Wojskowa Akademia Techniczna

im. Jarosława Dąbrowskiego w Warszawie



PROGRAMOWANIE ZDARZENIOWE

Dokumentacja projektu

pt.: Gra Gomoku

**Prowadzący:**

por. mgr inż. Jakub Kędzior

**Autor:**

Konrad Bobryk

**1. Opis problemu**

Gomoku to dwuosobowa gra logiczno-strategicza, której tradycja sięga VII wieku p.n.e. Z reguły jest rozgrywana za pomocą pionków Go (czarne i białe kamienie) na planszy Go o wymiarach 15x15 lub 19x19. Gracze na zmianę wykonują posunięcia kładąc kamień w swoim kolorze na jednym z pustych pól planszy. Zwycięża ten z graczy, który jako pierwszy ułoży dokładnie 5 swoich kamieni w pionie, poziomie lub po przekątnej.

**2. Koncepcja rozwiązania**

Podczas realizacji zadania ograniczyłem się do gier na planszy o wymiarach 15x15. Podczas gry gracze na zmianę wykonują ruch za pomocą myszy mając do dyspozycji ograniczony czas na posunięcie, który jest ustawiany przed rozpoczęciem rozgrywki. Niestety ze względu na brak czasu nie udało mi się również zaimplementować odmiany Swap2, która jest jedyną wersją gry graną na turniejach.

**3. Wymagania funkcjonalne**

Wymagania funkcjonalne zostały narzucone przez Prowadzącego. Mimo, że udało mi się zrealizować wszystkie te wymagania, to implementacja części z nich była bardzo wymuszona – w normalnych warunkach nie zdecydowałbym się na takie rozwiązania, natomiast tutaj ze względu na narzucone wymagania pewne rzeczy są zrealizowany w taki sposób.

**4. Stos technologiczny. Wykorzystane biblioteki i technologie.**

**4.1. Stos technologiczny**

Wykorzystane technologie to między innymi: Java 8, SQL, Swing, Maven. Wykorzystane zewnętrzne biblioteki (dodane pliki .jar): JTattoo, Apache Commons Net, Microsoft SQL.

**4.2. JTattoo**

a) Opis: Jest to biblioteka dostarczająca gotowe skórki do aplikacji.

b) Uzasadnienie wykorzystania: Jednym z wymagań funkcjonalnych było dodanie skórek do aplikacji. Skorzystanie z biblioteki JTattoo bardzo ułatwiło zadanie umożliwiając duży wybór, jak również prostą implementację.

c) Sposób wykorzystania: W menu aplikacji możemy zmienić skórkę poprzez wybranie odpowiedniej opcji. Sprowadza się to do skorzystania z UIManagera i SwingUtilities.

**4.3. Apache Commons Net**

a) Opis: Biblioteka dostarczająca różnych rozwiązań związanych z serwisami webowymi.

b) Uzasadnienie wykorzystania: Biblioteka ułatwia realizację wymagania związanego z wykorzystaniem zasobu webowego.

c) Sposób wykorzystania: NTPUDPClient oraz TimeInfo zostały wykorzystane (razem z innymi klasami spoza biblioteki) do pobrania aktualnej daty i godziny ze strony „1a.ncomputers.org”.

**4.4. Microsoft SQL**

a) Opis: Rozwiązania dotyczące bazy danych i języka SQL.

b) Uzasadnienie wykorzystania: Biblioteka potrzebna przy obsłudze bazy danych.

c) Sposób wykorzystania: Dołączenie pliku .jar do projektu pozwala na łączenie się z bazą danych.

**4.5. Dług technologiczny**

Ze względu na brak czasu w wielu miejscach w projekcie są pewne niedociągnięcia. Ponadto niektóre rozwiązania nie są optymalne, ponieważ była potrzeba realizacji jednego z wymagań.

**5. Wskazanie spełnienia wymagań dotyczących projektu**

**5.1. Aplikacja napisana w Swingu/Java Fx**

Wybraną technologią był Swing.

**5.2. Własne zdarzenia świadczące o wykonaniu dedykowanej logiki**

a) Zmiana języka aplikacji w trakcie jej działania

Jest to pierwsze (i zarazem jedyne w pełni „naturalne”) wykorzystanie własnych zdarzeń. Za pomocą przycisków w menu możemy wybrać język angielski lub język polski. Wszystkie komponenty zostają wtedy zaktualizowane do odpowiedniej wersji językowej.

b) Rozpoczęcie gry

Polega na tym, że niektóre zmienne w odpowiednich klasach przyjmują odpowiednie wartości.

c) Zakończenie gry

Jak wyżej – zmiana wartości niektórych parametrów.

d) Włączenie tła w niektórych panelach

Jest to rozwiązanie zastosowane „na siłę” – z braku pomysłu na dwa dodatkowe własne zdarzenia postanowiłem dodać JMenu, w którym możemy włączyć lub wyłączyć szare tło w niektórych panelach.

e) Wyłączenie tła w niektórych panelach

Jak wyżej.

**5.3. Własne klasy wyjątków semantycznie związane z problematyką błędów występujących w aplikacji**

a) NotAlphanumericCharactersException

W panelu opcji gry gracze wprowadzają swoje nicki, przy czym mają się one składać tylko ze znaków alfanumerycznych (małe i wielkie litery oraz cyfry). W przypadku gdy jeden z nicków zawiera niedozwolony znak zostaje rzucony ten wyjątek, a odpowiedni komunikat zostaje wyświetlony przy błędnym nicku gracza.

b) SameNicknamesException

Nicki graczy nie mogą być identyczne – gdyby tak było, to gracz grałby sam ze sobą, co oznaczałoby, że należy mu za tę samą grę przypisać zwycięstwo i porażkę, co jest bez sensu. Aby uniknąć takiej sytuacji zostaje rzucony wyżej wymieniony wyjątek, gdy nicki są takie same, a w panelu zostają wyświetlone odpowiednie komunikaty.

c) TooLongNickException

Nick gracza ma być nie dłuższy niż 15 znaków – gdy jest on dłuższy zostaje rzucony wyżej wymieniony wyjątek, a w panelu przy zbyt długim nicku pojawia się odpowiedni komunikat.

d) TooShortNickException

Nick gracza musi składać się z co najmniej 3 znaków – gdy jest on krótszy zostaje rzucony wyżej wymieniony wyjątek, a w panelu przy zbyt krótkim nicku pojawia się odpowiedni komunikat.

e) TooLongTimeForMoveException

Maksymalny czas na posunięcie gracza to 60 sekund – gdy użytkownik wpisze więcej zostaje rzucony powyższy wyjątek, a w panelu przy czasie na ruch pojawia się odpowiedni komunikat.

f) TooShortTimeForMoveException

Minimalny czas na posunięcie gracza to 10 sekund – gdy użytkownik wpisze mniej zostaje rzucony powyższy wyjątek, a w panelu przy czasie na ruch pojawia się odpowiedni komunikat.

**5.4. Komponenty graficzne własnego pomysłu**

a) MyTextArea rozszerzający klasę JTextArea

Wykorzystany w dwóch panelach (AboutMePanel oraz GameInfoPanel) pozwala dodać tło do obszaru tekstowego.

b) Rozszerzenie klas JScrollPane i JTable

Tutaj nie stworzyłem osobnej klasy, ponieważ modyfikacji wymagały 2 komponenty, ponadto zmiana ta zajmuje tylko kilka linijek. Efektem tego jest obraz wgrany jako tło do tabeli w panelu BestPlayersPanel.

c) Klasy rozszerzające JPanel

Kilka klas (między innymi GamePanel, GameOptionsPanel, MainMenuPanel itd.) rozszerza klasę JPanel. Nadpisuję tu metodę paintComponent(Graphics g), co pozwala między innymi dodać tło czy narysować planszę i pionki w panelu gry.

**5.5. Parametryzowanie aplikacji przez pliki properties**

W aplikacji są używane 4 pliki z rozszerzeniem .properties: Lang\_en\_GB i Lang\_pl (odpowiadające za internacjonalizację aplikacji, odpowiednio angielska i polska wersja językowa), logger (odpowiada za konfigurację logów) oraz appProperties, z której są odczytywane różne wartości wykorzystywane w aplikacji (między innymi wielkość okna, wielkość komponentów, wielkość czcionek itd.). Do ostatniego pliku wciąż można dodać nowe wartości – np. kolory czcionek,

**5.6. Opis źródła danych XML**

Jest to wymaganie, na które nie miałem pomysłu. Doszedłem więc do wniosku, że skoro od jakiegoś czasu chciałem zacząć naukę języka hiszpańskiego, to dlaczego by nie wyświetlać losowego hiszpańskiego słówka po zakończeniu gry? Zatem za każdym razem, gdy gra się zakończy oprócz standardowego komunikatu (zwycięzca, czy chcesz zagrać rewanż) wyświetlane jest losowe słowo po polsku, angielsku i hiszpańsku. Odzwierciedleniem tego jest plik spanishWords.xml, gdzie każdy „wierzchołek” (słowo) składa się z tłumaczenia danego słówka na język polski, angielski i hiszpański. Sam odczyt z pliku XML odbywa się w klasie ReadXML za pomocą parsera XML DOM.

**5.7. Dostarczenie danych do aplikacji z wykorzystaniem relacyjnej bazy danych**

Baza danych, z której korzystam jest bardzo prosta – zawiera 1 encję (Player), w której każdemu graczowi odpowiada jego nick, liczba rozegranych gier i liczba wygranych gier. Przy starcie aplikacji dane są pobierane z bazy i zapisywane w tabeli w panelu BestPlayersPanel, posortowane malejąco według liczby wygranych gier. Po zakończeniu każdej rozgrywki baza danych jest uaktualniana; jeśli gracze są już w bazie, to w zależności od wyniku jest zwiększana liczba wygranych i liczba rozegranych (w przypadku wygranej) gier lub tylko liczba rozegranych gier (w przypadku przegranej), natomiast jeśli graczy nie ma w bazie, to zostają oni dodani do bazy z uwzględnieniem wyniku ich pierwszej rozgrywki.

**5.8. Opis wykorzystanej usługi webowej**

Zdecydowałem się na wyświetlenie daty i godziny rozpoczęcia rozgrywki w panelu gry. W tym celu zastosowałem bibliotekę Apache Commons Net, a adres, z którego pobieram dane to 1a.ncomputers.org

**5.9. Wykorzystane wzorce projektowe**

a) Observer – przy implementacji własnych zdarzeń

b) Chain of responsibility – mechanizm logowania

c) Composite – swing, zagnieżdżone kontenery

d) Strategy – wykorzystanie layout managerów

e) Mediator – implementacja grup przycisków (ButtonGroup)

f) Command – zastosowanie Action

**5.10. Wskazanie operacji asynchronicznych**

Użyłem SwingWorkera przy wszelkich operacjach związanych z bazą danych, a także przy parsowaniu pliku XML.

**5.11. Mechanizm logowania**

Konfiguracja mechanizmu logowania znajduje się w pliku logger.properties. Logi są wyświetlane w konsoli, jak również są zapisywane do pliku w folderze logs. W obu przypadkach poziom, od którego podejmowana jest akcja to INFO. Ponadto w rotacji może znajdować się maksymalnie 10 plików logujących, a wielkość każdego z nich jest ograniczona z góry przez 50000 bajtów. Dodatkowo opcja append=true pozwala na dopisywanie do istniejących logów. Każda klasa ma swojego loggera, który jest używany za każdym razem, gdy zostaje rzucony wyjątek.

**5.12. Internacjonalizacja aplikacji (przystosowanie do 2 wersji językowych – polskiej i angielskiej)**

Do odczytu poszczególnych komunikatów w poszczególnych wersjach językowych służą pliki Lang. Domyślnym językiem aplikacji jest angielski. Zmiana języka aplikacji w trakcie wiąże się z naciśnięciem odpowiedniego przycisku w menu.

**5.13. Menu aplikacji i Action**

Po starcie aplikacji znajdujemy się w panelu menu głównego – wszystkie przyciski, które się tam znajdują mają swoje odpowiedniki w menu aplikacji, do których podpięta jest ta sama akcja.

**5.14. Zmiana PLAF**

Do realizacji tego wymagania użyłem biblioteki JTattoo. Użytkownik ma do dyspozycji 8 skórek aplikacji, może je zmieniać poprzez naciśnięcie odpowiedniej nazwy skórki w menu aplikacji.

**5.15. Zaawansowane widoki GUI**

Wszystkie klasy rozszerzające JPanel używają GridBagLayout do ułożenia komponentów, mimo, że w niektórych panelach tych komponentów jest bardzo mało. Najbardziej jaskrawym przykładem użycia GridBagLayout jest GamePanel, gdzie za jego pomocą jest rozmieszczonych 225 przycisków (są one niewidoczne), po ich kliknięciu w ich miejscu rysowany jest pionek gracza i zapisywany jest ruch.

**6. Problemy w trakcie implementacji**

Najczęściej spotykanym problemem (oprócz braku czasu na zastanowienie się nad nim) był brak pomysłów na „sensowne” wdrożenie poszczególnych wymagań, czego efektem jest np. wcześniej już wymienione wyświetlanie hiszpańskich słówek. Ponadto sporo problemów sprawiło mi stworzenie bazy danych (wybranie odpowiedniego narzędzia, dziwny i długi proces weryfikacji stosowany przez Microsoft).