączeń sieciowego. Tworzenie "Backend'u" gry Calico było wyzwaniem, ze wcześniej wymienionych powodów.

5 Bibliografia

[1] *Zasady gry Calico*. https://y8t6p8a4.map2.ssl.hwcdn.net/downloads/August2020/87c6836a98ae236a9c7c2b9ecedf81ca.pdf?fbclid=IwAR2qc0rvw0UClVv_XipcAGZbz6K_69XOE-_IS5tpjV3Jf6LjD32KlD9k61s.

[2] *Zasady gry Calico*. https://www.youtube.com/watch?v=EXQEzdVMd5I.

[3] *Socket Programming*. https://www.javatpoint.com/socket-programming.