# Atrybuty

* Każdy z nich jest opisywany przez:
  + wartość: liczbową lub tekstową
  + id:
    - nazwę
    - domenę – nazwę drugy atrybutów do której należy
* Każdy jest micro serwisem do zarzadzania sobą,
  + posiada metody do aktualizacji wartości swojej i atrybutów powiązanych
  + Wyjątkiem są klasy reprezentujące odporności, które dziedziczą po *AbstractResistance*

|  |  |  |  |
| --- | --- | --- | --- |
| **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

* implementacja: *AttrbuteValueChangeFactorImpl*
* 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%
* w jednym z konstruktorów tej klasy możliwe jest obliczenie współczynnika na podstawie dawnej i nowej wartości atrybutu
  + wykorzystywane w *Atrybutach powiązanych*

## Walidatory

* każdy obiekt przechowujący dane powinien implementować interfejs *Validatable*

## Powiązane atrybuty

* interfejs: *LinkedAttributesChange*
* każdy obiekt przechowuje mapę
  + atrybut –> zamiana w postaci wpółczynnika *double*
  + współczynnik zmienno przecinkowy określa o ile procent zmiany wwartości *atrybutu powiązanego* przypada na 1% zmiany *atrybutu głównego*
    - np. w Hp 0.1 dla Pancerza oznacza, że np. przy obniżeniu o 1% Punktów życia należy obliżyć 0.1% wartości pancerza,
* 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% (= 0.5),
  + współczynnik zmiany dla *Obrażeń* = 0.75
  + zatem ostateczny współczynnik = 0.5 \* 0.75 = 0.375
* **proces**
  + zmiany odbywa się w metodach [*atrybut*]*.updateReferencedAttributes*
  + przebieg:
    - obliczany jest *współczynnik zmiany* wartości atrybutu powiązanego w zależności od:
      * jego współczynnika zmiany
      * zmiany wartości atrybutu głównego
    - pobierana jest obecna wartość atrybutu powiązanego
    - jego wartość jest aktualizowana przed *współczynnik zmiany* (zawsze procentowy)

## Zmiana wartości atrybutów – proces

* Dane wejściowe:
  + HP = 200, zmiana = -100
* obliczanie współczynnika *zmiany atrybutu głównego*
  + 0.5 (-50%)
* pobranie nazwy atrybutu powiązanego i jego współczynnika zmiany:
  + atrybut = DAMAGE, zmiana = 0.75
* obliczenie procentowej zmiany dla *atrybutu powiąznego*
  + 0.5 \* 0.75 = 0,375
* zamiana wartości *atrybutu powiązanego* w zależności od współ. zmiany
  + wartość atrybutu ARMOR = 50
  + zmiana: 1.0 - 0.375 = 0.925
  + nowa wartość: 50 \* 0.925 = 46,25
* Wynik działania
  + **nowa wartość: 46,25**