

## EXERCICE 5 (4 points)

*Cet exercice porte sur la Programmation Orientée Objet.*

Les participants à un jeu de LaserGame sont répartis en équipes et s'affrontent dans ce jeu de tir, revêtus d'une veste à capteurs et munis d'une arme factice émettant des infrarouges.

Les ordinateurs embarqués dans ces vestes utilisent la programmation orientée objet pour modéliser les joueurs. La classe `Joueur` est définie comme suit :

```
1 class Joueur:
2     def __init__(self, pseudo, identifiant, equipe):
3         ''' constructeur '''
4         self.pseudo = pseudo
5         self.equipe = equipe
6         self.id = identifiant
7         self.nb_de_tirs_emis = 0
8         self.liste_id_tirs_recus = []
9         self.est_actif = True
10
11     def tire(self):
12         '''méthode déclenchée par l'appui sur la gachette'''
13         if self.est_actif == True:
14             self.nb_de_tirs_emis = self.nb_de_tirs_emis + 1
15
16     def est_determine(self):
17         '''methode qui renvoie True si le joueur réalise un
18         grand nombre de tirs'''
19         return self.nb_de_tirs_emis > 500
20
21     def subit_un_tir(self, id_recu):
22         '''méthode déclenchée par les capteurs de la
23         veste'''
24         if self.est_actif == True:
25             self.est_actif = False
26             self.liste_id_tirs_recus.append(id_recu)
```

1. Parmi les instructions suivantes, recopier celle qui permet de déclarer un objet `joueur1`, instance de la classe `Joueur`, correspondant à un joueur dont le pseudo est "Sniper", dont l'identifiant est 319 et qui est intégré à l'équipe "A":

**Instruction 1 :** `joueur1 = ["Sniper", 319, "A"]`

**Instruction 2 :** `joueur1 = new Joueur["Sniper", 319, "A"]`

**Instruction 3 :** `joueur1 = Joueur("Sniper", 319, "A")`

**Instruction 4 :** `joueur1 = Joueur{"pseudo":"Sniper",  
"id":319, "equipe":"A"}`

2. La méthode `subit_un_tir` réalise les actions suivantes :  
 Lorsqu'un joueur actif subit un tir capté par sa veste, l'identifiant du tireur est ajouté à l'attribut `liste_id_tirs_recus` et l'attribut `est_actif` prend la valeur `False` (le joueur est désactivé). Il doit alors revenir à son camp de base pour être de nouveau actif.
  - a. Écrire la méthode `redevenir_actif` qui rend à nouveau le joueur actif uniquement s'il était précédemment désactivé.
  - b. Écrire la méthode `nb_de_tirs_recus` qui renvoie le nombre de tirs reçus par un joueur en utilisant son attribut `liste_id_tirs_recus`.
  
3. Lorsque la partie est terminée, les participants rejoignent leur camp de base respectif où un ordinateur, qui utilise la classe `Base`, récupère les données.  
 La classe `Base` est définie par :
  - ses attributs :
    - `equipe` : nom de l'équipe (`str`). Par exemple, "A" ,
    - `liste_des_id_de_l_equipe` qui correspond à la liste (`list`) des identifiants connus des joueurs de l'équipe,
    - `score` : score (`int`) de l'équipe, dont la valeur initiale est 1000 ;
  - ses méthodes :
    - `est_un_id_allie` qui renvoie `True` si l'identifiant passé en paramètre est un identifiant d'un joueur de l'équipe, `False` sinon,
    - `incremente_score` qui fait varier l'attribut `score` du nombre passé en paramètre,
    - `collecte_information` qui récupère les statistiques d'un participant passé en paramètre (instance de la classe `Joueur`) pour calculer le score de l'équipe .

```

1 def collecte_information(self, participant):
2     if participant.equipe == self.equipe : # test 1
3         for id in participant.liste_id_tirs_recus:
4             if self.est_un_id_allie(id): # test 2
5                 self.incremente_score(-20)
6             else:
7                 self.incremente_score(-10)

```

- a. Indiquer le numéro du test (**test 1** ou **test 2**) qui permet de vérifier qu'en fin de partie un participant égaré n'a pas rejoint par erreur la base adverse.
- b. Décrire comment varie quantitativement le score de la base lorsqu'un joueur de cette équipe a été touché par le tir d'un coéquipier.

On souhaite accorder à la base un bonus de 40 points pour chaque joueur particulièrement déterminé (qui réalise un grand nombre de tirs).

4. Recopier et compléter, en utilisant les méthodes des classes `Joueur` et `Base`, les 2 lignes de codes suivantes qu'il faut ajouter à la fin de la méthode `collecte_information` :

```
..... #si le participant réalise un grand nombre de tirs  
..... #le score de la Base augmente de 40
```