Wrocław, 24.04.2023 r.

Platformy programistyczne .NET i Java

Zadanie #2

Cel zadania

Wielowątkowa aplikacja okienkowa w technologii Java.

Informacje dodatkowe

- Prezentacja aplikacji na zajęciach.
- Aplikacja udostępniona w postaci repozytorium.
- Zadanie powinno być udokumentowane:
  - komentarze w aplikacji + wygenerowana dokumentacja,
  - 1-2 strony A4 podsumowania w formacie PDF,
  - diagram UML.

Terminy

- Oddanie repozytorium z zadaniem – **23 czerwca (piątek)**.

Wybrane technologie

- Język programowania Java
- Edytor kodu, np. *Eclipse/IBM Software Architect* lub *Netbeans*:
  - https://www.eclipse.org/downloads/packages/
  - https://netbeans.org
- System kontroli wersji *Git* wraz z repozytorium na platformie *GitLab* lub *GitHub*
  - https://git-scm.com
  - https://about.gitlab.com
  - https://github.com
- Dokumentacja, np. generator *Javadoc* od *Oracle*:
  - https://docs.oracle.com/javase/7/docs/technotes/tools/windows/javadoc.html
- Diagram UML (*Unified Modeling Language* – zunifikowany język modelowania), np. *UMLet*, *Draw.io* (online), *MS Visio* (licencja):
  - https://www.umlet.com
  - https://www.draw.io/
  - https://www.microsoft.com/en/microsoft-365/visio/flowchart-software

Wymagania

Aplikacja powinna spełniać podane wymagania:

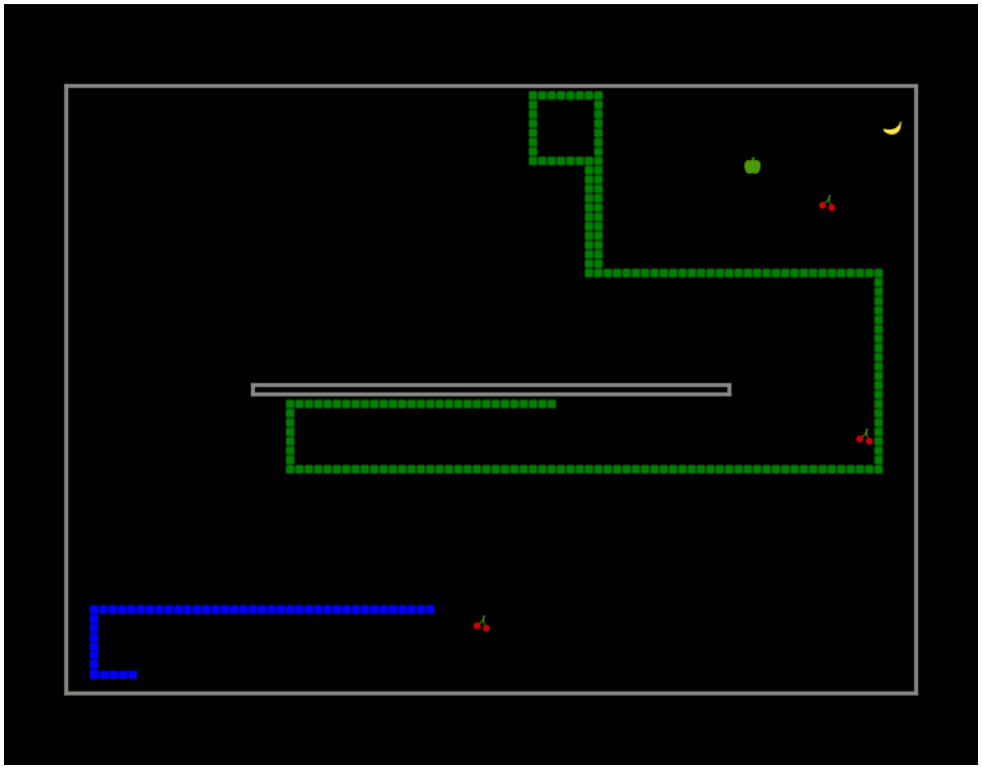
- Aplikacja napisana w języku programowania Java.
- Aplikacja okienkowa z interfejsem graficznym.
- Użycie biblioteki Swing (lub podobnej) do wyświetlania elementów graficznych.
- Aplikacja powinna zapewnić bazową oprawę graficzną typu „kolorowe piksele". Obiekty mogą być przedstawione w postaci prostych kwadratów, linii i kropek. *(**Notka:** za rozbudowaną szatę graficzną można otrzymać dodatkowe punkty, ale nie jest ona wymagana do pełnej oceny.)*
- Wykorzystanie klas i interfejsów.
- Wykorzystanie kolekcji w implementacji własnego algorytmu przetwarzania danych.
- Aplikacja wielowątkowa.
- Wątki powinny komunikować się/wchodzić w interakcje między sobą.
- Sterowanie wybranymi elementami aplikacji za pomocą sztucznej inteligencji (AI).
    - Inteligencja powinna być na takim poziomie by umożliwić podstawową rozgrywkę. Np. wąż powinien zbierać najbliższe owoce i unikać najbliższych przeszkód (nie jest konieczne planowanie najlepszej ścieżki dla wszystkich obiektów na planszy).
- Obsługa zapytań z klawiatury i myszy, np.:
    - Sterowanie elementami na planszy przy pomocy klawiatury.
    - Obsługa menu/okienka za pomocą myszy.
- Zapis i odczyt danych z plików (np. w postaci pliku tablicy najlepszych wyników).
- Wygenerowanie pliku z dokumentacją na podstawie komentarzy w kodzie.
- Udokumentowanie aplikacji przy pomocy diagramu UML. Diagram powinien przedstawiać przynajmniej kluczowe elementy aplikacji jak:
    - Uruchomienie aplikacji – metoda *main*,
    - Wykonanie głównej pętli programu,
    - Rysowanie elementów graficznych,
    - Aktualizacja stanu gry,
    - Odczyt danych wejścia/wyjścia (np. z klawiatury lub myszy).
- Dopuszczalne diagramy UML:
    - diagram klas (*Class Diagram*)
    - diagram aktywności (*Activity Diagram*)
    - diagram stanu (*State Machine Diagram*)

Przykładowe tematy

Gra typu „Snake”

- https://pl.wikipedia.org/wiki/Wąż_(gra_komputerowa)
- Obowiązują standardowe zasady gry.
- Gra 2-osobowa:
  - gracz #1 – sterowany przez użytkownika,
  - gracz #2 – sterowany przez AI.
- Losowo generowana plansza przy starcie gry:
  - plansza powinna zawierać stacjonarne przeszkody dla węża.
- Są dwa typy obiektów, które zbiera wąż:
  - stacjonarne (owoce) – określona pula generowana losowo na początku i uzupełniana na bieżąco w trakcie gry,
  - ruchome (żaba) – pojedynczy obiekt, który przemieszcza się po planszy i unika węża oraz pozostałych elementów (drugie AI).
- Program powinien obsługiwać przynajmniej 4 wątki: 2 węże, żabę i generator owoców.
- Obiekty na planszy nie powinny ze sobą kolidować, ani przenikać przez ściany.
- Zliczanie punktów i zapamiętanie najlepszych wyników.
- Obsługa „Game Over” i restartu gry.

(przykładowa grafika)



Gra typu „Pac-Man”

- https://pl.wikipedia.org/wiki/Pac-Man
- Obowiązują standardowe zasady gry (można pominąć „power-up” Pac-Man’a).
- Gra 1-osobowa – gracz steruje Pac-Man’em.
- Na planszy są 4 duchy, które przeszkadzają Pac-Man’owi:
    - każdy duch powinien być sterowany własnym algorytmem AI (https://pl.wikipedia.org/wiki/Pac-Man#Duchy)
    - Blinky (czerwony) – nieustępliwy, ciągle goni Pac-Man’a,
    - Pinky (różowy) – skrada się i atakuje z zasadzki,
    - Inky (niebieski) – nieprzewidywalny, losowo naśladuje zachowanie pozostałych duchów,
    - Clyde (pomarańczowy) – wałęsa się po planszy.
- Program powinien obsługiwać przynajmniej 4 wątki: Pac-Man’a i 3 z 4 duchów.
- Obiekty na planszy nie powinny ze sobą kolidować, ani przenikać przez ściany.
- Zliczanie punktów i zapamiętanie najlepszych wyników.
- Obsługa „Game Over” i restartu gry.

(przykładowa grafika)