Podstawowe aspekty programowania.

C#

Wydział Matematyki i Informatyki
Ćwiczenia: Michał Mnich

NFO.

Poniższe ćwiczenia mają na celu zapoznanie się z podstawowymi aspektami języka C# oraz frameworku .net, które zostały omówione na wykładzie. Poniżej znajduje się zestaw trzech zadań polegających na zaprojektowaniu mechanik prostej tekstowej gry.

Do pracy będzie wam potrzebna biblioteka:
Namespace: System
Assembly: System.Console.dll
https://docs.microsoft.com/en-us/dotnet/api/system.console?view=netcore-3.1

Prosze także o zapoznanie się bądź przypomnienie sobie ostatnich materiałów podanych na wykładzie. Na wszystkie zadania w tym dokumencie przewidziany jest jeden tydzien.

ZADANIE 1 – MENU GŁÓWNE I NOWA GRA – 2PKT

Po uruchomieniu gry proszę pokazać menu główne z dostępnymi opcjami:

```
Witaj w grze <Tutaj nazwa gry>
[1] Zacznij nową grę
[X] Zamknij program
```

Jeżeli wybierzemy nieprawidłową opcję program powinien nas poinformować, że wybrana opcja jest niewłaściwa i dać nam szansę ponownego wyboru.

W przypadku wybrania opcji [1]:

1. Ekran powinien się wyczyścić.
2. Użytkownik powinien zostać poproszony o nazwę bohatera:
    a. Należy usunąć z wprowadzonego tekstu niepotrzebne białe znaki.
    b. Należy zapewnić sensowną walidację nazwy, czyli na przykład:
        i. tylko znaki alfabetu,
        ii. co najmniej 2 niepuste znaki.
    c. W przypadku podania niepoprawnego imienia należy powiadomić o tym użytkownika i dać mu szansę ponownego wpisania.
3. Użytkownik powinien zostać poproszony o klasę bohatera – klasa bohatera powinna być typem wyliczanym(enumem `EHeroClass`):
    a. Wybór z listy:
        i. barbarzyńca,
        ii. paladyn,
        iii. amazonka
    b. W przypadku nieprawidłowego wyboru – komunikat i możliwość ponownego wyboru
    c. Przy okazji wyboru klasy należy potwierdzić nazwę użytkownika – coś w stylu „Witaj Jimmy Page, wybierz klasę bohatera:”

W efekcie ma zostać utworzony bohater – obiekt klasy Hero – z wypełnionym imieniem oraz klasą bohatera

4. Ekran powinien się wyczyścić po czym powinien się pojawić odpowiedni komunikat rozpoczynający grę na przykład: „Barbarzyńca Robert Plant wyrusza na przygodę”.
5. Po kliknięciu dowolnego przycisku powinno nastąpić zamknięcie programu.

W przypadku wyboru opcji [X]:

1. Powinno nastąpić zamknięcie programu.

Zadanie 2 – Rozmowa z ludźmi – 3pkt.

W wyniku realizacji poprzedniego zadania mamy stworzone menu główne gry i ekran kreowania postaci. Naszym kolejnym zdaniem będzie możliwość rozmawiania z ludźmi w miasteczku, do którego trafił nasz bohater.

Przede wszystkim, aby prowadzić rozmowę zazwyczaj fajnie jest mieć z kim rozmawiać dlatego zacznijmy od stworzenia klasy `NonPlayerCharacter` – w skrócie NPC. Niech NPC ma imię.

Do realizacji naszego dialogu będziemy potrzebować stworzyć dwie klasy:

- `NpcDialogPart` – klasa zawiera w sobie treść wypowiedzi oraz kolekcję `HeroDialogPart`ów, które są możliwe
- `HeroDialogPart` – klasa zawiera w sobie treść wypowiedzi bohatera oraz `NpcDialogPart`, do którego prowadzi

Przykładowa rozmowa wyglądałaby tak:

NPC: „Witaj, czy możesz mi pomóc dostać się do innego miasta?”
Hero: „Tak, chętnie pomogę”
NPC: „Dziękuję! W nagrodę otrzymasz ode mnie 100 sztuk złota”.
Hero: „Dam znać jak będę gotowy” <KONIEC>
Hero: „100 sztuk złota to za mało!”
NPC: „Niestety nie mam więcej. Jestem bardzo biedny”.
Hero: „OK, może być 100 sztuk złota.”
NPC: „Dziękuję.” <KONIEC>
Hero: „W takim razie radź sobie sam.” <KONIEC>
Hero: „Nie, nie pomogę, żegnaj.” <KONIEC>

Załóżmy, że nasze rozmowy zawsze zaczynają się od wypowiedzi NPC natomiast kończyć mogą się zarówno wypowiedziami NPCów jak i wypowiedziami bohatera.

Do naszej klasy NPC dopiszmy dodatkowo metodę:

```
NpcDialogPart StartTalking();
```

Warto by było jeszcze stworzyć miejsce, w którym NPC się znajduje. Stwórzmy więc klasę `Location`, która zawiera w sobie listę NPCów – niech w tym zadaniu będzie ich dwóch. Niech każdy z NPCów ma jakąś ścieżkę dialogową(jeden może mieć przykładową, drugiemu proszę coś wymyślić – może być na przykład kupowanie przedmiotów) oraz swoje imię[1]. Dodatkowo niech lokacja w grze ma swoją nazwę[2]

Wróćmy do wyniku zadania 1. Tuż po rozpoczęciu gry następowało jej zamknięcie. Proszę zmodyfikować grę tak, aby po jej rozpoczęciu tworzyła się nowa lokalizacja i pokazywało się menu:

```
Znajdujesz się w : {NazwaLokalizacji}. Co chcesz zrobić?
```

[1] Jeżeli nie chce się Państwu wymyślać to tutaj są na przykład nazwy NPCów z Diablo 2: Akara, Charsi, Deckard Cain, Gheed, Kashya, Warriv
[2] Nazwy lokacji z Forgotten Realms, jeżeli nie chce się Państwu wymyślać: Calimport, Neverwinter, Silverymoon.

```
[1] Porozmawiaj z {NazwaNPC1}”
[2] Porozmawiaj z {NazwaNPC2}”
[X] Zamknij program
```

Proszę napisać metodę wyświetlającą menu Lokalizacji w ten sposób, żeby nie trzeba było nic w niej modyfikować, jeżeli na przykład dojdą nowi NPCe, zmienią się ich imiona lub zmieni się jej nazwa. Jej deklaracja powinna wyglądać mniej więcej tak:

```
void ShowLocation(Location location);
```

Podobnie jak poprzednio, jeżeli wybrana opcja jest nieprawidłowa, powinniśmy wyświetlić użytkownikowi komunikat i poprosić o jej poprawę. Na początku każdego dialogu proszę czyścić ekran. Proszę odpowiednio zaimplementować wybór każdej z opcji i mechanizm dialogów. Po zakończeniu dialogu proszę wrócić do menu lokalizacji.

Proszę całą logikę przeprowadzania dialogu wydzielić do osobnej metody:

```
void TalkTo(NonPlayerCharacter npc);
```

ZMIENNE W DIALOGACH

Wyobraźmy sobie, że w naszej grze chcielibyśmy umieścić taki dialog:

```
NPC: „Hej czy to Ty jesteś tym słynnym {ImięPostaci} – pogromcą smoków?”
        Hero: „Tak, jestem {ImięPostaci}”
        NPC: „WOW! Miło poznać!”. <KONIEC>
        Hero: „Nie.” <KONIEC>
```

W tym celu zakodujmy w naszym dialogu zmienne. Póki co wprowadźmy tylko jedną dla imienia użytkownika. Niech zmienna wygląda tak #HERONAME#.

Napiszmy klasę `DialogParser`, niech przyjmuje ona w konstruktorze naszego bohatera. Stwórzmy również interfejs `IDialogPart`, który będzie implementowany przez `NpcDialogPart`

i `HeroDialogPart`. Niech zawiera on stringowy getter treści wypowiedzi.

Niech klasa `DialogParser` ma metodę:

```
string ParseDialog(IDialogPart)
```

Proszę zmodyfikować metodę odpowiedzialną za przeprowadzanie dialogów, aby przyjmowała dodatkowo obiekt `IDialogParser` oraz używała go do parsowania wypowiedzi.