2. Beadandó feladat dokumentáció

Készítette:
Zsolnai Martin
E-mail: tf94ll@inf.elte.hu

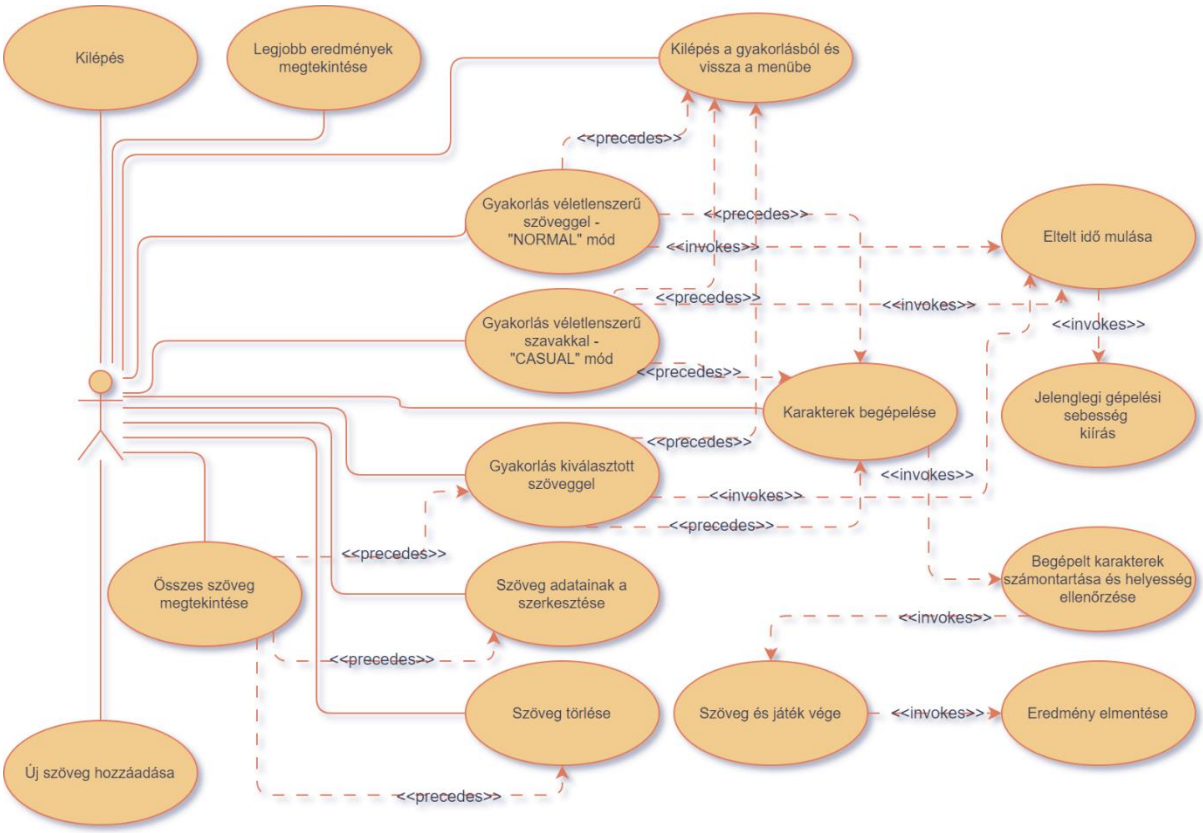
Feladat:

Készítsünk egy programot, amely segíti a gyorsgépelés gyakorlását, és a felhasználók sebességének és pontosságának fejlesztésére összpontosít. Ennek a programnak a neve Typoetry (a type és poetry szóból) és a felhasználók különféle irodalmi műveket, például verseket és idézeteket gépelhetnek le, miközben teljesítményüket WPM-ben (Words Per Minute) mérjük. A cél az, hogy szórakoztató és tanulságos módon lehetőséget nyújtsunk a gyorsírás gyakorlására, miközben különféle irodalmi szövegeket olvashatnak és tanulhatnak róluk. A képernyőn megjelenő szöveg írása során a felhasználók azonnali visszajelzést kapnak arról, hogy helyesen vagy helytelenül írták-e be a betűt, a háttér színének zöldre vagy pirosra váltásával. Emellett lehetőséget kell biztosítani arra is, hogy a felhasználók saját szövegeket adhassanak meg gyakorlásra, amelyeket el is tudnak menteni. Fontos, hogy az eddigi eredményeiket is vissza tudják nézni. Továbbá, ha a felhasználók nem szeretnének hosszabb irodalmi szövegeket gépelni, lehetőséget kell adni az egyszerű szavakkal való gyakorlásra is.

Elemzés:

- Három különböző módon gyakorolhatunk a programmal: a NORMAL módban egy véletlenszerűen kiválasztott szöveget gépelhetünk le, vagy lehetőségünk van egy olyan szöveg választására is, amellyel gyakorolni szeretnénk. Emellett elérhető a CASUAL mód is, amelyben az 500 leggyakrabban használt angol szóból véletlenszerűen kiválasztott 20 szót gépelhetünk le.
- A program fő működését több ablakos asztali alkalmazásként WPF grafikus felülettel valósítjuk meg, azonban az egyszerűbb kezelés érdekében a szövegek szerkesztése egy külön felugró ablakban történik.
- Az ablak közepén el van helyezve egy címke, ami az eddigi eredményinket mutatja és ez alatt van négy nyomógomb, amelyen különböző menü opciók közül választhatunk (NORMAL és CASUAL módok mellett még megtalálható a CHOOSE TEXT és az ADD TEXT opciók).

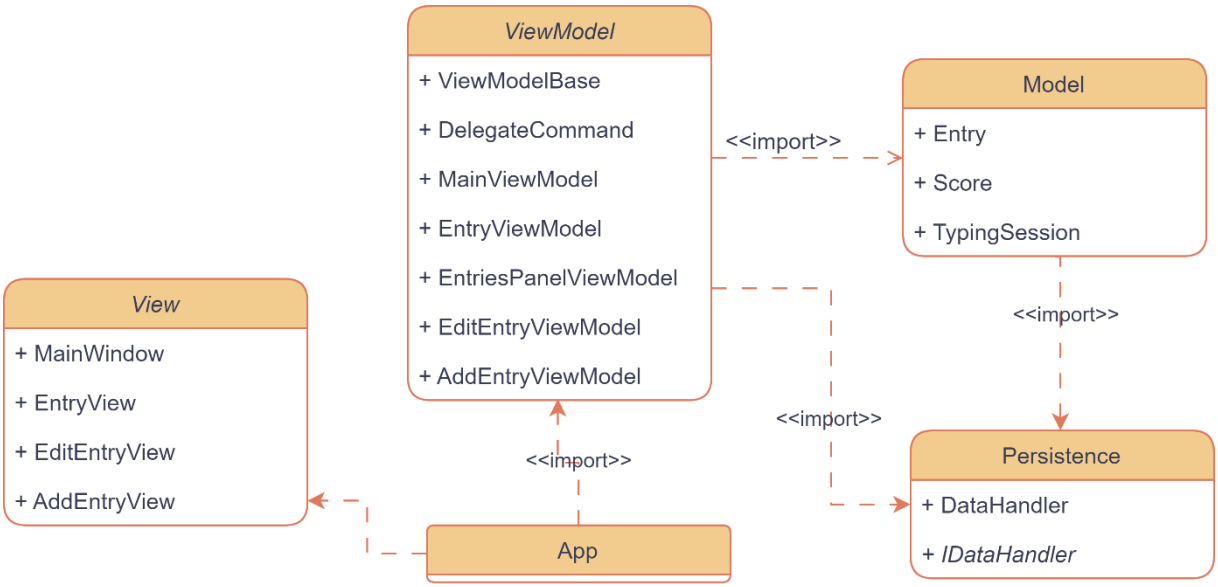
- Bármelyik gyakorlási mód kiválasztása után az ablak közepén megjelenik egy három másodperces időzítő, ami után a begépelendő szöveg jelenik meg egy szövegdobozban, amelyet szerkeszteni nem tudunk, viszont a bemeneti billentyűkre reagál és visszajelez, ha jó vagy rossz betűt írunk be. A szöveg írása közben a jobb alsó sarokban láthatjuk az eddigi teljesítményünket, illetve a bal alsó sarokban lévő gombbal vissza tudunk lépni a menübe.
- Ha végeztünk a gépeléssel, akkor egy dialógusablakban láthatjuk a végső eredményt, ami után a program azt el is menti.
- A CHOOSE TEXT gomb megnyomása után megjelenik az összes eddig elmentett szöveg. Itt lehetőség nyílik egy adott szöveggel való gyakorlásra, a szöveg szerkesztésére, valamint annak törlésére is.
- Szerkesztés esetén egy új ablak nyílik meg, ahol szöveges dobozok használatával beírhatjuk a frissített adatokat.
- Ha törölni szeretnénk, akkor még egy dialógusablakban is el kell fogadnunk, hogy törölni akarjuk az adott szöveget.
- Az ADD TEXT kiválasztása után egy a szöveg szerkesztésével megegyező felületet láthatunk (ez azonban nem nyit meg új ablakot), ahol a mezők kitöltése után el tudjuk menteni az általunk megadott szöveget.
- A felhasználói esetek az 1. ábrán láthatóak.



1. ábra: Felhasználói esetek diagramja

Tervezés:

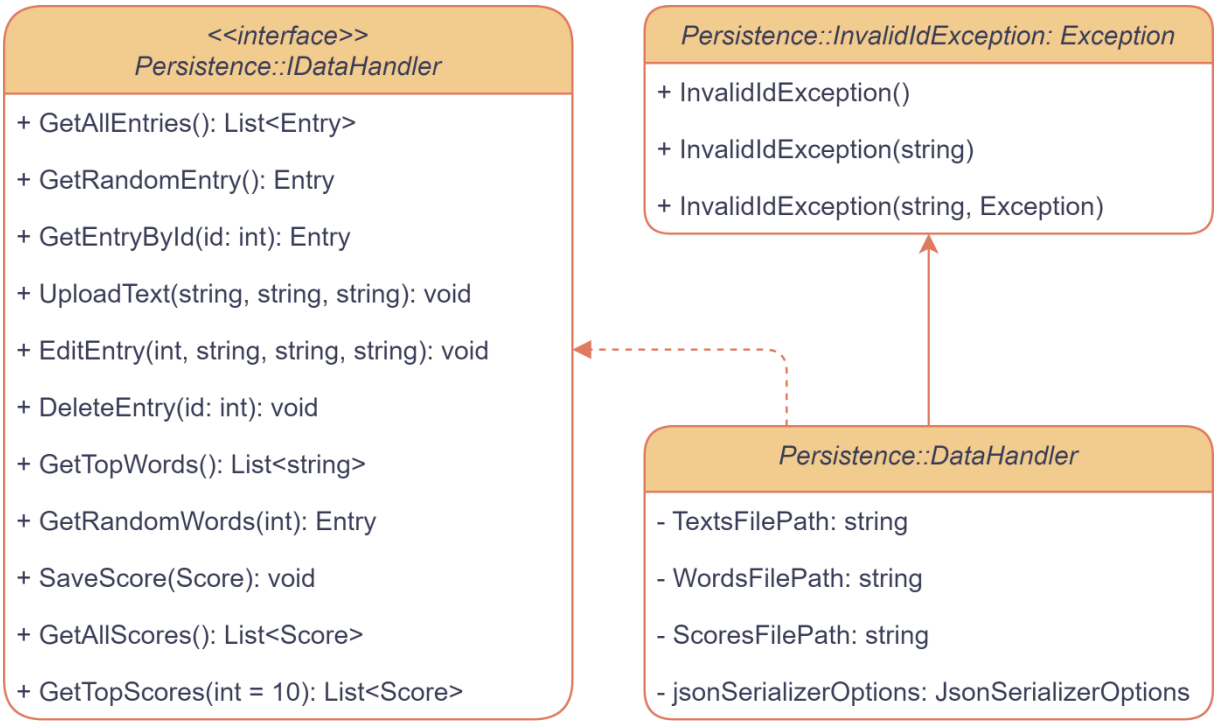
- Programszerkezet:
  - A programot MVVM architektúrában valósítjuk meg, ennek megfelelően **View**, **Model**, **ViewModel** és **Persistence** névtereket valósítunk meg az alkalmazáson belül. A program környezetét az alkalmazás osztály (App) végzi. A program csomagszerkezete a 2. ábrán látható.
  - A program szerkezetét két projektre osztjuk implementációs megfontolásból: a **Persistence** és **Model** csomagok a program felületfüggetlen projektjében, míg a **ViewModel** és **View** csomagok a WPF függő projektjében kap helyet.



2. ábra: Az alkalmazás csomagdiagramja

- Perzisztencia (3. ábra):
  - Az adatkezelés feladata a szövegek és hozzá tartozó információk, illetve az elért eredmények tárolása, betöltése és törlése. Ezeket a funkciókat a **DataHandler** osztály biztosítja.
  - Szövegek kezeléséhez több funkció is segít. **GetAllEntries()**: Az összes szöveg betöltése. **GetRandomEntry()**: Véletlenszerű szöveg kiválasztása és betöltése. **GetEntryById(int id**): Szöveg lekérdezése azonosító alapján. **UploadText(string title, string author, string text)**: Új szöveg mentése. **EditEntry(int id, string newTitle, string newAuthor, string newText)**: Szöveg szerkesztése. **DeleteEntry(int id)**: Szöveg törlése.
  - Szavak lekéréséhez: **GetRandomWords(int numberOfWords)**: Adott számú szó kiválasztása véletlenszerűen. **GetTopWords()**: Összes szó betöltése, ezt a modelben nem használjuk, viszont a **GetRandomWords** függvénynek szüksége van rá.
  - Eredmények kezelése: **SaveScore(Score score)**: Eredmény mentése. **GetAllScores()**, **GetTopScores(int top = 10)**: Eredmények betöltése és ranglista létrehozása.
  - Adattárolás: Az osztály JSON és szöveges fájlokkal dolgozik, alapértelmezésben a következő fájlokkal: **texts.json**: Mentett

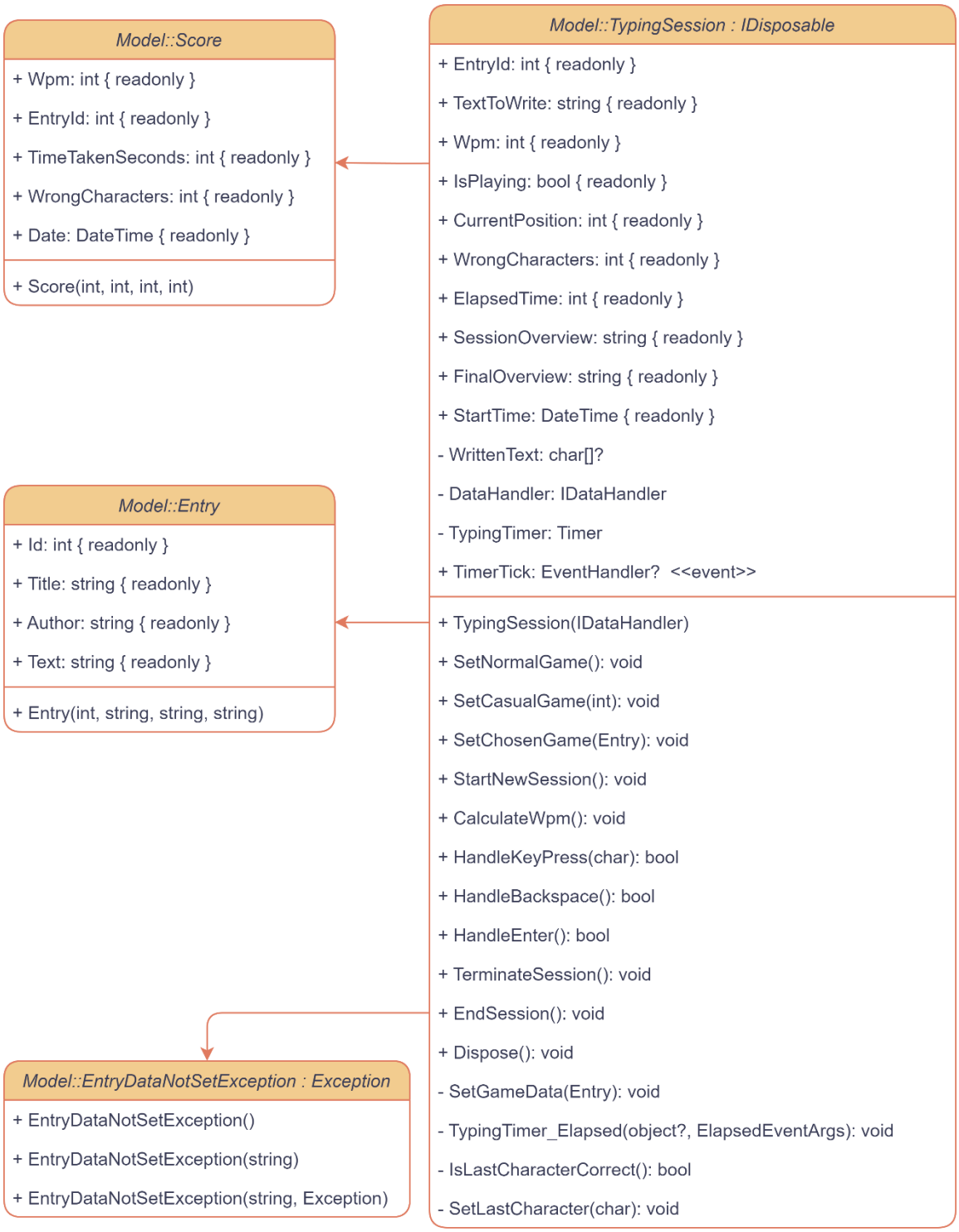
szövegek, **top500words.txt**: A szavakat tartalmazza, **scores.json**: Eredmények.



3. ábra: A Persistence csomag osztálydiagramja

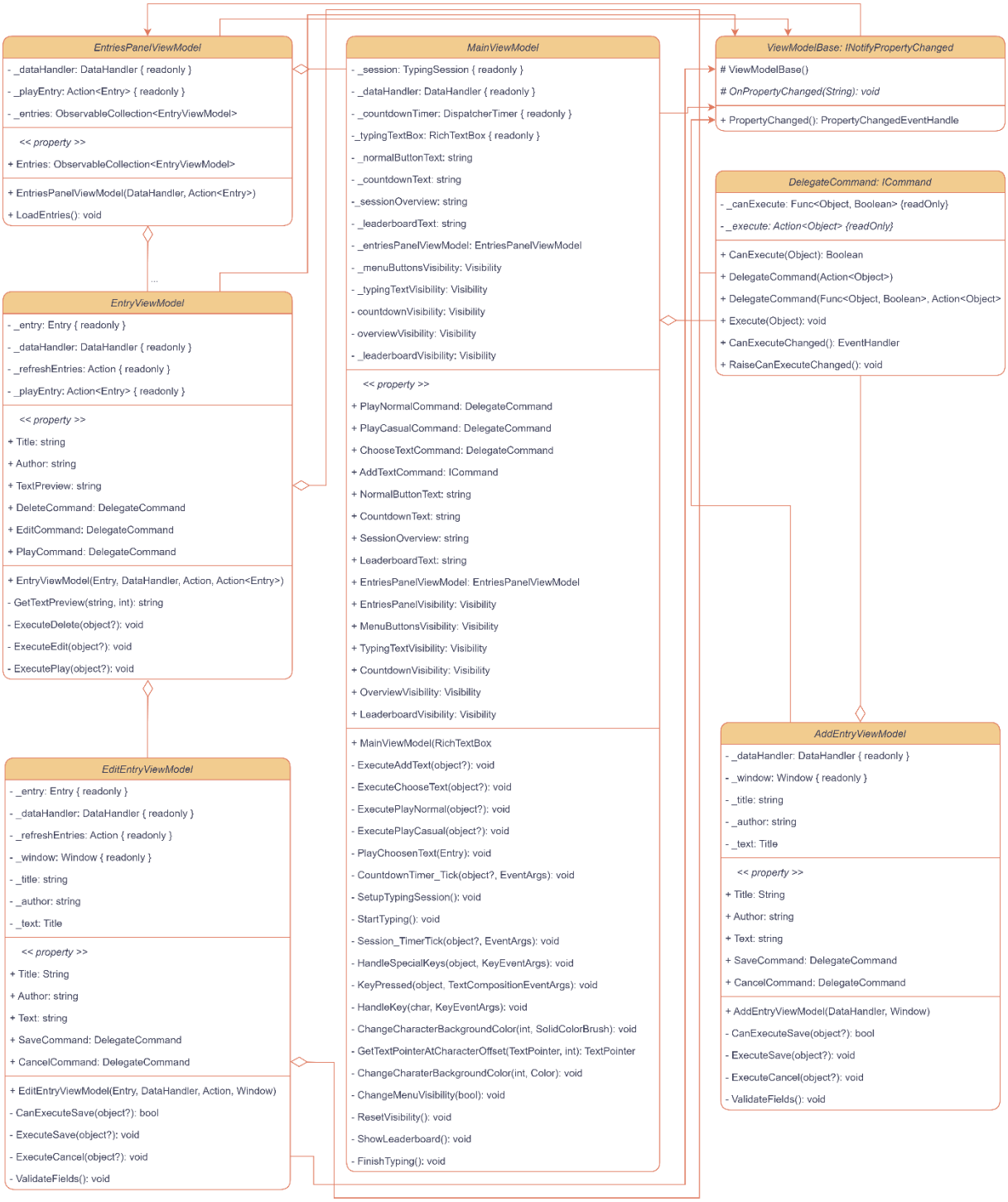
- Modell (4. ábra):
  - A **TypingSession** osztály kezeli a gépelési folyamatokat és a játékállapotot. Ez magában foglalja a helyes karakterek számítását, az eltelt idő nyomon követését és az eredmények mentését.
  - Új játék elindítása: **StartNewSession()**, amely a megadott szövegen alapuló gépelést kezdi meg, kezeli a hibás karaktereket és az aktuális pozíciót.
  - Különböző játékmódok indítása: **SetNormalGame()**, **SetCasualGame(int numberOfWords)**, és **SetChosenGame(Entry entry)** metódusokkal, amelyek véletlenszerű vagy kiválasztott szövegeket adnak a gépelési feladathoz.
  - A játékmenet leállítása és eredmény mentése: **EndSession()** menti a gépelési eredményt, míg a **TerminateSession()** csak leállítja a folyamatot.
  - A **TimerTick** esemény minden másodpercben frissíti az eltelt időt, míg a **CalculateWpm()** metódus számolja a szavak per perc (WPM) értékét.

- A játékmenet eseményeit a **HandleKeyPress()**, **HandleBackspace()**, és **HandleEnter()** funkciók vezérlik, amelyek a helyes vagy hibás karakterek kezeléséért felelősek.
- Az osztály egy **IDataHandler** interfészt használ a szövegek betöltéséhez és az eredmények mentéséhez.



4. ábra: A Model csomag osztálydiagramja

- Nézetmodell (5. ábra):
  - A nézetmodell megvalósításához felhasználunk egy általános utasítás (**DelegateCommand**), valamint egy ős változásjelző (**ViewModelBase**) osztályt.
  - A nézetmodell feladatait a **MainViewModel**, az **EntriesPanelViewModel**, az **EntryViewModel**, az **EditEntryViewModel** és az **AddEntryViewModel** végzik.
  - A **MainViewModel** kezeli a játékmódok közti választást, a felhasználói felület láthatóságát, az eddig elért eredmények megjelenítését, és a gépelési folyamat kezelését.
  - Az **EntriesPanelViewModel** betölti és megjeleníti az eddig elmentett szövegeket lehetőséget adva azok kezelésére.
  - **EntryViewModel** lehetőséget ad egy adott szöveg törlésére, szerkesztésére vagy akár az azokkal való gépelési folyamat indítására is.
  - Az **EditEntryViewModel** és **AddEntryViewModel** lehetőséget ad arra, hogy egy adott szöveget szerkeszthessünk vagy hozzáadjunk.



***5**. ábra: A nézetmodell osztálydiagramja*

- Nézet:
  - A **MainWindow** az alkalmazás főablaka, amely két fő komponensből áll:
    - Menüsáv: Egy **StackPanel** elemeként helyezkedik el, amely gombokat tartalmaz különböző funkciókhoz, mint a Normal,

Casual, Choose Text és Add Text. Ezek a gombok parancskötéssel kapcsolódnak a ViewModel-hez, és láthatóságukat adatkötéssel szabályozzuk.

- Fő tartalmi terület: Két fő szekcióval rendelkezik:
  - Gépelési terület: Egy **RichTextBox** segítségével jeleníti meg a gépelendő szöveget. A státuszokat, például visszaszámlálást és áttekintő szöveget, különböző **TextBlock** elemek jelenítik meg.
  - Szöveglista: Egy **ItemsControl** vezérlőn belül helyezkedik el, amely az **EntryView** elemeket jeleníti meg. Az elemek megjelenítése dinamikusan történik a bejegyzések adataiból.
- Reszponzivitás: Az ablak minimális mérethatárokkal és adaptív rács-elrendezéssel rendelkezik.

- **EntryView:** Az EntryView egy egyedi vezérlő, amely egy-egy szövegnek az adatait jeleníti meg:
  - Szerkezete: Egy **Border** elem veszi körül, amely tartalmaz egy rácsot három szekcióval:
  - Cím és szerző: Különböző szövegelemek jelenítik meg a bejegyzés címét és szerzőjét.
  - Előnézet: A bejegyzés szövegének rövid előnézete látható.
  - Műveleti gombok: Gombok a bejegyzés szerkesztésére, törlésére és lejátszására. Ezek az elemek parancsokat kötnek a ViewModel-hez.
  - Animációk: Az egérmutató fölé helyezésekor a **Border** elem mérete enyhén megnövekszik egy storyboard animáció segítségével.
- **EditEntryView:** Az EditEntryView az új szövegek létrehozására és meglévők szerkesztésére szolgál, van benne:
  - Egy négy sorból álló rács, amely külön szekciókat tartalmaz a cím, szerző és szöveg adatainak bevitelére.
  - Az utolsó sorban két gomb található (CANCEL és SAVE), amelyek műveleteket indítanak el.
  - Adatkötés: A beviteli mezők a ViewModel tulajdonságaihoz kötődnek, és a módosítások valós időben frissülnek.
- **AddEntryView:** Ugyanaz a felülettel rendelkezik, mint az **EditEntryView**, viszont itt a szöveget létrehozza, nem szerkeszti.

- Környezet (6. ábra):
    - Az **App** osztály induláskor inicializálja az alkalmazás globális eseménykezelését. Az osztály feladata az általános felhasználói műveletek, például billentyűleütések és drag-and-drop műveletek lekezelése és szabályozása.
    - Az **OnStartup** metódus felülírásával az alkalmazás indulásakor regisztrálásra kerülnek globális eseménykezelők:
        - **GlobalPreviewKeyDownHandler**: Kezeli a billentyűleütéseket, megakadályozva például a Ctrl-alapú műveleteket.
        - **GlobalPreviewDragOverHandler és GlobalPreviewDropHandler**: Megakadályozzák a drag-and-drop műveletek végrehajtását.
- Tesztelés:
    - A **TypingSession** modell funkcionalitása egységtesztek segítségével lett ellenőrizve a **TypingSessionTests** osztályban. Az alábbi tesztesetek kerültek megvalósításra:
        - **SetNormalGame_SetsGameDataCorrectly**: Ellenőrzi, hogy a normál játék beállítása során a megfelelő adatok (szöveg és azonosító) helyesen kerülnek beállításra.
        - **SetCasualGame_SetsGameDataCorrectly**: Ellenőrzi, hogy egy CASUAL játék beállítása során a megfelelő szöveg és azonosító helyesen került beállításra a megadott szavak számának figyelembevételével.
        - **StartNewSession_ThrowsException_WhenTextToWriteIsEmpty**: Ellenőrzi, hogy kivételt dob, ha az írandó szöveg üres, amikor új játékot indítanak.
        - **StartNewSession_SetsInitialValuesCorrectly**: Ellenőrzi, hogy az új játék indítása helyesen állítja be a kezdeti értékeket.
        - **HandleKeyPress_CorrectKey_ReturnsTrue**: Ellenőrzi, hogy helyes billentyűleütés esetén a metódus igazat ad vissza, és a pozíció megfelelően frissül.
        - **HandleKeyPress_IncorrectKey_ReturnsFalse**: Ellenőrzi, hogy helytelen billentyűleütés esetén a metódus hamisat ad vissza, és a hibás karakterek száma frissül.
        - **HandleBackspace_CorrectUsage_ReturnsTrue**: Ellenőrzi, hogy a megfelelő backspace használat esetén a metódus igazat ad vissza, és a pozíció és a hibás karakterek száma frissül.
        - **HandleBackspace_AtBeginning_ReturnsFalse**: Ellenőrzi, hogy a backspace használata a kezdő pozícióban hamisat ad vissza.
        - **HandleEnter_CorrectUsage_ReturnsTrue**: Ellenőrzi, hogy helyes használat esetén az Enter metódus igazat ad vissza, és a pozíció frissül.

- **HandleEnter_IncorrectUsage_ReturnsFalse**: Ellenőrzi, hogy helytelen használat esetén az Enter metódus hamisat ad vissza.
- **EndSession_SavesScore**: Ellenőrzi, hogy a játék végén a pontszámot helyesen elmenti.
- **CalculateWpm_CalculatesCorrectly**: Ellenőrzi, hogy a WPM (szó per perc) pontosan kerül kiszámításra.

- A **DataHandler** osztály funkcionalitása egységtesztek segítségével lett ellenőrizve a **DataHandlerTests** osztályban. Az alábbi tesztesetek kerültek megvalósításra:
  - **GetAllEntries_NoEntries_ReturnsEmptyList**: Ellenőrzi, hogy a GetAllEntries metódus üres listát ad vissza, ha nincsenek bejegyzések.
  - **UploadText_AddsEntryToFile**: Ellenőrzi, hogy a szöveg sikeresen hozzáadódik a fájlhoz.
  - **GetRandomEntry_ThrowsException_WhenNoEntries**: Ellenőrzi, hogy kivételt dob, ha megpróbálnak véletlen bejegyzést kérni, de nincsenek bejegyzések.
  - **GetEntryById_ReturnsCorrectEntry**: Ellenőrzi, hogy a megfelelő bejegyzést adja vissza az azonosító alapján.
  - **EditEntry_UpdatesEntrySuccessfully**: Ellenőrzi, hogy a bejegyzés sikeresen frissül.
  - **DeleteEntry_RemovesEntrySuccessfully**: Ellenőrzi, hogy a bejegyzés sikeresen törlődik.
  - **SaveScore_AddsScoreToFile**: Ellenőrzi, hogy a pontszám sikeresen hozzáadódik a fájlhoz.
  - **GetTopScores_ReturnsTopScores**: Ellenőrzi, hogy a legjobb pontszámokat helyesen adja vissza.
  - **GetTopWords_ThrowsException_WhenWordsFileNotFound**: Ellenőrzi, hogy kivételt dob, ha a szavakat tartalmazó fájl nem található.