Présentation BONUS PROJET 3, LAURENT JUNDT

* Ajout d'un bonus qui permet de définir un type par personnage qui servira au calcul de l'attaque.

Les type sont les suivants :

- Magus de type healer,
- Combattant de type human
- Dwarf de type elf
- Colosse de type rock

* Ajout d'une probabilité de coup critique par attaque ou soin par personnage, si un coup est un critique la puissance d'attaque de l'arme du personnage est doublée.

Lorsque une attaque se produit entre 2 personnages, les 2 types sont comparés entre eux afin d'attribuer un coefficient d'attaque qui sera multiplié par la puissance d'attaque de l'arme et du coup critique éventuel.

Partie technique et intégration du code

Partie coup critique :

Définition dans chaque class personnage le % de critique lui appartenant. Variable initialisée dans la class maitre Character :

```
class Character {

    let name: String
    var weapon: Weapon
    var life: Decimal
    let max_life: Decimal
    var crit: Int
    var type: Typeofcharacter

    init(name: String, weapon: Weapon, crit: Int,type: Typeofcharacter, max_life: Decimal) {
        self.name = name
        self.weapon = weapon
        self.crit = crit
        self.type = type

        self.max_life = max_life
        self.life = max_life

}
class Combattant: Character {

    init(name: String) {
        super.init(name: name, weapon: Sword(), crit: 15,type: Typeofcharacter.Combattant, max_life: 100)
    }
}
```

Ajout d'une constante random pour la partie critique dans la fonction attack() et healing() appartenant à la class Character. Le critique est indiqué au joueur par un print()

```
let number_crit = Int.random(in: 1 ... (100/crit))
```

Partie type de personnages :

Ajout d'une énumération dans la class Character.swift qui permet la définition des types pour chaque personnage

```
enum Typeofcharacter: String {
    case Magus = "healer"
    case Combattant = "human"
    case Colosse = "rock"
    case Dwarf = "elf"
}
```

Définition des facteurs d'attaque en fonction du type qui attaque et du type qui reçoit l'attaque dans la fonction type_factor() qui retourne un nombre décimal.

```
func type_factor(attack_factor_par: Decimal) -> Decimal{

    var factor: Decimal = 1
    //var factor depends what kind of character type
    factor = factor * attack_factor_par
    print("attack factor : \(factor)")

    return factor
}
```

Le paramètre de la fonction est entré lors des choix du type a l'aide d'un if suite à la comparaison des 2 types. Toujours dans la class Character fonction attack()

```
let number_crit = Int.random(in: 1 ... (100/crit))
var attack_factor: Decimal = 1

var crit_factor: Decimal = 1.00
 //types :
//human vs rock
print("Type \(type.rawValue) vs type \(who.type.rawValue)")
if type.rawValue == "human" && who.type.rawValue == "rock"{attack_factor = type_factor(attack_factor_par: 0.25)}
//human vs elf
if type.rawValue == "human" && who.type.rawValue == "elf"{attack_factor = type_factor(attack_factor_par: 1.5)}
//rock vs human
if type.rawValue == "rock" && who.type.rawValue == "human"{attack_factor = type_factor(attack_factor_par: 1.5)}
//rock vs elf
if type.rawValue == "rock" && who.type.rawValue == "elf"{attack_factor = type_factor(attack_factor_par: 1.8)}
//elf vs rock
if type.rawValue == "elf" && who.type.rawValue == "rock"{attack_factor = type_factor(attack_factor_par: 0.5)}
//elf vs human
if type.rawValue == "elf" && who.type.rawValue == "human"{attack_factor = type_factor(attack_factor_par: 0.75)}
```