PAYAN Mickaël

Projet 03 : Créez votre premier jeu en swift.

Durant ce projet, il était demandé de réaliser un jeu en utilisant la programmation orientée objet. Celui-ci doit être composé de deux joueurs. Chaque joueur a une équipe composée de trois personnages et chaque personnage des propriétés propres tel un nom, des points de vie et une arme.

Il était imposé de réaliser dans le code, dès le lancement du jeu, que chaque joueur sélectionne les personnages de son équipe. Le joueur devra choisir pour chaque personnage un nom différent de tous les autres personnages déjà créés dans le jeu.

```swift
20  private func chooseCharacterName(_ characterType: CharacterType) {
21      print("Choose \(characterType) name:")
22      if let name = readLine() {
23          if isCharacterNameAlreadyChosen(name) {
24              print("This name has been already choosen!")
25              chooseCharacterName(characterType)
26          } else if name.isEmpty {
27              print("😡 Is empty. You need to choose a name for your character! 😡")
28              chooseCharacterName(characterType)
29          } else if name != name.trimmingCharacters(in: .whitespacesAndNewlines) {
30              print(" Your character needs a name without spaces.")
31              chooseCharacterName(characterType)
32          } else {
33              switch characterType {
34              case .warrior:
35                  characters.append(Warrior(name: name))
36              case .colossus:
37                  characters.append(Colossus(name: name))
38              case .magus:
39                  characters.append(Magus(name: name))
40              case .priest:
41                  characters.append(Priest(name: name))
42              }
43              print("Your \(characterType) \(name) has been added to your team.")
44          }
45      }
46  }
47
48  private func isCharacterNameAlreadyChosen(_ name: String) -> Bool {
49      for character in characters {
50          if character.name == name {
51              return true
52          }
53      }
54      return false
55  }
```

class Player

```swift
86  func makeYourTeamBySelectingCharacters() {
87      print(" ⚠️ The same character cannot be selected more than once. Select your char
            characters per team. ⚠️")
88      while characters.count < maximumNumberOfSelectableCharacters {
89          print("""
90              \(name), you must choose \(3 - characters.count) character:
91              1 - ⚔️ Warrior ⚔️ - Simple, basic but efficient.
92              Damage: 20  ||  Lifepoint: 90
93              2 - 💪 Colossus 💪 - Thick smelly creature, can't even see his feet.
94              Damage: 10  ||  Lifepoint: 110
95              3 - 🧙🏾 Magus 🧙🏾 - Devastating power, but enough sensitive.
96              Damage: 25  ||  Lifepoint: 70
97              4 - 🙏 Priest 🙏 - Robust ally, incapable of causing harm.
98              Healing: 10  ||  Lifepoint: 90
99              """)
100
101             if let picks = readLine(), !picks.isEmpty {
102                 switch picks {
103                 case "1":
104                     chooseCharacterNameIfHasNotBeenAlreadyChosen(.warrior)
105                 case "2":
106                     chooseCharacterNameIfHasNotBeenAlreadyChosen(.colossus)
107                 case "3":
108                     chooseCharacterNameIfHasNotBeenAlreadyChosen(.magus)
109                 case "4":
110                     chooseCharacterNameIfHasNotBeenAlreadyChosen(.priest)
111                 default:
112                     print("Please, make your choice.")
113                 // if the choice is different of 1,2,3 or 4.
114                 }
115             } else {
116                 print("You need to choose a character, between 1,2,3 and 4.")
117                 // if the choice is empty.
118             }
119         }
120     }
121
122     func chooseCharacterNameIfHasNotBeenAlreadyChosen(_ characterType: CharacterType) {
123         if hasBeenAlreadyChosen(characterType) {
124             print("The \(characterType) is already chosen! Please, make another choice.")
125         } else {
126             chooseCharacterName(characterType)
127         }
128     }
```

La fonction « *chooseCharacterName* » (l.20) permet, comme son nom l'indique, de choisir le nom du personnage. Dans le corps de celle-ci sera implémentée (l.23) la fonction « *isCharacterNameAlreadyChosen* » (l.48) afin que chaque personnage choisi ne puisse pas avoir le même nom. Si aucune similitude n'est repérée, alors le personnage sera ajouté à l'équipe.
Tant qu'à la fonction « *makeYourTeamBySelectingCharacters* » (l.86), elle permettra la sélection des personnages tant que chaque joueur n'en dispose pas trois. Impossible d'en n'avoir moins ni plus.
Enfin elle sera implémentée à plusieurs reprises (l.104, l.106, l.108, l.110) de la fonction « *chooseCharacterNameHasNotBeenAlreadyChosen* » (l. 122), de façon que dans chaque équipe, il n'y est pas plusieurs fois le même personnage. Une fois vérifier, si le choix ne correspond à aucun autre, le joueur pourra choisir le nom de celui-ci, donc « *chooseCharacterName* » (l.20).

De plus, le combat sera au tour par tour. Le Joueur 1 sélectionne un personnage de son équipe puis sélectionne un personnage du Joueur 2 qui va subir l'attaque (ou sélectionne un de ses propres personnages en cas de soin). Les personnages attaquent avec une arme, c'est celle-ci qui déterminera la valeur de l'attaque. Ensuite, vérifier si la partie est terminée, et si ce n'est pas le cas, c'est au tour du Joueur 2.

```swift
    private var attackingPlayer: Player {
        players[turns % 2]
    }
    private var targetPlayer: Player {
        players[(turns + 1) % 2]
    }
```

class Game

Le tour par tour est réalisé par les deux propriétés *« attackingPlayer »* et *« targetPlayer »* via la technique du modulo.
La classe Character a une propriété *« weapon »* prenant pour valeur un type Weapon (se référer à l'organigramme). Celle-ci est initialisée, donc chaque classe fille de Character dispose d'une arme. La classe Weapon dispose d'une propriété *« name »* du type String et *« damage »* du type Int initialisées, ainsi que de plusieurs classes filles. Donc chaque classe filles dispose d'une arme différente avec une valeur d'attaque différente.

```swift
    func isDefeat() -> Bool {
        let charactersAreDead = characters.filter { $0.isDead() }
        if charactersAreDead.count == maximumNumberOfSelectableCharacters {
            return true
        }
        return false
    }
}
```

class Player

La fonction *« isDefeat »* va venir filtrer si tous les personnages d'un joueur sont morts.

Enfin, lorsque la partie est terminée, et donc que tous les personnages d'une équipe sont morts, afficher le joueur qui a gagné et les statistiques du jeu qui comprennent le nombre de tour.
Petit plus, il est souhaité qu'il y est un peu de hasard. Un coffre qui contient une arme peut apparaitre et conférer un bonus au personnage l'ayant obtenu.

```swift
    private func statistics() {
        guard let winner = players.filter({ !$0.isDefeat() }).first else { return }

        print("""
            ❌ THE GAME IS OVER ❌
            🎊 The winner is --|  \(winner.name)  |-- 🎊
            ♻️ The game was finished in \(turns)turns. ♻️
            🎁 Bonus number: \(bonusCount) 🎁
            """)
    }
```

class Game

La partie est terminée lorsque la fonction *« statistics »* à filtrer lequel des deux joueurs à tous ses personnages morts, donc lequel des deux joueurs a perdu. Par la suite, cette fonction même affichera la fin du jeu, ainsi que le nom du vainqueur.

```swift
104 |   private func getsRandomBonus(fightingCharacter: Character) {
105         guard arc4random_uniform(100) < percentageOfHavingBonus else {
106             return
107         }
108         if fightingCharacter is Warrior, hasAlreadyGotBonus(player: attackingPlayer, weapon: DoubleSwords()) {
109             return
110         } else if fightingCharacter is Colossus, hasAlreadyGotBonus(player: attackingPlayer, weapon: GiantFronde()) {
111             return
112         } else if fightingCharacter is Magus, hasAlreadyGotBonus(player: attackingPlayer, weapon: VoidStaff()) {
113             return
114         } else if fightingCharacter is Priest, hasAlreadyGotBonus(player: attackingPlayer, weapon: TibetanBowl()) {
115             return
116         } else {
117             bonusCount += 1
118             fightingCharacter.addBonusToWeapon()
119         }
120     }
```

class Game

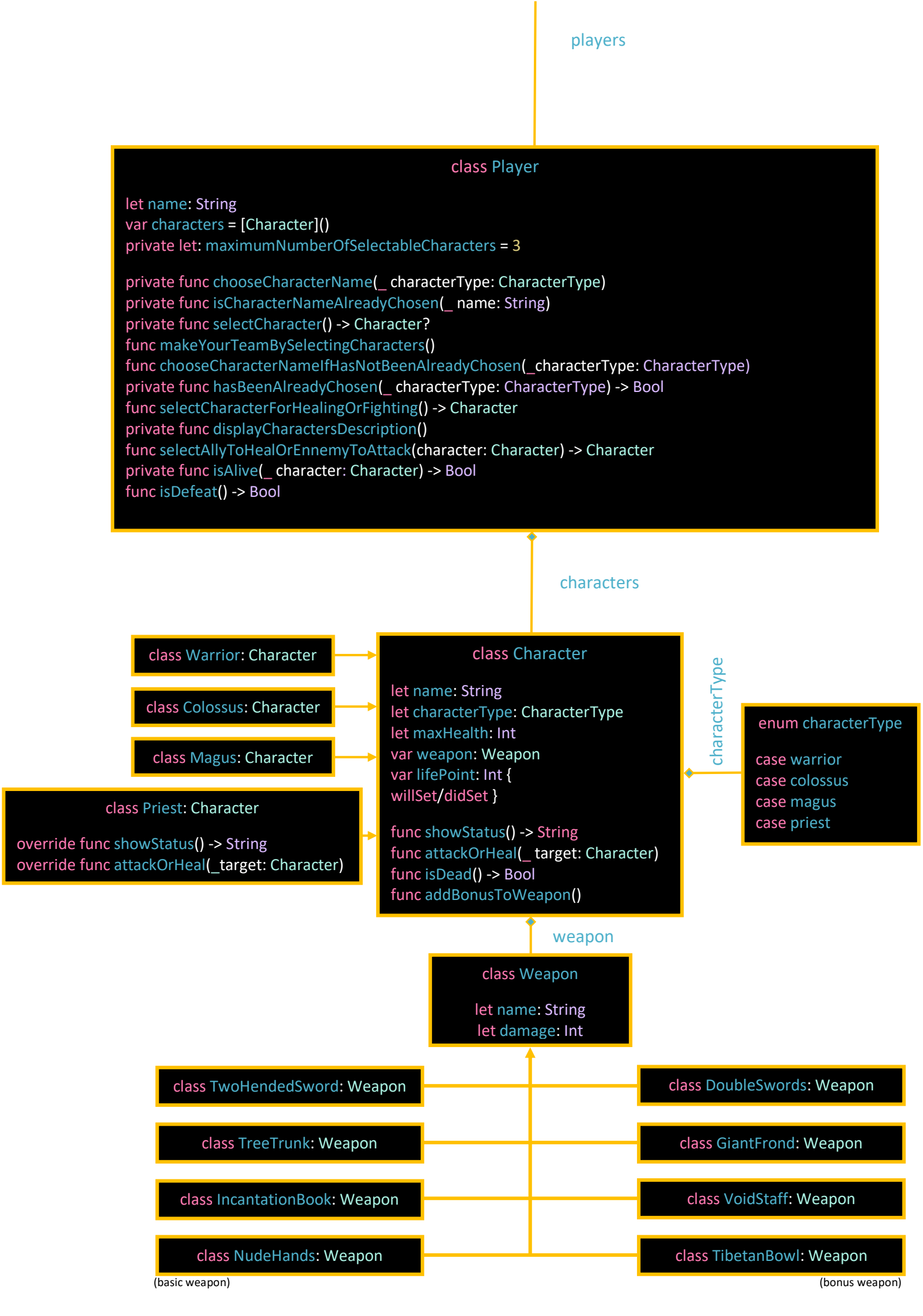
Enfin pour le plus demandé, une arme bonus aléatoire peut apparaître et conférer un bonus au personnage l'ayant obtenu grâce à la fonction *« getsRandomBonus ».*

ORGANIGRAMME :

class Game

```swift
private var players = [Player]()
private let maximumNumberOfSelectablePlayers = 2
private var bonusCount = 0
private let percentageOfHavingBonus = 5
private var turns = 1
private var attackingPlayer: Player { players[turns % 2}
private var targetPlayer: Player { players[(turns + 1) % 2] }

func start()
private func createPlayers()
private func hasNameAlreadyBeenChosen(_ name: String) -> Bool
private func selectCharacterForEachPlayer()
private func makeFight()
private func chooseTarget(_ fightingCharacter: Character) -> Character
private func getsRandomBonus(_ fightingCharacter: Character)
private func hasAlradyGotBonus(player: Player, weapon: Weapon) -> Bool
private func statistics()
private func exitGame()
```


players
class Player
let name: String
var characters = [Character]()
private let: maximumNumberOfSelectableCharacters = 3
private func chooseCharacterName(_ characterType: CharacterType)
private func isCharacterNameAlreadyChosen(_ name: String)
private func selectCharacter() -> Character?
func makeYourTeamBySelectingCharacters()
func chooseCharacterNameIfHasNotBeenAlreadyChosen(_characterType: CharacterType)
private func hasBeenAlreadyChosen(_ characterType: CharacterType) -> Bool
func selectCharacterForHealingOrFighting() -> Character
private func displayCharactersDescription()
func selectAllyToHealOrEnnemyToAttack(character: Character) -> Character
private func isAlive(_ character: Character) -> Bool
func isDefeat() -> Bool
characters
class Warrior: Character
class Colossus: Character
class Magus: Character
class Priest: Character
override func showStatus() -> String
override func attackOrHeal(_target: Character)
class Character
let name: String
let characterType: CharacterType
let maxHealth: Int
var weapon: Weapon
var lifePoint: Int {
willSet/didSet }
func showStatus() -> String
func attackOrHeal(_ target: Character)
func isDead() -> Bool
func addBonusToWeapon()
characterType
enum characterType
case warrior
case colossus
case magus
case priest
weapon
class Weapon
let name: String
let damage: Int
class TwoHendedSword: Weapon
class TreeTrunk: Weapon
class IncantationBook: Weapon
class NudeHands: Weapon
(basic weapon)
class DoubleSwords: Weapon
class GiantFrond: Weapon
class VoidStaff: Weapon
class TibetanBowl: Weapon
(bonus weapon)