

Algorithmes et Pensée Computationnelle

Consolidation 1

Les exercices de cette série sont une compilation d'exercices semblables à ceux vus lors des semaines précédentes. Le but de cette séance est de consolider les connaissances acquises lors des travaux pratiques des semaines 1 à 6.

Le code présenté dans les énoncés se trouve sur Moodle, dans le dossier **Code**.

1 Introduction et architecture des ordinateurs

Le but de cette section est de comprendre le fonctionnement d'un ordinateur. La série d'exercices sera axée autour de conversions en base binaire, décimale, hexadécimale, base 5 et de calcul de base en suivant le modèle Von Neumann.

1.1 Conversion

Question 1: (🕒 10 minutes) Conversion

1. Convertir le nombre $FFFFFF_{(16)}$ en base 10.
2. Convertir le nombre $4321_{(5)}$ en base 10.
3. Convertir le nombre $ABC_{(16)}$ en base 2.
4. Convertir le nombre $254_{(10)}$ en base 15.
5. Convertir le nombre $11101_{(2)}$ en base 10.

💡 Conseil

N'oubliez pas qu'en Hexadécimal, A vaut 10, B vaut 11, C vaut 12, D vaut 13, E vaut 14 et F vaut 15.

>_ Solution

1. $FFFFFF_{(16)} = 16777215_{(10)}$
2. $4321_{(5)} = 586_{(10)}$
3. $ABC_{(16)} = 101010111100_{(2)}$
4. $254_{(10)} = 11E_{(15)}$
5. $11101_{(2)} = 29_{(10)}$

1.2 Conversion et arithmétique

Question 2: (🕒 5 minutes) Conversion, addition et soustraction :

Effectuez les opérations suivantes :

1. $10110101_{(2)} + 00101010_{(2)} = \dots_{(10)}$
2. $70_{(10)} - 10101010_{(2)} = \dots_{(10)}$

💡 Conseil

Convertissez dans une base commune avant d'effectuer les opérations.

>_ Solution

1. $10110101_{(2)} + 00101010_{(2)} = 223_{(10)}$
3. $70_{(10)} - 10101010_{(2)} = -100_{(10)}$

1.3 Modèle de Von Neumann

Dans cette section, nous allons simuler une opération d'addition dans le **modèle de Von Neumann**, il va vous être demandé à chaque étape (FDES) de donner la valeur des registres.

État d'origine :

A l'origine, notre **Process Counter (PC)** vaut **00100001**.

Dans la mémoire, les instructions sont les suivantes :

Adresse	Valeur
00100001	00110100
00101100	10100110
01110001	11111101

Les registres sont les suivants :

Registre	Valeur
00	01111111
01	00100000
10	00101101
11	00001100

Les opérations disponibles pour l'unité de contrôle sont les suivantes :

Numéro	Valeur
00	ADD
01	XOR
10	MOV
11	SUB

Question 3: (🕒 5 minutes) Fetch

À la fin de l'opération **FETCH**, quelles sont les valeurs du **Process Counter** et de l'**Instruction Register** ?

💡 Conseil

Pour rappel, l'unité de contrôle (Control Unit) commande et contrôle le fonctionnement du système. Elle est chargée du séquençage des opérations. Après chaque opération **FETCH**, la valeur du Program Counter est incrémentée (valeur initiale + 1).

>_ Solution

$$PC = 00100001_{(2)} + 1 = 00100010_{(2)}$$

$$IR = 00110100_{(2)}$$

Question 4: (🕒 5 minutes) Decode

1. Quelle est la valeur de l'opération à exécuter ?
2. Quelle est l'adresse du registre dans lequel le résultat doit être enregistré ?
3. Quelle est la valeur du premier nombre de l'opération ?
4. Quelle est la valeur du deuxième nombre de l'opération ?

Conseil

Pensez à décomposer la valeur de l'**Instruction Register** pour obtenir toutes les informations demandées.

Utilisez la même convention que celle présentée dans les diapositives du cours (Architecture des ordinateurs (Semaine 2) - Diapositive 15)

Les données issues de la décomposition de l'**Instruction Register** ne sont pas des valeurs brutes, mais des références. Trouvez les tables concordantes pour y récupérer les valeurs.

>_ Solution

00 : **ADD** (valeur de l'opération à exécuter)

11 : Adresse du registre dans lequel le résultat doit être enregistré

01 : 00100000₍₂₎ (premier nombre)

00 : 01111111₍₂₎ (deuxième nombre)

Question 5: (🕒 5 minutes) Execute

Quel est résultat de l'opération ?

Conseil

Toutes les informations permettant d'effectuer l'opération se trouvent dans les données de l'**Instruction Register**.

>_ Solution

$$00100000_{(2)} + 01111111_{(2)} = 10011111_{(2)}$$

2 Logiciels système

Question 6: (🕒 5 minutes) Sous Linux et MacOS, laquelle de ces commandes modifie le filesystem ?

1. `ls -la`
2. `sudo rm -rf ~/nano`
3. `sudo kill -9 3531`
4. `more nano.txt`
5. Aucune réponse n'est correcte.

>_ Solution

La commande `sudo rm -rf ~/nano` permet de supprimer le répertoire `nano` situé dans le dossier `/Users/<Utilisateur_courant>` en mode super-utilisateur (utilisateur ayant des droits étendus sur le système).

Conseil

Attention !

Certaines commandes listées ci-dessus peuvent avoir des conséquences irréversibles.

Pour avoir une description détaillée d'une commande, vous pouvez ajouter `man` devant la commande sous Linux/MacOS ou ajouter `-h`, `--help` ou `/?` après la commande sous Windows.

3 Programmation de base

Question 7: (🕒 10 minutes)

1. Convertir $52_{(10)}$ en base 2 sur 8 bits.
2. Convertir $100_{(10)}$ en base 2 sur 8 bits.
3. Calculer en base 2 la soustraction de $01100100_{(2)}$ par $00110100_{(2)}$.
4. Déterminer au complément à deux l'opposé (multiplication par -1 en base 10) de $0110000_{(2)}$.

💡 Conseil

- Se référer aux techniques apprises dans la série 1 et la série 3
- Faire un tableau des puissances de 2 sur 8 bits.

>_ Solution

- $52_{(10)} = 32_{(10)} + 16_{(10)} + 4_{(10)} = 00110100_{(2)}$
- $100_{(10)} = 64_{(10)} + 32_{(10)} + 4_{(10)} = 01100100_{(2)}$
- $01100100_{(2)} - 00110100_{(2)} = 0110000_{(2)}$
- **Complément à 1** : $\text{not}(0110000_{(2)}) = 1001111_{(2)}$
- **Complément à 2** : Complément à 1 + 1 = $1001111_{(2)} + 1_{(2)} = 1010000_{(2)}$

4 Itération et récursivité

Question 8: (🕒 15 minutes) Itération et Récursivité

Créez une fonction itérative, puis une fonction récursive qui calculent le nombre de voyelles présentes dans un texte donné.

💡 Conseil

Pour la version itérative, parcourez toute la chaîne de caractère et incrémentez un compteur lorsque vous avez une voyelle.
Pour la version récursive, diminuez systématiquement la taille de votre chaîne de caractère. Si l'élément actuel est une voyelle, ajoutez 1, sinon, ajoutez 0.
Aidez vous d'une liste de toutes les voyelles et de la fonction `in` en Python (`List.contains()` en Java).

Voici les templates :

Python

```
1 def nb_voyelles_itérative(T,S) :
2     #TODO
3
4 def nb_voyelles_réursive(T,S) :
5     #TODO
6
7 voyelles = ['a','e','i','o','u','y']
8 texte = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean molestie elit ipsum, a tincidunt urna aliquet\
9 eget. Praesent et quam vitae justo hendrerit tristique. Ut malesuada ligula in mi ultricies tempor. Fusce blandit \
10 turpis sapien, in gravida orci aliquet et. Morbi in metus efficitur, volutpat purus sit amet, scelerisque massa. \
11 Vivamus vehicula justo quis leo feugiat fringilla. Maecenas sagittis ultrices accumsan. Cras libero est, gravida in\
12 eros ac, luctus ullamcorper nisi."
13
14 print(nb_voyelles_itérative(texte,voyelles))
15 print(nb_voyelles_réursive(texte,voyelles))
```

Java

```
1 import java.util.List;
2
3 public class Main {
4
5     public static int nb_voyelles_itérative(String S, List L){
6         //TODO
7     }
8
9     public static int nb_voyelles_réursive(String S, List L){
10        //TODO
11    }
12    public static void main(String[] args) {
13        List voyelles = List.of('a','e','i','o','u','y');
14        String texte = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean molestie elit ipsum, a tincidunt urna
15            aliquet" +
16            " eget. Praesent et quam vitae justo hendrerit tristique. Ut malesuada ligula in mi ultricies tempor. Fusce blandit" +
17            " turpis sapien, in gravida orci aliquet et. Morbi in metus efficitur, volutpat purus sit amet, scelerisque massa." +
18            " Vivamus vehicula justo quis leo feugiat fringilla. Maecenas sagittis ultrices accumsan. Cras libero est, gravida in"
19        +
20            " eros ac, luctus ullamcorper nisi.";
21        System.out.println(nb_voyelles_itérative(texte, voyelles));
22        System.out.println(nb_voyelles_réursive(texte, voyelles));
23    }
24 }
```

>_ Solution

Python :

```
1 def nb_voyelles_itérative(T,S) :
2     c = 0
3     for i in T :
4         if i in S :
5             c += 1
6         else :
7             pass
8     return c
9
10 def nb_voyelles_réursive(T,S) :
11     if len(T) == 1 :
12         if T[0] in S :
13             return 1
14         else :
15             return 0
16     else :
17         if T[0] in S :
18             return 1 + nb_voyelles_réursive(T[1:],S)
19         else :
20             return 0 + nb_voyelles_réursive(T[1:],S)
21
22 voyelles = ['a','e','i','o','u','y']
23 texte = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean molestie elit ipsum, a tincidunt urna
24     aliquet\
25     eget. Praesent et quam vitae justo hendrerit tristique. Ut malesuada ligula in mi ultricies tempor. Fusce blandit \
26     turpis sapien, in gravida orci aliquet et. Morbi in metus efficitur, volutpat purus sit amet, scelerisque massa. \
27     Vivamus vehicula justo quis leo feugiat fringilla. Maecenas sagittis ultrices accumsan. Cras libero est, gravida
28     in\
29     eros ac, luctus ullamcorper nisi."
30
31 print(nb_voyelles_itérative(texte,voyelles))
32 print(nb_voyelles_réursive(texte,voyelles))
```

>_ Solution

Java :

```
1  import java.util.List;
2
3  public class Main {
4
5      public static int nb_voyelles_itérative(String S, List L){
6          int c = 0;
7          for(int i = 0; i<S.length(); i++){
8              if(L.contains(S.charAt(i))){
9                  c ++;
10             }
11         }
12         return c;
13     }
14
15     public static int nb_voyelles_réursive(String S, List L){
16         if (S.length() == 1){
17             if (L.contains(S.charAt(0))) {
18                 return 1;
19             }
20             else{
21                 return 0;
22             }
23         }
24         else{
25             if (L.contains(S.charAt(0))) {
26                 return 1 + nb_voyelles_réursive(S.substring(1,S.length()),L);
27             }
28             else{
29                 return 0 + nb_voyelles_réursive(S.substring(1,S.length()),L);
30             }
31         }
32     }
33     public static void main(String[] args) {
34         List voyelles = List.of('a','e','i','o','u','y');
35         String texte = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean molestie elit ipsum, a
36             tincidunt urna aliquet" +
37             " eget. Praesent et quam vitae justo hendrerit tristique. Ut malesuada ligula in mi ultricies tempor.
38             Fusce blandit" +
39             " turpis sapien, in gravida orci aliquet et. Morbi in metus efficitur, volutpat purus sit amet, scelerisque
40             massa." +
41             " Vivamus vehicula justo quis leo feugiat fringilla. Maecenas sagittis ultrices accumsan. Cras libero est,
42             gravida in" +
43             " eros ac, luctus ullamcorper nisi.";
44
45         System.out.println(nb_voyelles_itérative(texte, voyelles));
46         System.out.println(nb_voyelles_réursive(texte, voyelles));
47     }
48 }
```

Question 9: (🕒 5 minutes) Lecture de code (Récursivité)

Qu'affichent les programmes suivants ?

💡 Conseil

Ces deux programmes comportent des fonctions itératives, lisez bien le code de haut en bas et lorsque la fonction fait appel à elle-même, revenez au début de la fonction et effectuez de nouveau les instructions avec les nouveaux paramètres.
Une feuille de papier pourrait vous être utile !

Programme 1 :

```
1 def recursion_1(S) :
2     if len(S) == 1 :
3         return S[0]
4     else :
5         return S[0] + recursion_1(S[1:]) + S[0]
6
7 print(recursion_1("Python"))
```

Programme 2 :

```
1 def recursion_2(L) :
2     if len(L) == 1 :
3         print(L[0])
4     else :
5         recursion_2(L[1:])
6         print(L[0])
7         recursion_2(L[1:])
8
9 Liste = ["J","adore","Python"]
10 recursion_2(Liste)
```

>_ Solution

Programme 1 :

PythonohyP

>_ Solution

Programme 2 :

Python
adore
Python
J
Python
adore
Python

5 Fonctions, mémoire et exceptions

Question 10: (🕒 10 minutes) Fonctions et Exceptions (Python)

Écrivez une fonction `check_name()` qui prend en argument le paramètre `name` et qui affiche dans la console "Bonjour suivi du nom passé en paramètre". Levez une exception de type `ValueError` si le paramètre n'est pas de type `str` ou si c'est une chaîne de caractères vide.

💡 Conseil

Pour trouver le type d'une variable, on utilise la méthode `type(variable)`.

>_ Solution

Python :

```
1 def check_name(nom):
2     if type(nom) != str or nom == "":
3         raise ValueError("Invalid type of argument")
4     else:
5         print("Bonjour", nom)
```

Question 11: (🕒 5 minutes) Gestion d'exceptions (Java)

Quel mot clé est utilisé pour déclencher explicitement une exception ?

1. catch
2. throw
3. raise
4. try

>_ Solution

Le mot clé pour déclencher une erreur en Java est throw.

Question 12: (🕒 5 minutes) Gestion d'exceptions (Java)

Qu'affiche le programme suivant ?

```
1 public class Question12{
2     public static void main(String args[]) {
3         try {
4             int i = 7 / 0;
5             System.out.print("OUTPUT 1");
6         } catch (ArithmeticException e)
7         {
8             System.out.print("OUTPUT 2");
9         }
10    }
11 }
```

>_ Solution

Le programme affiche OUTPUT 2.

6 Programmation orientée objet - classes et héritage

Question 13: (🕒 15 minutes) Poker Game

Cet exercice vous demande de construire un jeu de poker en implémentant un programme Java. Celui-ci comprend trois parties importantes : un paquet de cartes (un “deck”), un joueur, et une carte, et elles devraient être représentées par trois classes respectives.

Suivez les instructions ci-dessous pour écrire le programme.

La classe Carte

- Attributs : **valeur** (int), **couleur** (int). Attention, la couleur d'une carte est représentée ici par une valeur de 0 à 3 au lieu d'une chaîne de caractères.
- Implémentez un constructeur qui prend en argument une valeur et une couleur.
- Implémentez un getter **getInfo()** qui affiche dans la console la valeur et la couleur de la carte, vous pouvez vous aider de fonctions privées **getCouleur()** et **getValeur()** pour afficher les cas particuliers.


```

1 public class Carte {
2
3     private int valeur;
4     private int couleur;
5
6     public static final int HEART = 0;
7     public static final int DIAMOND = 1;
8     public static final int SPADE = 2;
9     public static final int CLUB = 3;
10
11
12     public Carte(int valeur, int couleur){
13         ...
14     }
15
16     public void getInfo(){
17         ...
18     }
19
20     public String getValeur() {
21         ...
22     }
23
24     public String getCouleur() {
25         ...
26     }
27 }
28

```

La classe Joueur

- Attributs : **mainDeCartes** (**Carte[]**) (un tableau de carte de taille maximal 2), **balance** (**int**) (le solde total du joueur), **mise** (**int**) (la mise du joueur), **monTour** (**boolean**) (valant true si c'est au tour du joueur).
- Implémentez le constructeur qui prend en argument **deux cartes** (puis les ajoute à sa main), une balance initiale, une mise initiale de 0. Vous pouvez initialiser **monTour** comme **false** au début.
- Implémentez une fonction **miser()** qui propose une mise sur la table mais ne retourne rien (utiliser le mot **void**). Définir la nouvelle mise du joueur.
- Implémentez la méthode **montrerMain()** qui montre les cartes sur la table.
- Implémentez une méthode **gagner()** qui prend en argument la somme des gains sur la table que le joueur vient de gagner, ajoutez là à son solde.
- **Attention** : **miser()** et **gagner()** s'appliquent seulement si c'est au tour du joueur.

```

1 import java.util.ArrayList;
2
3 public class Joueur {
4
5     private ...
6     private ...
7     private ...
8     private ...
9
10
11     public Joueur(...){
12         ...
13     }
14
15     public void montrerMain(){
16         ...
17     }
18
19     public void miser(int valeur){
20         ...
21     }
22
23     public void gagner(int valeur){
24         ...
25     }
26
27 }
28

```

La classe Paquet :

- Attributs : **paquet** (`ArrayList<Carte>`) (un jeu de cartes complet), **NBR.CARTES** (`int`) (un attribut final et static (constante) ayant pour valeur **52**), **NBR.MELANGES** (`int`) (une constante qui correspond au nombre de mélanges effectués de valeur **100**).
- Dans le constructeur ne prenant aucun argument, générer le deck en créant au fur et à mesure des cartes.
- Implémenter une fonction public **melanger()** qui mélange le jeu de carte et appeler là dans le constructeur après avoir construit le jeu de carte.
- Implémenter une fonction getter **getCarte()** qui retourne la carte placée en haut de la pile et la retire du jeu.

Conseil

Utilisez `Random r = new Random();` sa fonction `nextInt()` et utilisez des index et une variable `Carte` temporaire pour pouvoir échanger les positions des cartes.

```

1  import java.util.ArrayList;
2  import java.util.Random;
3
4  public class Paquet {
5
6      private ...
7      ...
8      ...
9
10     public Paquet(){
11         ...
12     }
13
14
15     public void melanger(){
16         ...
17     }
18
19
20
21     public Carte getCarte(){
22         ...
23     }
24 }
```

>_ Solution

```
1 public class Carte {
2
3     private int valeur;
4     private int couleur;
5
6     public static final int HEART = 0;
7     public static final int DIAMOND = 1;
8     public static final int SPADE = 2;
9     public static final int CLUB = 3;
10
11
12     public Carte(int valeur, int couleur){
13         this.valeur = valeur;
14         this.couleur = couleur;
15     }
16
17     public void getInfo(){
18         System.out.println("Carte: " + getValeur() + " de " + getCouleur());
19     }
20
21     public String getValeur() {
22         String res = "";
23         switch (valeur) {
24             case 11:
25                 res = "Valet";
26                 break;
27             case 12:
28                 res = "Dame";
29                 break;
30             case 13:
31                 res = "King";
32                 break;
33             default:
34                 res = Integer.toString(valeur);
35                 break;
36         }
37     }
38     return res;
39 }
40
41     public String getCouleur() {
42         String res = "";
43         switch (couleur) {
44             case 0:
45                 res = "Coeur";
46                 break;
47             case 1:
48                 res = "Carreau";
49                 break;
50             case 2:
51                 res = "Pic";
52                 break;
53             default:
54                 res = "Trèfle";
55                 break;
56         }
57     }
58     return res;
59 }
60 }
```

>_ Solution

```
1  import java.util.ArrayList;
2
3  public class Joueur {
4
5      private Carte[] hand;
6      private int balance;
7      private boolean monTour;
8      private String nom;
9
10
11     public Joueur(Carte c1, Carte c2, int initialeBalance, String nom){
12         hand = new Carte[2];
13         hand[0] = c1;
14         hand[1] = c2;
15         monTour = false;
16         balance = initialeBalance;
17         this.nom = nom;
18     }
19
20
21     public void montrerMain(){
22         for (int i = 0; i < hand.length; i++){
23             hand[i].getInfo();
24         }
25     }
26
27     public void miser(int valeur ){
28         if (monTour && balance-valeur > 0 ){
29             balance -= valeur;
30             System.out.println("Le joueur " + nom + " vient de miser: " + Integer.toString(valeur));
31             System.out.println("La nouvelle Balance: " + Integer.toString(balance));
32         } else {
33             System.out.println("Pas assez d'argent!");
34         }
35     }
36 }
37
38 public void gagner(int valeur){
39     if (monTour) {
40         this.balance += valeur;
41     }
42 }
43 }
44 }
```

>_ Solution

```
1  import java.util.ArrayList;
2  import java.util.Random;
3
4  public class Paquet {
5
6      private ArrayList<Carte> paquet;
7      public static final int NBR_CARTES = 52;
8      public static final int NBR_MELANGES = 100;
9
10     public Paquet(){
11         this.paquet = new ArrayList<>();
12
13         for ( int couleur = Carte.HEART; couleur <= Carte.CLUB; couleur++ ) {
14             for (int valeur = 1; valeur <= 13; valeur++) {
15                 paquet.add(new Carte(valeur, couleur));
16             }
17         }
18         melanger();
19     }
20
21
22     public void melanger(){
23         int index_1, index_2;
24
25         Random generator = new Random();
26         Carte temp;
27
28         for (int i=0; i<NBR_MELANGES; i++) {
29             index_1 = generator.nextInt( paquet.size() - 1 );
30             index_2 = generator.nextInt( paquet.size() - 1 );
31             temp = paquet.get( index_2 );
32             paquet.set( index_2 , paquet.get( index_1 ) );
33             paquet.set( index_1, temp );
34         }
35     }
36
37 }
38
39
40 public Carte getCarte(){
41     return paquet.remove(0);
42 }
43
44
45 }
```

Question 14: (🕒 15 minutes) Un jeu de rôle avec des personnages

Dans cet exercice, vous allez mettre en place un simple jeu de rôle. Le concept d'héritage sera très utile dans notre jeu de rôle étant donné que les différentes classes de personnages possèdent certains attributs ou actions similaires. Les personnages de notre jeu sont le Guerrier, le Paladin, le Magicien et le Chasseur.

Ainsi, il semble intéressant de construire une première classe **Personnage**. Un personnage est un objet qui possède plusieurs arguments :

- **nom (String)** : le nom du personnage
- **niveau (int)** : le niveau du personnage
- **pv (int)** : les points de vie du personnage
- **vitalite (int)** : la vitalité (ou santé) du personnage
- **force (int)** : la force du personnage
- **dexterite (int)** : la dextérité du personnage
- **endurance (int)** : l'endurance du personnage
- **intelligence (int)** : l'intelligence du personnage

Suivez les instructions ci-dessous pour compléter le programme.

- Implémentez le constructeur de la classe **Personnage** qui prend tous les attributs cités ci-dessus en argument.
- Il peut être intéressant d’afficher les caractéristiques de votre personnage. Après avoir implémenté les getters utiles, implémentez une méthode **getInfo()** dans la classe **Character** qui affiche des informations du personnage dans la console.
- Chaque personnage de ce jeu a un compteur de leur vie restante. Celui-ci va être manipulé par un **setter**. Définissez le **setter** dans la classe **Personnage**.
- Implémentez les classes **Guerrier**, **Paladin**, **Magicien** et **Chasseur** qui héritent de **Personnage** en écrivant tout d’abord leur constructeur respectif.

```

1  import java.util.Random;
2
3  public abstract class Personnage {
4
5      private ...
6      ...
7
8      public Personnage(...){
9          this.nom = ...
10         ...
11     }
12
13     public void getInfo() {
14         ...
15     }
16     ...
17     ...
18     ...
19 }

```

- Pour chacun des personnages, implémentez une méthode **attaqueBasique()** qui prend un autre personnage en argument et ne retourne rien. Celle-ci crée une attaque de votre choix en fonction des caractéristiques des personnages (ex : l’attaque du guerrier dépendra de sa force, l’attaque du chasseur de son endurance etc.), et détermine les points de vie restants en soustrayant la gravité de l’attaque à la vitalité du personnage. Affichez le nom de celui que vous avez attaqué et ses points de vie restants.
- Les méthodes communes à toutes les sous-classes, doivent être déclarées abstraites dans la classe parente **Personnage**. Changer cette classe pour qu’elle soit maintenant abstraite avec la/les méthode(s) abstraite(s) correspondante(s).
- **Attention** : Utiliser un “setter” pour réduire les **pv** de l’autre personnage.

```

1
2  public class Guerrier extends Personnage {
3
4  }
5
6  public class Magicien extends Personnage {
7
8  }
9
10 public class Paladin extends Personnage {
11
12 }
13
14 public class Chasseur extends Personnage {
15
16 }

```

>_ Solution

```
1  import java.util.Random;
2
3  public abstract class Personnage {
4
5      private String nom;
6      private int niveau;
7      private int pv;
8
9      private int vitalite;
10     private int force;
11     private int dexterite;
12     private int endurance;
13     private int intelligence;
14
15
16     public Personnage(String nom, int niveau, int pv, int vitalite, int force, int dexterite, int endurance, int
        intelligence){
17         this.nom = nom;
18         this.niveau = niveau;
19         this.pv = pv;
20
21         this.vitalite = vitalite;
22         this.force = force;
23         this.dexterite = dexterite;
24         this.endurance = endurance;
25         this.intelligence = intelligence;
26     }
27 }
28
29 public void getInfo() {
30     System.out.println("Nom" + getNom());
31     System.out.println("Niveau" + getNiveau());
32     System.out.println("Vitalité" + getVitalite());
33     System.out.println("Force" + getForce());
34     System.out.println("Dexterité" + getDexterite());
35     System.out.println("Endurance" + getEndurance());
36     System.out.println("Intelligence" + getIntelligence());
37 }
38
39
40 public abstract void attaqueBasique(Personnage other);
41
42 public String getNom() {
43     return nom;
44 }
45
46 public int getIntelligence() {
47     return intelligence;
48 }
49
50 public int getNiveau() {
51     return niveau;
52 }
53
54 public int getPv() {
55     return pv;
56 }
57
58 public int getVitalite() {
59     return vitalite;
60 }
61
62 public int getForce() {
63     return force;
64 }
65
66 public int getDexterite() {
67     return dexterite;
68 }
69
70 public int getEndurance() {
```

>_ Solution

```
71     return endurance;
72 }
73
74 public void setPv(int pv) {
75     if (pv > 0){
76         this.pv = pv;
77     }
78     else{
79         this.pv = 0;
80     }
81 }
82 }
```


>_ Solution

```
1 public class Guerrier extends Personnage {
2
3     public Guerrier(String nom, int niveau, int pv, int vitalite, int force, int dexterite, int endurance, int
        intelligence) {
4         super(nom, niveau, pv, vitalite, force, dexterite, endurance, intelligence);
5     }
6
7     public void attaqueBastique(Personnage other){
8         int attaque = (int) (getForce()*0.5);
9
10        other.setPv(other.getVitalite()-attaque);
11
12        System.out.println(this.getNom() + " attaque " + other.getNom());
13        System.out.println("Basic attaque og : " + attaque);
14        System.out.println(this.getNom() + " PV:" + this.getPv());
15        System.out.println(other.getNom() + " PV:" + other.getPv());
16    }
17 }
18
19
20 public class Magicien extends Personnage {
21
22
23     public Magicien(String nom, int niveau, int pv, int vitalite, int force, int dexterite, int endurance, int
        intelligence) {
24         super(nom, niveau, pv, vitalite, force, dexterite, endurance, intelligence);
25     }
26     @Override
27     public void attaqueBastique(Personnage other){
28         int attaque = (int) (getIntelligence()*0.5);
29
30        other.setPv(other.getVitalite()-attaque);
31
32        System.out.println(this.getNom() + " attaque " + other.getNom());
33        System.out.println("Basic attaque og : " + attaque);
34        System.out.println(this.getNom() + " PV:" + this.getPv());
35        System.out.println(other.getNom() + " PV:" + other.getPv());
36    }
37 }
38
39 }
40
41 public class Paladin extends Personnage {
42
43     public Paladin(String nom, int niveau, int pv, int vitalite, int force, int dexterite, int endurance, int
        intelligence) {
44         super(nom, niveau, pv, vitalite, force, dexterite, endurance, intelligence);
45     }
46
47
48     @Override
49     public void attaqueBastique(Personnage other){
50         int attaque = (int) (getEndurance()*0.7);
51
52        other.setPv(other.getVitalite()-attaque);
53
54        System.out.println(this.getNom() + " attaque " + other.getNom());
55        System.out.println("Basic attaque og : " + attaque);
56        System.out.println(this.getNom() + " PV:" + this.getPv());
57        System.out.println(other.getNom() + " PV:" + other.getPv());
58    }
59 }
60 }
61
62
63 public class Chasseur extends Personnage {
64
65     public Chasseur(String nom, int niveau, int pv, int vitalite, int force, int dexterite, int endurance, int
        intelligence) {
66         super(nom, niveau, pv, vitalite, force, dexterite, endurance, intelligence);
67     }
68 }
```

>_ Solution

```
68
69 @Override
70 public void attaqueBasique(Personnage other){
71     int attaque = (int) (getDexterite()*0.5);
72
73     other.setPv(other.getVitalite()-attaque);
74
75     System.out.println(this.getNom() + " attaque " + other.getNom());
76     System.out.println("Basic attaque og : " + attaque);
77     System.out.println(this.getNom() + " PV:" + this.getPv());
78     System.out.println(other.getNom() + " PV:" + other.getPv());
79
80 }
81 }
```

7 Quiz général

7.1 Python

Question 15: (🕒 2 minutes)

En Python, 'Hello' équivaut à "Hello".

1. - Vrai
2. - Faux

>_ Solution

Vrai : En Python, les doubles guillemets et les guillemets sont équivalents.

Question 16: (🕒 2 minutes)

Dans une fonction, nous pouvons utiliser les instructions `print()` ou `return`, elles ont le même rôle.

1. - Vrai
2. - Faux

>_ Solution

Faux

`print()` permet uniquement d'afficher un message dans la console. Autrement dit, `print()` sert à communiquer un message à l'utilisateur final du programme, celui-ci n'ayant pas accès au code.

`return` est une déclaration qui s'utilise à l'intérieur d'une fonction pour renvoyer le résultat de la fonction lorsqu'elle a été appelée. Exemple : la fonction `len(L)` renvoie la longueur de la liste L.

Admettons que nous ayons une fonction qui renvoie le double d'un nombre. Dans notre programme, nous souhaitons ajouter le résultat de cette fonction à un nombre quelconque. En Python, notre programme ressemblera à ça :

```
1 def double_nombre(nombre):
2     return nombre ** 2
3
4
5 nombre1 = 2
6 resultat = nombre1 + double_nombre(2)
7 print(resultat)
```

Notre programme renverra une erreur de type `TypeError: unsupported operand type(s) for +: 'int' and 'NoneType'` si on remplace la ligne 2 par `print(nombre**2)` car la fonction `double_nombre` renverra `None`.

Question 17: (🕒 2 minutes)

Lorsqu'on fait appel à une fonction, les arguments doivent nécessairement avoir le(s) même(s) noms tel(s) que définit dans la fonction. Exemple :

```
1 def recherche_lineaire(Liste, x):
2     for i in Liste:
3         if i == x:
4             return x in Liste
5     return -1
6
7 Liste = [1,3,5,7,9]
8 x = 3
9
10 recherche_lineaire(Liste,x)
```

1. - Vrai
2. - Faux

>_ Solution

Faux

Le nom des variables données en argument n'a aucune importance tant que le type de variable est respecté. Dans notre exemple, la fonction s'attend à recevoir une **liste** comme premier argument et un **entier** comme deuxième argument. Ici, nous aurions pu nommer la liste **nbr_impair** et x **valeur** et appeler la fonction **recherche_lineaire(nbr_impair, valeur)**

Question 18: (🕒 2 minutes)

Si le programme Python contient une erreur, celle-ci sera détectée avant l'exécution du programme.

1. - Vrai
2. - Faux

>_ Solution

Faux : En Python, les erreurs sont détectées pendant l'exécution du programme.

Question 19: (🕒 2 minutes)

Il est possible de faire appel à une fonction définie "plus bas" dans le code sans que cela ne pose problème.

```
1 import math
2
3 nombre_decimal_pi(4)
4
5 def nombre_decimal_pi(int):
6     return round(math.pi,int)
```

1. - Vrai
2. - Faux

>_ Solution

Faux : À l'exception des fonctions intégrées (il s'agit des fonctions déjà intégrées au langage Python telles que **print()**, **len()**, **abs()**, etc...). Dans les langages suivant une logique de programmation impérative (c'est le cas de Java et Python), une fonction doit nécessairement être définie **avant** d'être appelée. https://fr.wikipedia.org/wiki/Programmation_imp%C3%A9rative

Question 20: (🕒 5 minutes)

Les trois fonctions suivantes renvoient-elles systématiquement des résultats identiques ?

Les fonctions sont censées retourner le nombre **pi** avec le nombre de décimales (au moins une et au maximum 15) indiqué en paramètre.

<pre> 1 import math 2 3 def nombre_decimal_pi(int): 4 if int > 15: 5 int = 15 6 elif int < 0: 7 int = 1 8 resultat = round(math.pi,int) 9 return resultat 10 11 print(nombre_decimal_pi(-2)) 12 print(nombre_decimal_pi(4)) 13 print(nombre_decimal_pi(20)) </pre>	<pre> 1 import math 2 3 def nombre_decimal_pi(int): 4 if int > 15: 5 resultat = round(math.pi,15) 6 elif int < 0: 7 resultat = round(math.pi,1) 8 else: 9 resultat = round(math.pi,int) 10 return resultat 11 12 print(nombre_decimal_pi(-2)) 13 print(nombre_decimal_pi(4)) 14 print(nombre_decimal_pi(20)) </pre>	<pre> 1 import math 2 3 def nombre_decimal_pi(int): 4 if int > 15: 5 return round(math.pi,15) 6 elif int < 0: 7 return round(math.pi,1) 8 else: 9 return round(math.pi,int) 10 11 12 print(nombre_decimal_pi(-2)) 13 print(nombre_decimal_pi(4)) 14 print(nombre_decimal_pi(20)) </pre>
--	--	---

1. - Vrai
2. - Faux

>_ Solution

Vrai : Les trois fonctions produisent des résultats identiques. Si besoin, exécutez le code dans IntelliJ.

7.2 Java

Question 21: (🕒 3 minutes)

Observez les deux programmes suivants en Java. Lequel a-t-il une bonne structure et peut être compilé sans erreur ?

<pre> 1 //Programme A 2 public class Main { 3 4 public static void main(String[] args) { 5 ma_function(); 6 autre_fonction(); 7 8 9 static void ma_function(){ 10 System.out.println("Voici ma fonction!"); 11 } 12 13 static void autre_fonction(){ 14 System.out.println("Une autre fonction!"); 15 } 16 } 17 } </pre>	<pre> 1 //Programme B 2 public class Main { 3 4 public static void main(String[] args) { 5 ma_function(); 6 autre_fonction(); 7 } 8 9 static void ma_function(){ 10 System.out.println("Voici ma fonction!"); 11 } 12 13 static void autre_fonction(){ 14 System.out.println("Une autre fonction!"); 15 } 16 } </pre>
--	---

1. Programme A
2. Programme B

>_ Solution

Le programme B

Le fichier dans son ensemble représente une classe Java, ici la classe s'appelle **Main** (Vous aurez plus d'informations sur le fonctionnement des classes lorsque nous aborderons la programmation orientée objet). À l'intérieur de cette classe se trouve la méthode **public static void main()**, il s'agit de la porte d'entrée de notre programme, celle que l'on exécute et celle dans laquelle nous rédigeons notre code.

Les autres fonctions, qui peuvent être appelées, se définissent au sein de la classe au même échelon que la fonction **public static void main()** comme dans le programme B ci-dessus.

Question 22: (🕒 2 minutes) Exercice 1

L'indentation des lignes de code en Java est aussi importante qu'en Python.

1. - Vrai
2. - Faux

>_ Solution

Faux

En Java, le compilateur ne prend pas en compte l'indentation pour interpréter le programme, il comprend la structure à l'aide des parenthèses, des accolades et encore des point-virgules qui indiquent la fin d'une instruction. Toutefois, l'indentation est un aspect important de la programmation car elle sert à bien structurer visuellement votre code.

En Python, l'indentation définit la structure de votre code. Elle est donc indispensable pour la bonne interprétation et la bonne exécution de votre programme.