GUI
Projekt 2

Do wyboru jeden temat z dwóch poniżej zamieszczonych

Temat 1 – Gra „Memory”

Napisz program będący grą w stylu “*memory*”. Gra polega na odnajdywaniu par takich samych kart. Gracz odsłania 2 karty. Jeśli są to takie same karty, zostają one widoczne na planszy, jeśli nie, karty będą zakrywane z powrotem (po 2 sekundach). Celem gracza jest zdjęcie wszystkich kart przy możliwie najszybszym czasie.

Należy zapewnić w pełni funkcjonalny interfejs graficzny. Konsola poleceń (*CLI*) może być jedynie pomocą informacyjną, ale nie może zachodzić tam żadna znacząca interakcja użytkownika z programem.

Program po uruchomieniu powinien wyświetlać menu główne składające się z opcji:

- *New Game* - nowa gra
- *High Scores* - tabela wyników
- *Exit* – wyjście

Po uruchomieniu nowej gry, gracz zostanie zapytany w osobnym oknie jakiej wielkości pole chce rozwiązać. Należy zauważyć, że sumaryczna ilość kart musi być ***parzysta***, aby każda karta miała swoją parę (na przykład w siatce 3x3 znajduje się 9 kart, więc po odkryciu 4 par jedna zostanie bez). W związku z tym należy zapewnić walidację rozmiarów planszy oraz wyświetlić stosowny ***błąd*** użytkownikowi.

Po uruchomieniu gry w nowym oknie wyświetlane są zakryte karty, a licznik czasu rusza (warto zauważyć, że licznik powinien być realizowany w osobnym ***wątku***, żeby nie blokować interakcji z oknem). Gra toczy się według wyżej wymienionych zasad do momentu odkrycia wszystkich kart. Należy zapewnić możliwość przerwania gry w dowolnym momencie poprzez wybrany przez Państwa złożony ***skrót klawiszowy*** (np. ***Ctrl+Shift+Q***), który spowoduje powrót do menu głównego.

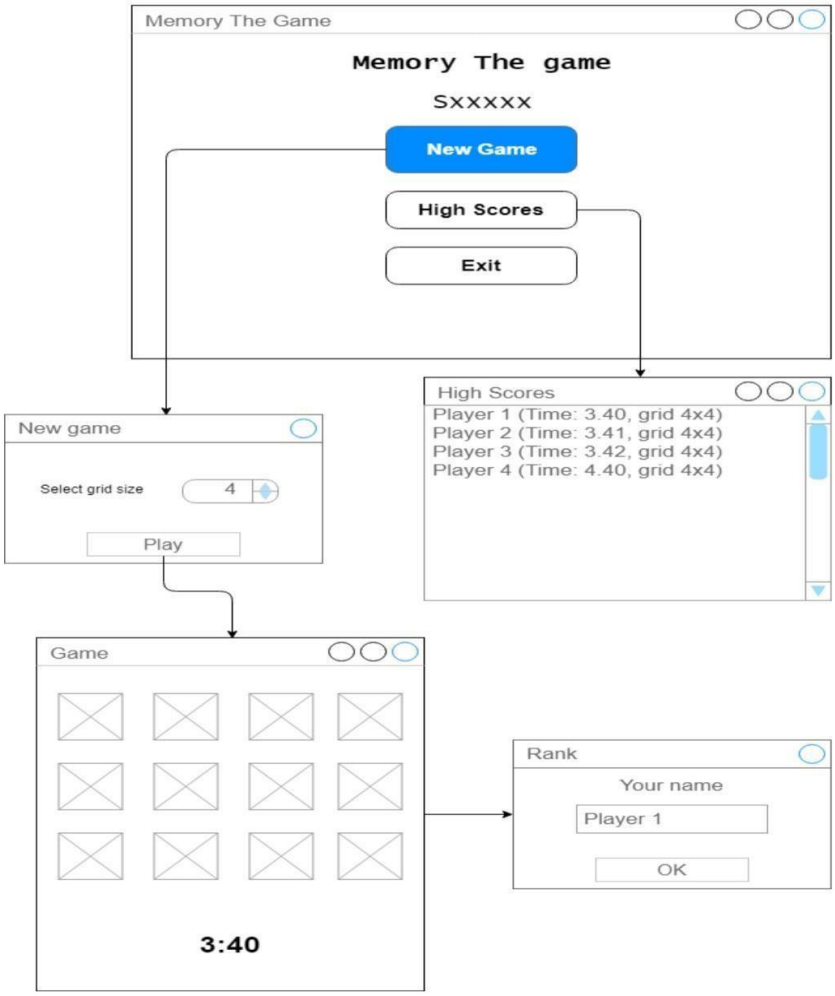
Po zakończeniu gry, w nowym oknie gracz proszony jest o swoją nazwę pod jaką ma być zapisywany w rankingu.

Dodatkową funkcjonalnością w grze, powinno być pamiętanie sytuacji w której użytkownik poprawnie odgadł parę kart, nie podglądając ich wcześniej. Ilość takich sytuacji powinna być uwzględniona w rankingu.

Ranking liczony jest na podstawie czasu i wielkości planszy (np. ***punkty = wielkość planszy/czas w sekundach***). W rankingu należy także dodać punkty zależnie od ilości poprawnie odgadniętych pierwszych par. Ranking powinien być posortowany malejąco względem liczby uzyskanych punktów. Należy zapewnić trwałość rankingu po ponownym uruchomieniu aplikacji, czyli należy go przechowywać w pliku na dysku. Postać przechowywanych danych jest sprawą drugorzędną i nie musi być czytelna dla gracza (można wykorzystać np. interfejs ***Serializable***).

Po wybraniu opcji rankingu z menu głównego, zostaje on wyświetlony użytkownikowi. Ponieważ okno rankingu może być relatywnie duże, dlatego należy zadbać o paski przewijania, w razie gdyby nie mieścił się on w oknie racjonalnych rozmiarów.

Przykładowa wizualizacja gry:



Wymagania:

- Zaimplementowanie głównego mechanizmu gry, opierającego się o wybór i odkrywanie kart oraz zapamiętywanie pierwszych odkrytych par – 5p.
- Zapewnienie skalowalności głównego okna z grą oraz pasek przewijania okna *Highscore* w wypadku wyświetlania wielu elementów – 2p.
- Umożliwienie wyboru wielkości planszy (siatki z kartami do odkrycia) oraz walidacja wyboru użytkownika – 3p.
- Zaimplementowanie mechanizmu liczącego upływ czasu – 3p.
- Po rozegranej partii umożliwienie graczowi wpisanie imienia i zapisanie go w *Highscore* wraz z uzyskanym wynikiem. Wyniki powinny być posortowane malejąco – 3p.
- Zapewnienie graficznego awersu i rewersu kart, awers powinien mieć przynajmniej klika (5) możliwości, aby skomplikować grę – 3p.
- Dane dotyczące *Highscore* powinny być przechowywane w pliku na dysku, zapisywane po skończeniu rozgrywki oraz wczytywane przy uruchomieniu gry. – 3p.
- Wykorzystanie wzorca MVC (wyjaśnienie jego składowych) oraz zastosowanie/obsługa wyjątków – 3p.

Aby projekt został poddany ocenie, musi zawierać minimalną funkcjonalność przypominającą grę. Należy zastosować w projekcie wzorzec projektowy MVC

Uwaga:

- ***Brak znajomości dowolnej linii kodu lub plagiat skutkować będzie wyzerowaniem punktacji za ten projekt.***
- ***W ocenie projektu poza praktyczną i merytoryczną poprawnością będzie brana również pod uwagę optymalność, jakość i czytelność napisanego przez Państwa kodu. (Rozwiązania nie optymalne, nie czytelne, bądź siłowe skutkują odjęciem punktów)***
- ***Ważną częścią projektu jest wykorzystanie między innymi: dziedziczenia, kolekcji, interfejsów lub klas abstrakcyjnych, lambda-wyrażeń, typów generycznych, dodatkowych funkcjonalności lub struktur oraz innych elementów charakterystycznych (ale tylko w naturalny sposób, nic na siłę)***

Temat 2 – Gra „Snake”

Napisz program będący grą w stylu “*snake*”. Gra polega na poruszaniu się wężem po mapie o *n* kolumnach i *m* wierszach. Wartości n i m są dowolne, ustalane podczas rozpoczynania rozgrywki, w związku z czym program musi być elastyczny i uniwersalny.

Wąż poruszając się po mapie zbiera punkty pojawiające się w losowych miejscach mapy (zaznaczone w wybrany przez Państwa sposób). Każdy zebrany punkt wydłuża długość naszego węża o 1. Dodatkowo istnieją punkty skracające, które są tak samo punktowane do ogólnego rankingu, ale mają funkcjonalność skrócenia węża o 1. Takie punkty powinny pojawiać się w stosunku co najwyżej 1:10 w stosunku do zwykłych punktów. Gdy wąż osiągnie długość 1 i zdobędzie punkt skracający długość, to dodane zostają punkty, jednak długość nie ulega zmianie. Gra trwa dopóki gracz nie uderzy wężem w ścianę lub samego siebie.

Wąż porusza się określoną przez Państwa prędkością (na przykład 1 ruch na 500 milisekund). Celem gracza jest zdobycie jak największej ilości punktów przy możliwie najszybszym czasie.

Należy zapewnić w pełni funkcjonalny interfejs graficzny. Konsola poleceń (*CLI*) może być jedynie pomocą informacyjną, ale nie może zachodzić tam żadna znacząca interakcja użytkownika z programem.

Program po uruchomieniu powinien wyświetlać menu główne składające się z opcji:

- *New Game* - nowa gra
- *High Scores* - tabela wyników
- *Exit* – wyjście

Po uruchomieniu nowej gry, gracz zostanie zapytany w osobnym oknie na jakiej wielkości planszy chce toczyć rozgrywkę. Po zatwierdzeniu rozmiarów, w nowym oknie wyświetlana jest plansza z początkowym wężem o długości 1, a licznik

czasu rusza (warto zauważyć, że licznik powinien być realizowany w osobnym ***wątku***, żeby nie blokować interakcji z oknem). Gra toczy się według wyżej wymienionych zasad. Należy zapewnić możliwość przerwania gry w dowolnym momencie poprzez wybrany przez Państwa złożony ***skrót klawiszowy*** (np. ***Ctrl+Shift+Q***), który spowoduje powrót do menu głównego.

Pozycja startowa węża jest dowolna, zaś nawigacja może się odbywać na przykład poprzez klawisze klawiatury, lub klikanie na przyciski wskazujące kierunek, utworzone przez Państwa w oknie nowej gry.

Pozycja głowy węża powinna być odróżnialna graficznie od reszty jego ciała.

Po zakończeniu gry, w nowym oknie gracz proszony jest o swoją nazwę pod jaką ma być zapisywany w rankingu.

Ranking liczony jest na podstawie zdobytych sumarycznie punktów i ułamka czas/wielkości planszy przemnożonego przez pewną wagę określającą istotność czasu w wyniku. Ranking powinien być posortowany malejąco względem uzyskanych punktów. Należy zapewnić trwałość rankingu po ponownym uruchomieniu aplikacji, czyli należy go przechowywać w pliku na dysku. Postać przechowywanych danych jest sprawą drugorzędną i nie musi być czytelna dla gracza (można wykorzystać np. interfejs ***Serializable***).

Po wybraniu opcji rankingu z menu głównego, zostaje on wyświetlony użytkownikowi. Ponieważ okno rankingu może być relatywnie duże, dlatego należy zadbać o paski przewijania, w razie gdyby nie mieścił się on w oknie racjonalnych rozmiarów.

Wymagania:

- Zaimplementowanie mechanizmu poruszania się węża oraz jego kontrolę – 5p.
- Generowanie się punktów na mapie i możliwość „zjedzenia” ich – 3p.
- Zapewnienie mechanizmu upływu czasu oraz określenie prędkości węża – 3p.
- Wybór wielkości planszy wraz z walidacją danych (np. ujemne wartości) – 3p.

- Zapewnienie skalowalności głównego okna z grą oraz pasek przewijania *Highscore* w wypadku wyświetlania wielu elementów – 2p.
- Po rozegranej partii umożliwienie graczowi wpisanie imienia i zapisanie go w Highscore wraz z uzyskanym wynikiem. Wyniki powinny być posortowane malejąco – 3p.
- Dane dotyczące Highscore powinny być przechowywane w pliku na dysku, zapisywane po skończeniu rozgrywki oraz wczytywane przy uruchomieniu gry. – 3p.
- Wykorzystanie wzorca MVC (wskazanie jego części składowych) oraz obsługa wyjątków – 3p.

Aby projekt został poddany ocenie, musi zawierać minimalną funkcjonalność przypominającą grę.

Należy zastosować w projekcie wzorzec projektowy MVC.

Uwaga:

- ***Zabrania się wykorzystywania narzędzi WYSIWYG do generowania okien (tzw. Window Builder`ów, Scene Builder`ów i innych).***
- ***Brak znajomości dowolnej linii kodu lub plagiat skutkować będzie wyzerowaniem punktacji za ten projekt.***
- ***W ocenie projektu poza praktyczną i merytoryczną poprawnością będzie brana również pod uwagę optymalność, jakość i czytelność napisanego przez Państwa kodu. (Rozwiązania nie optymalne, nie czytelne, bądź siłowe skutkują odjęciem punktów)***
- ***Ważną częścią projektu jest wykorzystanie między innymi: dziedziczenia, kolekcji, interfejsów lub klas abstrakcyjnych, lambda-wyrażeń, typów generycznych, dodatkowych funkcjonalności lub struktur oraz innych elementów charakterystycznych (ale tylko w naturalny sposób, nic na siłę)***