Politechnika Świętokrzyska w Kielcach Wydział Elektrotechniki, Automatyki i Informatyki	
Programowanie Obiektowe 2 - Projekt – Informatyka II rok, Rok akademicki - 2023/2024	
Temat projektu: Saper	**Zespół:** Mateusz Figiel Kacper Giełżecki **Grupa:** 2ID12A

1. Ogólny opis projektu wraz z informacjami o technologiach, framework'ach, bibliotekach używanych w projekcie:

Użyte technologie:

Frameworki:

- Maven
- JUnit 5

Biblioteki:

- AWT (abstract window toolkit)
- Swing

Środowisko programistyczne:

- IntelliJ IDEA

Język programowania:

- Java

Opis:

Nasz projekt skupia się na stworzeniu klasycznej gry w Sapera, zaimplementowanej w języku Java. Aby ułatwić rozwój i testowanie, użyto zaawansowanego środowiska programistycznego IntelliJ IDEA, a także korzystano z popularnych bibliotek graficznych AWT i Swing do tworzenia interfejsu użytkownika.

Głównym celem projektu było stworzenie gry o trzech różnych poziomach trudności: łatwym (Easy), średnim (Medium) i trudnym (Hard). Po uruchomieniu programu, użytkownikowi

ukazuje się okno menu, gdzie może wybrać preferowany poziom trudności. Każdy z wybranych poziomów otwiera planszę gry z zakrytymi polami, reprezentującymi nieodkryte obszary. Gracz ma możliwość odkrywania pól, a także flagowania tych, które podejrzewa o ukrywanie bomb. Gracz ma także możliwość zrestartowania rozgrywki przyciskiem znajdującym się u góry wyświetlanego okna.

Różnice pomiędzy poziomami trudności polegają głównie na ilości umieszczonych bomb, przy zachowaniu identycznej wielkości mapy. Projekt został zabezpieczony przed błędami dzięki implementacji testów przy użyciu frameworka JUnit 5. Testy te sprawdzają poprawność funkcji i logiki gry, co przyczynia się do zwiększenia niezawodności aplikacji.

Po zakończeniu rozgrywki użytkownik może spotkać jedno z dwóch okienek: wygranej lub przegranej. Okna te automatycznie znikają po upływie 3 sekund, co pozwala graczowi na szybkie przejście do kolejnej partii bez zbędnego oczekiwania.

Warto zauważyć, że struktura projektu opiera się na frameworku Maven, co ułatwia zarządzanie zależnościami i budowę aplikacji. Dzięki zastosowaniu wspomnianych technologii, projekt Sapera w Javie stanowi kompleksowe podejście do implementacji gry logicznej, uwzględniając zarówno aspekty interfejsu użytkownika, jak i testowania kodu.

2. Informacje na temat funkcjonalności projektu:

Framework Maven:

- Zarządzanie Zależnościami: Projekt korzysta z frameworka Maven do efektywnego zarządzania zależnościami i budowania aplikacji.

Framework JUnit5:

- Zastosowanie Frameworku: Projekt wykorzystuje framework JUnit 5 do implementacji testów jednostkowych.

- Testy Funkcjonalności: Testy sprawdzają poprawność funkcji i logiki gry, co zwiększa niezawodność aplikacji.

Interfejs użytkownika:

- Okno Menu: Po uruchomieniu programu pojawia się okno menu, które umożliwia użytkownikowi wybór poziomu trudności.

- Plansza Gry: Po wyborze poziomu trudności użytkownik otrzymuje planszę gry z zakrytymi polami reprezentującymi obszary do odkrycia.

- Interakcja: Gracz może kliknąć na pola, aby odkryć je, oraz oznaczać je flagami w przypadku podejrzenia o obecność bomby. Gracz ma również możliwość zrestartowania obecnej rozgrywki wciskając przycisk znajdujący się u góry okna gry.

Poziomy trudności:

- Trzy Poziomy: Istnieją trzy poziomy trudności - łatwy (Easy), średni (Medium), trudny (Hard), z różną ilością bomb.

- Identyczna Mapa: Mimo różnic w ilości bomb, wielkość mapy pozostaje taka sama dla każdego poziomu trudności.

Okna wygranej i przegranej:

- Okno Wygranej: Po zakończeniu gry z wynikiem zwycięstwa, pojawia się okno informujące o sukcesie.

- Okno Przegranej: W przypadku przegranej, pojawia się okno informujące o niepowodzeniu.

- Automatyczne Znikanie: Obie formy okienek znikają automatycznie po 3 sekundach, umożliwiając szybkie rozpoczęcie nowej rozgrywki.

3. Informacje na temat sposobu uruchomienia oraz obsługi projektu:

Projekt został zaprojektowany do uruchamiania się poprzez dowolne środowisko programistyczne, które obsługuje język Java. Aby pomyślnie uruchomić program, konieczne jest dostępność bibliotek AWT oraz Swing. Ponadto, wymagane jest skonfigurowanie frameworków Maven oraz JUnit 5, aby zapewnić poprawne zarządzanie zależnościami i umożliwić wykonywanie testów jednostkowych.

W celu skutecznego uruchomienia projektu, konieczne jest również odpowiednie skonfigurowanie plików znajdujących się w folderze projektu (takich jak pliki .png). Wymagane dane konfiguracyjne obejmują wszystkie niezbędne ustawienia, takie jak ścieżki do zasobów, połączenia z bazą danych czy inne parametry konfiguracyjne specyficzne dla projektu.

Po spełnieniu wszystkich wymagań i skonfigurowaniu projektu, można uruchomić go za pomocą klasy Main. Warto upewnić się, że wszystkie kroki konfiguracyjne zostały dokładnie wykonane, aby uniknąć potencjalnych trudności podczas uruchamiania projektu.

4. Informacje na temat stworzonych klas, metod, funkcji (bez kodu źródłowego) z opisem ich podstawowej

funkcjonalności (przyjmowanymi parametrami, wartościami zwracanymi) oraz ich przeznaczeniem:

Klasy:

Main.java:

- Klasa **Main** pełni rolę punktu wejścia do programu, gdzie w metodzie **main** tworzony jest obiekt klasy **Menu**, inicjalizując go szerokością i wysokością.

- Klasa **Menu** reprezentuje menu gry i jest tworzona z określoną szerokością i wysokością. Jej konstruktor przyjmuje te parametry. O ile w dostarczonym kodzie brak konkretnych metod wewnątrz **Menu**, można przypuszczać, że w rzeczywistej implementacji zawiera metody obsługujące interakcję z graczem, uruchamiające nową grę czy przekierowujące do innych ekranów gry.

GUI.java:

- Klasa GUI jest odpowiedzialna za tworzenie interfejsu graficznego użytkownika (GUI) dla gry w Saper. W polu window przechowywany jest obiekt klasy JFrame, reprezentujący główne okno aplikacji. Konstruktor klasy ustawia rozmiar okna, kolor tła, lokalizację, tytuł, niemożliwość zmiany rozmiaru, układ (layout), a następnie sprawia, że okno staje się widoczne dla użytkownika.

- W praktyce, implementacja GUI może obejmować dodatkowe metody obsługujące różne komponenty interfejsu, takie jak przyciski, etykiety czy pola tekstowe. Te metody mogą reagować na zdarzenia użytkownika, jak również manipulować interfejsem w zależności od logiki gry. W dostarczonym kodzie koncentrujemy się głównie na ustawieniach okna, a brakujące elementy interfejsu mogą być dodawane w dalszych etapach implementacji.

Menu.java:

- Klasa Menu pełni kluczową rolę w zarządzaniu interfejsem użytkownika dla menu gry Saper. Jej konstruktor inicjalizuje obiekt GUI i ustawia kontener, a następnie dodaje przyciski reprezentujące różne poziomy trudności. Każdy przycisk jest powiązany z obiektem buttonMouseListener, który reaguje na zwolnienie myszy nad przyciskiem. W zależności od wybranego poziomu trudności, tworzone są nowe obiekty klas GUI, Saper i PrzyciskReset z odpowiednimi parametrami.

- Warto zwrócić uwagę, że w metodzie czytajPrzyciski() zastosowano skrócony zapis

switch z wykorzystaniem składni switch expression dostępnej od Javy 12. Ten kod zakłada dostępność klas GUI, Saper i PrzyciskReset oraz ich konstruktorów, które przyjmują określone parametry. Kluczowym celem klasy Menu jest umożliwienie graczowi wyboru poziomu trudności przed rozpoczęciem gry w Saper.

OknoPrzegranej.java:

- Klasa OknoPrzegranej odpowiada za prezentację interfejsu informującego o przegranej w grze Saper. W jej konstruktorze tworzony jest obiekt GUI o określonych wymiarach, ustawiany jest tytuł głównego okna na "PRZEGRANA", a następnie przygotowywane są komponenty interfejsu, takie jak etykieta (JLabel) służąca do wyświetlenia obrazu związanej z przegraną.

- Dodatkowo, konstruktor ustawia obraz przegranej oraz konfiguruje timer, który automatycznie zamyka okno po upływie pewnego czasu. Metoda zamknijOkno() jest odpowiedzialna za zamknięcie okna poprzez pobranie rodzica (JFrame) z kontenera con i wywołanie na nim metody dispose().

- Dodano również metodę getWindow(), która jest getterem dla pola window, umożliwiając dostęp do obiektu klasy JFrame reprezentującego główne okno przegranej.

- Klasa OknoPrzegranej skupia się na prezentacji graficznego interfejsu w sytuacji przegranej w grze Saper, zapewniając kompleksowe rozwiązanie do obsługi tego scenariusza.

OknoWygranej.java:

- Klasa OknoWygranej odpowiada za prezentację interfejsu informującego o wygranej w grze Saper. W konstruktorze tworzony jest obiekt GUI o określonych wymiarach, ustawiając jednocześnie tytuł głównego okna na "WYGRANA". Klasa zawiera komponenty interfejsu, takie jak etykieta (JLabel) służąca do wyświetlania obrazu związanej z wygraną.

- Warto zwrócić uwagę, że klasa wykorzystuje obraz zasobu, który jest wczytywany poprzez ImageIcon zdefiniowany w lokalizacji "src/main/resources/Wygrana.png". Ponadto, konstruktor ustawia obraz w etykiecie oraz konfiguruje timer do automatycznego zamknięcia okna po upływie pewnego czasu.

- Metoda zamknijOkno() jest odpowiedzialna za zamknięcie okna wygranej poprzez pobranie rodzica (JFrame) z kontenera con i wywołanie na nim metody dispose(). Dodatkowo, dostarczono getter getWindow(), który umożliwia dostęp do obiektu JFrame reprezentującego główne okno wygranej.

- Klasa OknoWygranej dostarcza kompleksowe rozwiązanie dla prezentacji interfejsu informującego o wygranej, uwzględniając elementy graficzne oraz funkcję automatycznego zamknięcia okna po pewnym czasie.

Pole.java:

- Klasa Pole stanowi reprezentację pojedynczego pola w grze Saper. Każde pole przechowuje informacje dotyczące liczby bomb w jego sąsiedztwie (bombyobok), czy zostało naciśnięte (wcisniety), czy na nim znajduje się bomba (bomba), oraz czy została postawiona flaga (flaga). Dodatkowo, klasa zawiera referencję do obiektu klasy JButton (button), który może być związany z danym polem.

- Konstruktor klasy jest prosty i nie przyjmuje żadnych parametrów. Inicjalizuje on obiekt Pole, ustawiając wartości początkowe pól na domyślne. Kluczowym celem tej klasy jest przechowywanie informacji o pojedynczym polu, które może być częścią planszy gry.

- Warto zauważyć, że klasa Pole nie zawiera specyficznych metod obsługujących interakcje czy logikę gry, co sugeruje, że taka funkcjonalność mogłaby zostać zaimplementowana w innych klasach, które operują na obiektach klasy Pole. Klasy te mogą obsługiwać interakcje gracza, obliczać liczby bomb w sąsiedztwie oraz implementować logikę gry.

PrzyciskReset.java:

- Klasa PrzyciskReset jest odpowiedzialna za reprezentację przycisku resetowania w grze Saper. Konstruktor klasy inicjalizuje obiekt, ustawiając jego współrzędne, szerokość i wysokość, oraz tworzy przycisk graficzny. Przycisk ten może zmieniać swoje ikony w zależności od różnych interakcji gracza, takich jak przytrzymanie, puszczenie, przegrana czy wygrana.

- W klasie zdefiniowano cztery ikony przycisku:
- IconReset - domyślna ikona przycisku resetowania.
- IconResetPressed - ikona przycisku resetowania w momencie, gdy jest przytrzymany.
- IconResetLose - ikona przycisku resetowania w przypadku przegranej.
- IconResetWygrana - ikona przycisku resetowania w przypadku wygranej.

- Konstruktor dodaje przycisk do określonego kontenera i przypisuje mu nasłuchiwanie zdarzeń myszy przy użyciu obiektu buttonMouseListener. Dodatkowo, dostępne są metody, takie jak przyciskGdyTrzymamy(), przyciskGdyPuszczamy(), przyciskPrzegrana(), przyciskWygrana(), które zmieniają ikony przycisku w zależności od interakcji gracza.

- Klasa PrzyciskReset dostarcza kompleksowe rozwiązanie dla interfejsu przycisku resetowania w grze Saper, umożliwiając dynamiczną zmianę wyglądu w zależności od akcji użytkownika oraz sytuacji w grze.

Saper.java:

- Klasa Saper.java pełni kluczową rolę w implementacji gry w Saper.

Klasa Saper:

- Klasa Saper jest centralnym elementem implementacji gry w Sapera. Posiada pola przechowujące informacje o szerokości, wysokości planszy oraz liczbie min. Dodatkowo, klasa zawiera obiekty ImageIcon reprezentujące różne ikony wykorzystywane w grze, takie jak ikony pól, miny czy flagi. Pole pole to dwuwymiarowa tablica obiektów klasy Pole, reprezentujących poszczególne pola na planszy.

Konstruktor:

- Konstruktor klasy Saper inicjalizuje obiekt gry ustawiając jego wymiary, liczbę min, ikony oraz obsługę przycisków. Następnie rozmieszcza pola na planszy, generuje bomby i oblicza liczbę bomb sąsiadujących z każdym polem.

Metody:

- czytajPrzyciski():
- Metoda ta tworzy obiekt klasy MouseAdapter, który obsługuje zdarzenia myszy związane z przyciskami na planszy. Decyduje, co powinno się wydarzyć po naciśnięciu i puszczeniu przycisku myszy.

- odkryjPole(int x, int y):
- Metoda służy do odkrywania pól na planszy. Rekurencyjnie odkrywa puste obszary, wykorzystując rekurencję na obszarze 3x3.

- flagowanie(int x, int y):
- Metoda dodaje lub usuwa flagę na polu o określonych współrzędnych.

- rozlozPola():
- Metoda rozmieszcza przyciski reprezentujące pola na planszy gry.

- generujBomby():
- Metoda generuje bomby na losowych polach planszy.

- sprawdzIleObok():
- Metoda sprawdza, ile bomb sąsiaduje z każdym polem na planszy i zapisuje tę informację w odpowiednich polach klasy Pole.

- resetGame():
- Metoda resetuje grę, przywracając początkowe ustawienia planszy.

- przegrana():
- Metoda wywoływana po przegranej grze. Wyświetla okno z informacją o przegranej, odkrywa pozostałe bomby na planszy i zamraża możliwość dalszej gry.

- czyWygrana():
- Metoda sprawdza, czy gracz wygrał, czyli czy odkrył wszystkie pola bez min. Jeśli tak, wyświetla okno z informacją o wygranej.

- odkrywanieFlaga(int x, int y):
- Metoda odpowiada za odkrywanie pól wokół kliknięte obszaru, gdy liczba flag wokół jest równa liczbie bomb sąsiadujących z danym polem.

Opis:

- Klasa Saper integruje logikę gry, obsługę interakcji z użytkownikiem, generowanie planszy oraz obsługę zdarzeń związanych z przyciskami. Współpracuje również z klasami OknoPrzegranej, OknoWygranej oraz PrzyciskReset, tworząc kompletną rozgrywkę w grze Saper.

5. Informacje na temat ilości pracy włożonej przez poszczególnych członków zespołu w tworzeniu projektu:

Mateusz Figiel: 50%

Kacper Giełżecki: 50%