### Atrybuty

* Każdy z nich posiada:
  + wartość,

id: nazwę oraz domenę – nazwę drugy atrybutów do które należy

|  |  |  |  |
| --- | --- | --- | --- |
| **Atrybut** | **Klasa** | **Kod** | **Opis** |
| Nazwa | UnitName | description.name |  |
| Notka | UnitNote | description.note |  |
| Liczba punktów doświadczenia |  | progress.currentXp |  |
| Punkty doświadczenia  do następnego poziomu |  | progress.nextLvlXp |  |
| Poziom |  | progress.level |  |
| Rodzaj | UnitType | description.type | W – wojownik, R – strzelec, S – wsparcie, M – Mag,   * **uwaga**: nie jest wczytywany z properties,  a wyznaczany na pods. *zasięgu* oraz *liczby celi* |
| Punkty życia | UnitHitPoints | hitPoints.current |  |
| Max punkty życia | UnitMaxHitPoints | hitPoints.max |  |
| Pancerz | UnitArmor | resistance.armor | Jego procentowa wytrzymałość wpływa na wszystkie odporności |
| Odporność | UnirResistance**X** | resistance.X | **X** = Fire, Water, Air, Death, Poison, Mind, Life, Earth, |
| Atak |  | attack.sourceX | Fire, Water, Air, Death, Poison, Mind, Meele, Projectile, Life, Earth   * **uwaga**: MEELE = PROJECTILE, rozróżnienie istnieje tylko dla sprawdzania *pancerza,* |
| Celność |  | attack.accuracy |  |
| Obrażenia | UnitDamage | attack.damage |  |
| Inicjatywa |  | attack.initiative |  |
| Zasięg |  | attack.range |  |
| Cele |  | attack.targets |  |
| Ruch |  | action.stamina |  |
| „Gniew” |  | action.specialPoints |  |

### Modyfikatory wartości atrybutów:

* jako dane wejściowe są opisywane przez liczby
  + całkowite np. 120 = dodać 120, -50 = odjąć 50
  + zmiennoprzecinkowe: 50.0 = +50%, zaś -33.0 = -33%
    - w programie są przechowywane jako liczby: pomiędzy (0;1)
    - w plikach z danymi są zapisywane w postaci X.0 np. 50.0 = 50%
* obsługuje je klasa *AttrbuteValueChangeFactorImpl*
* klasa posiada też funkcjonalność (konstruktor) do obliczania zmiany wartości atrybutu na podstawie jego dawnej i nowej wartości
  + est to wykorzystywane w *Atrybutach powiązanych*

### Walidatory

* każdy obiekt przechowujący dane powinien implementować interfejs *Validatable*

### Powiązane atrybuty

* implementują interfejs *LinkedAttributesChange*
* każdy z nich przechowuje mapę: atrybut – zamiana w postaci wpółczynnika *double*
* przy zmniejszanu wartości atrybutów powiązanych wymagana jest znajomość zmiany atrybutu głównego, tak aby można było odpowiednio przeskalować zmianę na atrybut powiązany np.
  + zmiana atrybutu głównego HP = -50%,
  + współczynnik zmiany dla *Obrażeń* = 0.75
  + zatem ostateczny współczynnik = -50% \* 0.75 = -37,5%
* **proces**
  + zmiany odbywa się w metodach *atrybut.updateReferencedAttributes*
  + przebieg:
    - obliczana jest zmiana wartości atrybutu powiązanego w zależności od jego współczynnika zmiany (*Linked...*) oraz zmiany wartości atrybutu głównego
    - pobierana jest obecna wartość atrybutu powiązanego
    - jego wartość jest aktualizowana przed współczynnik (...*Factor*)