# Interrogation d'informatique
# EURINSA-AMERINSA 1ère année - Avril 2018-19

**Durée totale :** *1h30*
**Documents autorisés :** *Une feuille A4 recto-verso.*

- Respectez les **conventions de codage** et choisissez des **noms de variables expressifs**.
- Barème indicatif. Une mauvaise présentation peut faire perdre jusqu'à 1 point.
- Le sujet est sur 5 pages, et les 2 exercices sont distincts.

## 1 Compréhension d'un programme (5 points)

Les deux classes `Restaurant` et `Test`, données en annexe, ne compilent pas. Sur l'annexe, signalez la liste des erreurs contenues dans ces classes et proposez une correction.

**Remarque** : On prend l'hypothèse que les déclarations des attributs de la classe `Restaurant` **sont correctes**.

## 2 Écriture de programmes : Objets - "Tourne tourne petit moulin" (15 points)

Dans cet exercice, on s'intéresse à d'anciens moulins à eau, qui peuvent être rénovés et équipés afin de produire de l'électricité. Le but est d'étudier la rentabilité de l'installation d'un tel système pour le meunier/la meunière. Un meunier est un propriétaire de moulin. Le schéma simplifié d'un moulin à eau est donné sur la figure 1. Le prix d'électrification est la somme d'argent nécessaire pour équiper le moulin afin qu'il puisse produire de l'électricité.

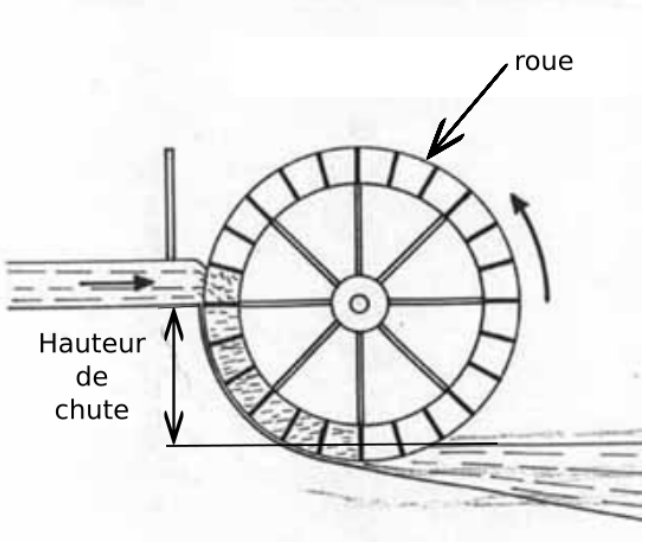


FIGURE 1 – Schéma de la roue d'un moulin à eau

Vous devez écrire les classes `Moulin`, `Meunier` et une classe utilisatrice `Test`. Le diagramme de classe du programme est donné sur la figure 2.

**Remarques :**
- Les 3 classes à écrire peuvent être écrites **dans un ordre quelconque**.
- Chaque classe peut être écrite en supposant que les autres classes et méthodes existent.
- **Les questions ne sont pas classées par ordre de difficulté croissante.**

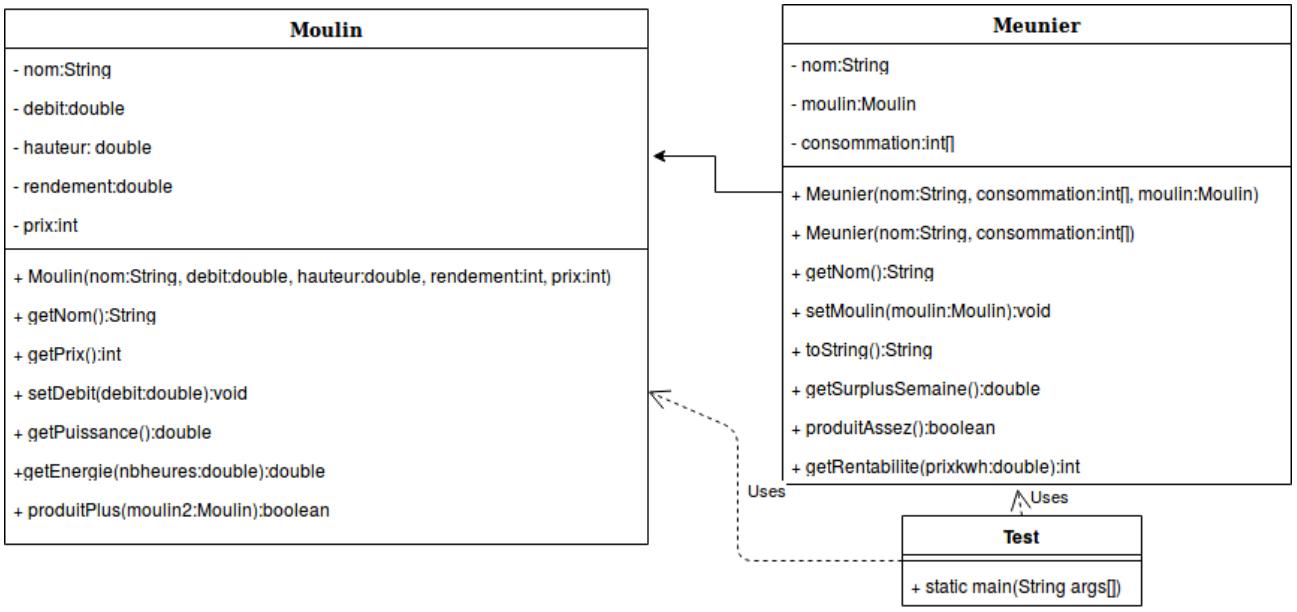


FIGURE 2 – Schéma UML du programme final

## 2.1 Classe `Test` (3 points)

En vous aidant du diagramme de classe pour déterminer les méthodes adéquates, écrivez le code de la classe utilisatrice `Test` qui doit effectuer les actions suivantes :

- Créer le moulin "Joli vent" ayant un débit de $3.6m^3/s$, une hauteur de chute de 1.1m, un rendement de 60%, et un prix d'électrification de 125000 euros *(Ces valeurs sont à transmettre au constructeur telle quelle, sans mettre les unités)*.
- Créer le moulin "Petit Caillou" ayant un débit de $2.5m^3/s$, une hauteur de chute de 2.3m, un rendement de 65%, et un prix d'électrification de 150000 euros.
- Créer le meunier "Alice" qui possède le moulin *Joli Vent* et dont la consommation hebdomadaire est : $[7, 7, 8, 5, 6, 2, 2]$.
- Afficher la description textuelle de Alice.
- Créer le meunier "Maxime" dont la consommation est : $[2, 2, 5, 4, 3, 9, 9]$.
- Affecter ensuite à *Maxime* le moulin *Petit Caillou*.
- En considérant que le prix d'un kilowattheure est de 0.14 euros, déterminer si c'est le moulin de Alice ou celui de Maxime qui sera rentabilisé en premier [1].

## 2.2 La classe `Moulin` (5.5 points)

Un moulin est caractérisé par :

- Son nom
- Le débit de la rivière qui le traverse
- La hauteur de la chute d'eau sur la roue
- Son rendement, c'est-à-dire le pourcentage de l'énergie produite qui est réellement récupérée par le système
- Le prix de son aménagement pour électrification.

On considère que l'on a déjà déclaré la classe par `public class Moulin{...}`.

1. La méthode `getRentabilite` renvoie le nombre de semaines avant rentabilité de l'électrification du moulin. Elle renvoie -1 si le moulin n'est pas rentabilisable par ce meunier.

*(Q2.2.1) Déclarez les attributs de la classe* `Moulin`

*(Q2.2.2) Écrivez un constructeur ayant 5 paramètres : le nom du moulin, son débit, sa hauteur de chute, son rendement, et son prix. Le rendement est un pourcentage qui doit être passé au constructeur sous forme d'entier (entre 0 et 100), et divisé par cent pour enregistrer ce pourcentage sous forme de double (entre 0 et 1). Si le rendement donné est supérieur à 100, le rentrer comme égal à 100.*

Pour avoir des informations sur le moulin, on veut pouvoir connaitre son nom et son prix. Le débit pouvant changer en fonction de travaux sur la rivière sur laquelle est installée le moulin, on veut pouvoir le mettre à jour.

*(Q2.2.3) Écrivez les getters* `getNom`*, et* `getPrix ()` *et le setter* `setDebit ()`

La puissance instantanée de l'électricité (en kilowatts) produite par le moulin se calcule en fonction de son débit et de sa hauteur de chute. Elle est donnée par la formule :

$$puissance = rendement \times debit \times hauteur \times g$$

Où $g$ est la constante gravitationnelle. On prendra ici $g = 9.81$.

*(Q2.2.4) Écrivez la méthode* `getPuissance`*, qui calcule la puissance instantanée du moulin*

On veut également pouvoir calculer l'énergie produite par le moulin sur un nombre d'heures donné (en kilowattheures).

$$energie = puissance \times nombre\,d'heures$$

*(Q2.2.5) Écrivez la méthode* `getEnergie(double nbHeures)`

Pour un potentiel acheteur de moulin, il est important de savoir, entre deux moulins, lequel peut produire le plus d'électricité.

*(Q2.2.6) Écrivez la méthode* `produitPlus(Moulin moulin2)`*, qui renvoie vrai si le moulin auquel elle s'applique a une puissance instantannée strictement plus grande que le moulin moulin2.*

## 2.3 La classe `Meunier`(6.5 points)

Le meunier, propriétaire du moulin, est caractérisé par :
— Son nom
— Son moulin
— Sa consommation électrique journalière, pour alimenter ses différents appareils électriques (en kilowattheures). On représentera cette consommation par un tableau de 7 cases, ou chaque case représente la consommation journalière (la case 0 représente le lundi).

*(Q2.3.1) Déclarez les attributs de la classe* `Meunier`

*(Q2.3.2) Écrivez le constructeur de la classe* `Meunier` *ayant comme paramètres un nom, un* `Moulin`*, et un tableau de consommation électrique.*

*(Q2.3.3) Écrivez un deuxième constructeur de la classe* `Meunier` *ayant 2 paramètres : un nom et un tableau de consommation électrique. Le moulin est alors initialisé à* `null`*.*

*(Q2.3.4) Écrivez le setter* `setMoulin`*, qui permet d'affecter un Moulin à un meunier.*

On veut pouvoir récupérer une description textuelle d'un meunier sous la forme :

```
Alice possède le moulin JoliVent. Sa consommation hebdomadaire est la suivante:
7 | 7 | 8 | 5 | 6 | 2 | 2 |
```

*(Q2.3.5) Écrivez la méthode* `toString`*, qui renvoie la description textuelle d'un meunier.*

Lorsque le moulin produit plus d'électricité qu'il n'en consomme, le meunier peut la revendre à EDF. À l'inverse, il peut en acheter lorsqu'il n'en produit pas assez. On considère que le moulin produit de l'électricité en permanence, c'est à dire 24 heures sur 24.

*(Q2.3.6) Écrivez la méthode* `getSurplusSemaine` *qui calcule, pour une semaine, la différence entre la production du moulin (calculée par la méthode* `getEnergie`*) et la consommation électrique du meunier.*

Pour que son installation soit rentable, un meunier doit consommer, sur une semaine, moins d'électricité que son moulin en produit.

*(Q2.3.7) Écrivez la méthode* `produitAssez` *qui renvoie vrai si le meunier consomme moins que son moulin ne produit.*

Afin d'amortir le prix conséquent de l'électrification du moulin, le meunier veut savoir en combien de temps son investissement initial sera amorti, étant donné le prix de vente du kilowattheure. On considère que le prix du kilowattheure reste constant.

*(Q2.3.8) Écrivez la méthode* `getRentabilite(double prixKwh)`*, qui détermine au bout de combien de semaines le meunier aura rentabilisé l'électrification de son moulin, en fonction d'un prix du kilowattheure. Elle renvoie -1 si le moulin ne sera jamais rentable.*

# 3 Annexe

**Cette feuille doit être détachée et rendue avec votre copie, avec les corrections apportées. Pensez à marquer votre nom, prénom et numéro de groupe.**

Nom : Prénom : Groupe :

```
1  public class Restaurant {
2      private int[] tailleTables; // Nombre de personnes pouvant etre accueillies par une table i
3      private boolean[] tablesOccupees; // Memorise si une table est occupee (true) ou non (false)
4
5      public Restaurant(int[7] tables) {
6          this.tailleTables = tailleTables;
7          tablesOccupees = new boolean[tailleTables.length];
8
9          for(int i=0 ; i<tailleTables.length ; i++){
10             resto.libererTable(i);
11         }
12     }
13     public int getTailleUneTable(int t) {
14         return tailleTables[t];
15     }
16     public void libererTable(table){
17         tablesOccupees[table]=false;
18     }
19     public void toString(){
20         String res = "Etat d'occupation des tables du restaurant : \n";
21         for(int i=0;i<tablesOccupees.length ; i++) {
22             res = res+"  Table "+i+" occupee ? "+tablesOccupees[i]+"\n";
23         }
24         return res;
25     }
26
27     // Renvoie vrai si et seulement si la table est libre et peut accueillir le nb de personnes
28     public boolean estTablePossible(int table, int nbPersonnes) {
29         return !tablesOccupees[table] && tailleTables[table] >= nbPersonnes;
30     }
31
32     /* Choisit une table pour accueillir le groupe et renvoie le numero de cette table.
33        Si pas de table dispo compatible et disponible, renvoie -1. */
34     public boolean placerGroupe(int tailleGroupe) {
35         int proposition = -1;
36         for(int i=0 ; i< tailleTables.length ; i++) {
37             if(estTablePossible(i, nbPersonnes)) {
38                 if(proposition == -1 || tailleTables[i] < tailleTables[proposition]) {
39
40                     proposition = i;
41                 }
42             }
43         }
44         if(proposition > -1) {
45             tablesOccupees[proposition]=true;
46         }
47         return proposition;
48     }
49 }
```

```
1  public class Test {
2      public static void main(String[] args){
3          int[] tables = {2, 2, 4, 4, 6, 10, 2};
4          Restaurant resto = new Restaurant("Brasserie de la place", tables);
5          int tailleGroupe=7;
6          int table = placerGroupe(tailleGroupe);
7
8          if(table != -1) {
9              System.out.println("Merci pour la table.");
10             if(getTailleUneTable(table) >= tailleGroupe +2) {
11                 System.out.println("Et super grande :D");
12             }
13         } else {
14             System.out.println("Allons voir ailleurs.");
15         }
16     }
17 }
```