Dokumentacja programu

Program posiada:

- 7 klas
- 1 interfejs
- 1 klasę abstrakcyjną

Klasa Main:

- Metoda main: Obsługuje grę, odbiera informacje od użytkownika za pomocą BufferedReader'a. Po wybraniu postaci wchodzimy w rozgrywkę. Petla while działa dopóki gracz nie zginie lub nie dotrzemy do skarbu.

Klasa Event:

- Zmienna BufferedReader reader: potrzebny do komunikacji z użytkownikiem
- Zmienna String temp: tymaczasowa zmienna do przetrzymywania treści jaką użytkownikiem przesłał BufferedReaderem
- Zmienna Random rng: Losowanie liczby

- Konstruktor Event(*Typ*, *Hero*): Konstruktor przenosi nas do konkretnej Metody obsługującej wydarzenie danego *Typu,* który może mieć inny przebieg w zależności od postaci (*Hero*) jaką gramy.
- Metoda Event1(Hero): Encounter z przeciwnikiem, w którym gracz może przegrać w zależności od podjętego wyboru
- Metoda Event2(Hero): Znalezienie skrzyni, gracz musi odpowiedź na zagadkę by otworzyć skrzynie
- Metoda Event3(): Znalezienie głównego skarbu
- Metoda Event4(Hero): Losowa pułapka w której gracz może zginąć jeśli podejmie złą decyzje

Klasa Mapa:

- Zmienna Random rng: Losowanie liczby
- Tablica dwuwymiarowa Map typu Int: Przedstawia mape
- Zmienne typu int pozycja_skarbu_i/j : Potrzebne w jednej z umiejetnosci
- Zmienne typu int i, j: Aktualna pozycja gracza

- Konstruktor Mapa(): Generuje losową mapę
- Gettery get_i, get_j: zwracają pozycję gracza, kolejno rząd i kolumnę
- Metody void Ruch_prawo/lewo/przod/tyl: przesuwają gracza kolejno: w prawo, w lewo, naprzód, w tyl
- Gettery get_room(), get_room(i, j): pierwszy zwraca wartość pokoju w którym znajduje się gracz, drugi zwraca wartość pokoju o danej pozycji i,j
- Gettery get_pozycjaskarbu_i(), get_pozycjaskarbu_j: zwracają rząd i kolumnę w której znajduje się skarb

- Metoda open_room(): otwiera zamknięty w pokoj w którym znajduje się gracz
- Metoda after_event(): ustawia typ pokoju na 5 (neutralny) po jakims evencie np. encounter z przeciwnikiem którego pokonal gracz
- Metoda Możliwe_drogi(): Wypisuje wszystkie możliwe ruchy jakie gracz może wykonać, np. gry gracz znajdzie się na lewym krancu mapy to nie może isc dalej w lewo, dlatego wyświetlą mu się tylko 3 możliwe kierunki.

//Strategy

Interfejs Postac:

- Metoda Perk(map): Wykonuje umiejętność unikalną dla każdej postaci, wykorzystuje mape
- Metoda IsAlive(): zwraca boolowską wartość informującą o tym czy bohater żyje
- Getter get_sila(): zwraca liczbe całkowitą reprezentującą siłe bohatera
- Getter get_zrecznosc(): zwraca liczbe całkowitą reprezentującą zrecznosc bohatera
- Getter get_uzycia(): (): zwraca liczbe całkowitą reprezentującą liczbe uzyc umiejetnosc bohatera
- Metoda set_alive(val): ustawia zmienna Alive na wartość val (bool)
- Metoda set_Sila(val): ustawia zmienna Sila na wartość val (int)
- Metoda set_Zrecznosc(val): ustawia zmienna Zrecznosc na wartość val (int)
- Metoda Add_Key(val): dodaje klucz o danej wartości
- Metoda Has_key(val): zwraca wartość boolowska true gdy bohater posiada klucz o danej wartości, wpp. zwraca false

Klasa Lara_Croft (implementacja inferfejsu Postac) :

- Zmienne sila, zrecznosc reprezentują statystyki
- Zmienna boolowska Alive informuje czy postac zyje
- Zmienna Uzycia_umiejetnosci informuje ile razy użytkownik może skorzystać z umiejetnosci bohatera
- Konstruktor Lara_Croft(): ustawia odpowiednio statystyki postaci itp.
- Implementacja Interfejsu Postac

Klasa Indiana_Jones (implementacja inferfejsu Postac) :

- Zmienne sila, zrecznosc reprezentują statystyki
- Zmienna boolowska Alive informuje czy postac zyje
- Zmienna Uzycia_umiejetnosci informuje ile razy użytkownik może skorzystać z umiejetnosci bohatera
- Konstruktor Indiana_Jones(): ustawia odpowiednio statystyki postaci itp.
- Implementacja Interfejsu Postac

//

//Template

Klasa abstrakcyjna Przeciwnicy:

- Zmienne Sila, Zrecznosc w postaci liczb całkowitych reprezentujące statystyki
- Gettery get_sila(), get_zrecznosc(): zwracają konkretne statystyki

- Metoda abstrakcyjna Atak(): wykonuje konkretny atak zależny jaki jest przeciwnik

Klasa Zlodziej_zPistoletem dziedzicząca po klasie Przeciwnicy:

- Trywialna implementacja metody Atak() opisująca sposób w jaki przeciwnik atakuje

Klasa Zlodziej_zNozem dziedzicząca po klasie Przeciwnicy:

- Trywialna implementacja metody Atak() opisująca sposób w jaki przeciwnik atakuje

//