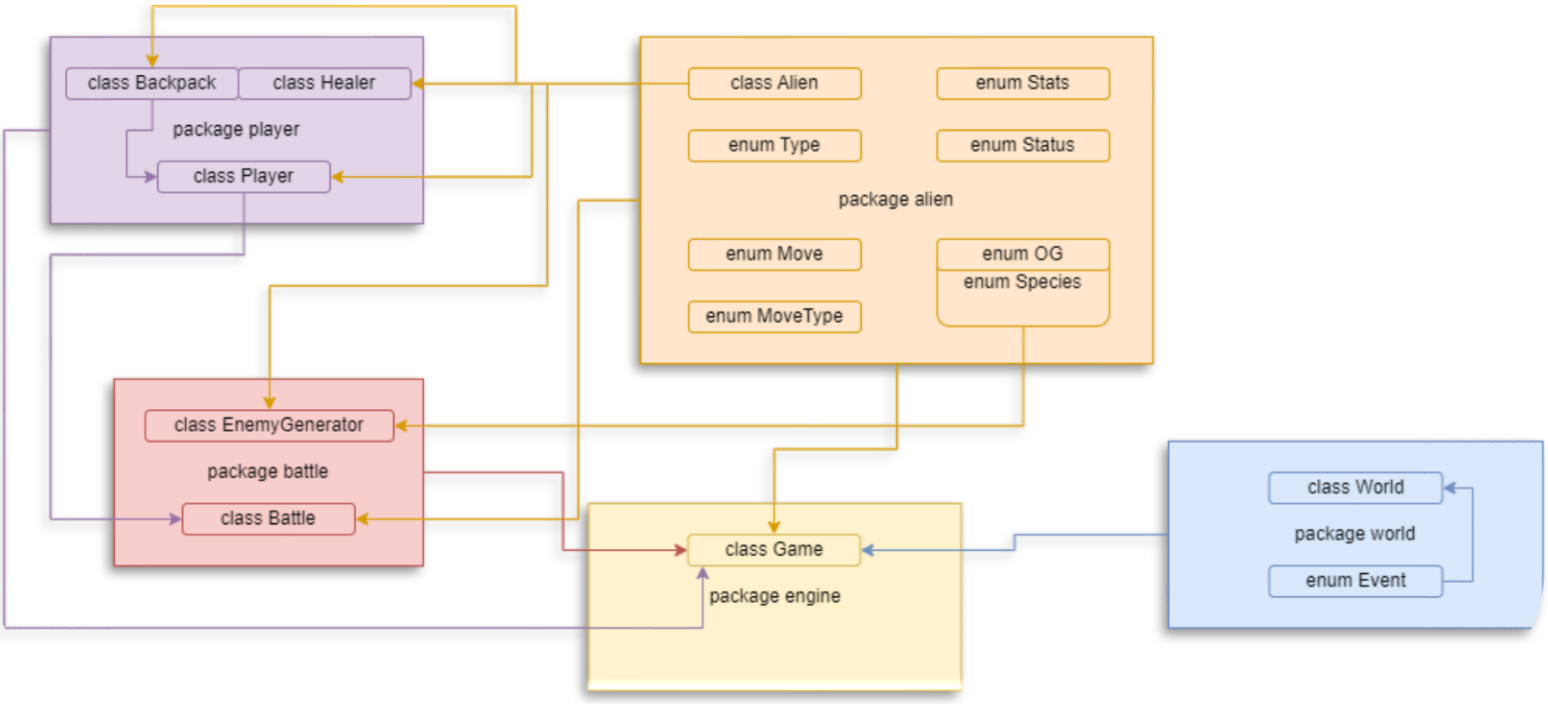

Projekt: Tekstowa gra inspirowana serią Ben 10

Gra tekstowa jest bazowana na świecie Ben 10, przenosi gracza do uniwersum serii, w którym wcielamy się w rolę Benjamina Tennysona, młodego bohatera posiadającego Omnitrix - tajemnicze urządzenie, które umożliwia mu przemianę w różne obce istoty.

Celem gry jest ochrona Ziemi przed inwazją kosmicznych złoczyńców, wykorzystując swoje umiejętności i narzędzia, aby rozwiązywać zagadki, pokonywać przeciwników i zbierać przedmioty potrzebne do kontynuacji misji.

Opis klas:

- package player
  - Klasa Player – klasa reprezentująca gracza w grze
    - inBattle (boolean): Określa, czy gracz jest aktualnie w trakcie walki.
    - money (int): Przechowuje ilość pieniędzy gracza.
    - NAME (String): Przechowuje imię gracza (stała wartość, niezmienialna).
    - backpack (Backpack): Obiekt klasy Backpack, przechowujący przedmioty zgromadzone przez gracza.
    - party (Alien[]): Tablica obiektów klasy Alien, przechowująca kosmitów w omnitrixie gracza.
    - Player(String name): Inicjalizuje obiekt gracza, ustawiając początkowe wartości atrybutów.
    - isInBattle(): Zwraca informację, czy gracz jest w trakcie walki.
    - getMoney(): Zwraca ilość pieniędzy gracza.
    - getName(): Zwraca imię gracza.
    - getBackpack(): Zwraca obiekt klasy Backpack, reprezentujący plecak gracza.
    - getParty(): Zwraca tablicę kosmitów w omnitrixie gracza.
    - setBattleState(boolean inBattle): Ustawia stan walki gracza.
    - isPartyEmpty(): Sprawdza, czy omnitrix gracza jest pusty.

        - addToParty(Alien alien): Dodaje obiekt klasy Alien do drużyny gracza.
        - swapSlots(int one, int two): Zamienia miejscami dwa kosmitów w omnitrixie gracza na podstawie podanych indeksów.
        - toString(): Zwraca tekstową reprezentację obiektu gracza.
        - + dodatkowe metody/obiekty dodane podczas pracy nad projektem
    - Klasa Healer - klasa pomocnicza odpowiedzialna za leczenie drużyny gracza
        - Healer(): Zapewnia, że klasa Healer nie będzie miała instancji, ponieważ zawiera tylko metody statyczne.
        - healParty(Player p): Metoda statyczna, która przyjmuje obiekt klasy Player i leczy wszystkich kosmitów w drużynie gracza.
        - + dodatkowe metody/obiekty dodane podczas pracy nad projektem
    - Klasa Backpack - klasa reprezentująca plecak gracza, przechowująca przedmioty.
        - items (Map): Mapa przechowująca przedmioty w plecaku, gdzie kluczem jest identyfikator przedmiotu, a wartością jest ilość tego przedmiotu.
        - Backpack(): Inicjalizuje obiekt plecaka.
        - getMap(): Zwraca mapę przedmiotów w plecaku.
        - + dodatkowe metody/obiekty dodane podczas pracy nad projektem
- package alien
    - Klasa Alien - klasa reprezentująca kosmitę w grze.
        - species (Species): Gatunek kosmity.
        - status (boolean[]): Tablica przechowująca statusy kosmity.
        - level (byte): Poziom kosmity.
        - totalExpForNextLevel (int): Całkowita liczba punktów doświadczenia potrzebna do osiągnięcia następnego poziomu.
        - totalExp (int): Całkowita liczba punktów doświadczenia kosmity.
        - currentStats (short[]): Tablica przechowująca obecne statystyki kosmity.
        - inBattleStats (short[]): Tablica przechowująca statystyki kosmity podczas walki.
        - moveSet (Move[]): Tablica przechowująca zestaw ruchów dostępnych dla kosmity.
        - Alien(Species species): Tworzy nowego kosmitę na poziomie 5 dla danego gatunku.
        - Alien(Species species, int level): Tworzy nowego kosmitę na podanym poziomie dla danego gatunku.
        - getStatus(): Zwraca tablicę statusów kosmity.
        - takeDamage(int damage): Odejmuje zadane obrażenia od punktów życia kosmity.
        - isFainted(): Sprawdza, czy kosmita został pokonany.
        - canLearnNewMove(): Sprawdza, czy kosmita może nauczyć się nowego ruchu.
        - initializeMoves(): Inicjalizuje zestaw ruchów kosmity na podstawie dostępnych ruchów w jego gatunku i poziomie.
        - getMoveSet(): Zwraca zestaw ruchów kosmity.
        - getName(): Zwraca imię kosmity.
        - getType(): Zwraca typ kosmity.
        - setLevel(int level): Ustawia poziom kosmity.

        - getLevel(): Zwraca poziom kosmity.
        - recalculateStats(): Przelicza statystyki kosmity na podstawie jego obecnego poziomu.
        - calculateStat(final Stats stat): Oblicza wartość konkretnej statystyki kosmity.
        - getInBattleStat(final Stats stat): Zwraca statystykę kosmity podczas walki.
        - getCurrentStat(final Stats stat): Zwraca obecną statystykę kosmity.
        - resetStats(): Resetuje statystyki kosmity (oprócz punktów życia) po walce.
        - getInBattleHp(): Zwraca obecne punkty życia kosmity.
        - addExp(int newExp): Dodaje punkty doświadczenia kosmity.
        - levelUp(): Zwiększa poziom kosmity i aktualizuje jego statystyki.
        - revive(): Przywraca kosmitę do życia po walce.
        - toString(): Zwraca tekstową reprezentację kosmity.
        - + dodatkowe metody/obiekty dodane podczas pracy nad projektem
    - Klasa Move - klasa reprezentująca ruch kosmity w grze.
        - NAME (String): Nazwa ruchu.
        - TYPE (Type): Typ ruchu.
        - POWER (short): Siła ruchu.
        - ACCURACY (byte): Celność ruchu.
        - MOVE_TYPE (MoveType): Typ ruchu (np. fizyczny, specjalny).
        - SIDE_EFFECT (Status): Efekt uboczny ruchu (null, jeśli brak).
        - STAT_LOWERED (Stats): Obniżona statystyka ruchu (null, jeśli brak).
        - PP (byte): Punkty ruchu.
        - battlePP (byte): Obecne punkty ruchu w walce.
        - battleAccuracy (byte): Obecna celność ruchu w walce.
        - Move(String name, Type type, MoveType m, byte PP, short power, byte accuracy): Tworzy nowy ruch z podanymi parametrami.
        - getStatusEffect(): Zwraca efekt uboczny ruchu.
        - getMoveType(): Zwraca typ ruchu.
        - getPower(): Zwraca siłę ruchu.
        - setPP(byte PP): Ustawia obecne punkty ruchu w walce.
        - downPP(): Zmniejsza obecne punkty ruchu w walce.
        - getCurrentPP(): Zwraca obecne punkty ruchu w walce.
        - getTotalPP(): Zwraca całkowitą liczbę punktów ruchu.
        - addPP(int PP): Dodaje punkty ruchu do obecnej liczby w walce.
        - resetAccuracy(): Resetuje obecną celność ruchu.
        - resetPP(): Resetuje obecne punkty ruchu do pierwotnej wartości.
        - getName(): Zwraca nazwę ruchu.
        - getType(): Zwraca typ ruchu.
        - getAccuracy(): Zwraca celność ruchu.
        - toString(): Zwraca tekstową reprezentację ruchu
        - + dodatkowe metody/obiekty dodane podczas pracy nad projektem
    - Enumeracja MoveType - enumeracja reprezentująca typy ataków używanych przez kosmitów w grze.

        - PHYSICAL: Reprezentuje ataki fizyczne, które opierają się na sile fizycznej kosmity.
        - SPECIAL: Reprezentuje ataki specjalne, które opierają się na mocy specjalnej kosmity.
        - STATUS: Reprezentuje ataki statusowe, które mają na celu wpływanie na status przeciwnika (np. otrucie, paraliż).
        - + dodatkowe metody/obiekty dodane podczas pracy nad projektem
    - Klasa Species - klasa reprezentująca różne gatunki kosmitów w grze.
        - NAME (typ: String): Nazwa kosmity.
        - INDEXNUMBER (typ: String): Numer indeksu kosmity.
        - LEARNSET (typ: Map<Move, Integer>): Mapa ruchów, które kosmita może opanować wraz z poziomem wymaganym do nauczenia danego ruchu.
        - BASE_STATS (typ: short[]): Tablica przechowująca bazowe statystyki kosmity.
        - TYPE (typ: Type[]): Tablica przechowująca typy kosmity.
        - getLearnset(): Zwraca mapę ruchów, które kosmita może opanować.
        - calculateExp(int level): Oblicza wartość doświadczenia kosmity na podstawie podanego poziomu.
        - getBaseStat(int stat): Zwraca bazową wartość wybranej statystyki kosmity.
        - getName(): Zwraca nazwę kosmity.
        - getType(): Zwraca tablicę zawierającą typy kosmity.
        - + dodatkowe metody/obiekty dodane podczas pracy nad projektem
    - Enumeracja Stats - wyliczenie reprezentujące statystyki kosmitów w grze.
        - HP: Reprezentuje punkty życia kosmity.
        - ATTACK: Reprezentuje statystykę ataku kosmity.
        - DEFENSE: Reprezentuje statystykę obrony kosmity.
        - SP_ATTACK: Reprezentuje statystykę specjalnego ataku kosmity.
        - SP_DEFENSE: Reprezentuje statystykę specjalnej obrony kosmity.
        - SPEED: Reprezentuje statystykę prędkości kosmity.
        - + dodatkowe metody/obiekty dodane podczas pracy nad projektem
    - Enumeracja Status - wyliczenie reprezentujące statusy, które mogą wpływać na kosmitów w grze i modyfikować ich statystyki.
        - BURN: Reprezentuje status "oparzenie", który może obniżać statystykę ataku kosmity.
        - CONFUSION: Reprezentuje status "zamieszanie", który może sprawić, że kosmita może wykonać niezamierzone działania.
        - FREEZE: Reprezentuje status "zamrożenie", który może uniemożliwić kosmicie wykonywanie ruchów.
        - PARALYZE: Reprezentuje status "paraliż", który może zmniejszyć szanse kosmity na wykonanie ruchu.
        - POISON: Reprezentuje status "otrucie", który może zadawać dodatkowe obrażenia kosmicie co turę.
        - SEED: Reprezentuje status "nasienie", który może powodować odnawianie się punktów życia kosmity co turę.

        - SLEEP: Reprezentuje status "sen", który może sprawić, że kosmity jest nieaktywny przez kilka tur.
        - + dodatkowe metody/obiekty dodane podczas pracy nad projektem
    - Enumeracja Type - wyliczenie reprezentujące różne typy kosmitów w grze oraz ich przewagi i słabości.
        - Wyliczone typy: Reprezentuje typ, na co jest superefektywny, niezbyt efektywny i na co nie ma efektu.
        - isSuperEffectiveAgainst(Type t): Metoda zwracająca wartość logiczną, czy dany typ jest superefektywny względem podanego typu.
        - isNotVeryEffectiveAgainst(Type t): Metoda zwracająca wartość logiczną, czy dany typ jest niezbyt efektywny względem podanego typu.
        - hasNoEffectOn(Type t): Metoda zwracająca wartość logiczną, czy dany typ nie ma efektu względem podanego typu.
        - + dodatkowe metody/obiekty dodane podczas pracy nad projektem
- package battle
    - Klasa Battle - reprezentuje pojedynki pomiędzy graczem a kosmitami w grze.
        - player1: Obiekt klasy Player, reprezentujący gracza biorącego udział w walce.
        - opponent: Obiekt klasy Alien, reprezentujący kosmitę walczącego przeciwko graczowi.
        - player: Obiekt klasy Alien, reprezentujący aktualnie aktywnego kosmitę gracza.
        - alienSlot: Numer slotu, w którym znajduje się kosmita gracza w omnitrixie.
        - running: Flaga określająca, czy walka jest w trakcie.
        - Battle(Player p, Alien opponent): Inicjalizuje obiekt Battle przyjmując obiekt gracza i kosmitę przeciwnika jako argumenty.
        - useMove(Move m): Wykorzystuje wybrany ruch przez gracza. Wybiera losowy ruch dla przeciwnika i wykonuje odpowiednie działania, takie jak obliczanie obrażeń, sprawdzanie trafień i skuteczności typów, aktualizacja punktów zdrowia itp.
        - setRunning(boolean running): Ustawia stan walki na podstawie przekazanego argumentu running.
        - isRunning(): Sprawdza, czy walka nadal trwa, na podstawie stanu running oraz punktów zdrowia gracza i kosmity przeciwnika.
        - toString(): Przesłonięcie metody toString(), zwracające informacje o graczu i kosmicie przeciwnika.
        - + dodatkowe metody/obiekty dodane podczas pracy nad projektem
    - Klasa EnemyGenerator - generuje losowe kosmity do walki z graczem.
        - generateAlien(): Generuje losowego kosmitę jako obiekt klasy Alien. Wykorzystuje klasę Species, która zawiera listę dostępnych gatunków kosmitów. Wybiera losowy gatunek kosmity z tej listy i tworzy nowy obiekt Alien z wybranym gatunkiem
        - + dodatkowe metody/obiekty dodane podczas pracy nad projektem
- package engine
    - Klasa Game - reprezentuje główną logikę gry.

        - main(String[] args): Metoda główna programu, rozpoczynająca działanie gry. Wykorzystuje klasę Scanner do odczytywania danych wprowadzanych przez użytkownika z konsoli. Zawiera logikę gry, taką jak wybór kosmity, interakcje z graczem (np. walka, leczenie), obsługę opcji wyboru itp.
        - + dodatkowe metody/obiekty dodane podczas pracy nad projektem
- package world
    - Klasa World - reprezentuje świat gry, w którym gracz może eksplorować różne obszary, napotykać wydarzenia i prowadzić walki.
        - World(Player player): Tworzy nowy obiekt świata na podstawie podanego gracza.
        - explore(): Implementuje logikę eksploracji świata, gdzie gracz może poruszać się po różnych regionach, napotykać na wydarzenia itp.
        - handleEvent(Event event): Obsługuje pojedyncze wydarzenie w świecie gry. Może to obejmować interakcję z postaciami, znalezienie przedmiotu, rozpoczęcie walki itp.
        - startBattle(Alien opponent): Rozpoczyna walkę między graczem a przeciwnikiem (przekazanym jako parametr). Tworzy obiekt Battle i uruchamia logikę walki.
        - + dodatkowe metody/obiekty dodane podczas pracy nad projektem
    - Klasa Event - reprezentuje różne wydarzenia występujące w grze. Każde wydarzenie posiada przypisany opis, który opisuje charakterystykę danego wydarzenia. Przykładowe wydarzenia w klasie Event:
        - FIND_ITEM (Znalezienie przedmiotu): Wydarzenie, w którym gracz odnajduje cenny przedmiot lub skarb podczas eksploracji świata gry.
        - TALK_TO_NPC (Rozmowa z NPC): Wydarzenie, w którym gracz wchodzi w interakcję z niezależnymi postaciami (NPC) i prowadzi dialog w celu uzyskania informacji, zadań lub nagród.
        - EXPLORE_AREA (Eksploracja obszaru): Wydarzenie, w którym gracz odkrywa nowe obszary świata gry, badając tajemnicze miejsca, ukryte lokacje lub nieznane regiony.
        - COMPLETE_QUEST (Ukończenie zadania): Wydarzenie, w którym gracz wykonuje określone zadanie lub misję i otrzymuje nagrodę lub spełnia pewne warunki postęp w fabule gry.
        - + dodatkowe metody/obiekty dodane podczas pracy nad projektem