Fms Academy 2022
Formation Java Spring Angular
Module 3 : Algorithmique

mohamed.el-babili@fms-ea.com

Sommaire

- Algorithme & Adn
- Plan ou recette d'un algorithme
- Un programme, c'est du gâteau !
- Variables et Entrées/sorties
- Programme complet avec automatisation du process
- Algorithmique : méthodo pour réaliser des algos (1) (2)
- Déboguer : comment résoudre les imprévus ?
- QQ critères d'une application de qualité
- Répéter pour maitriser
- Comment optimiser un algorithme + qualité du code ?
- Abstraction/Imagination et les rythmes d'apprentissage
- Paradigme de programmation
- C'est quoi le métier de développeur ?
- Pour un cerveau en bonne santé
- Ressources

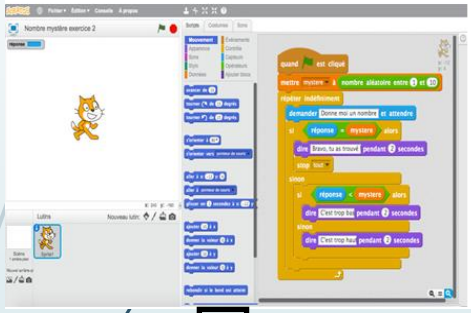
Algorithme & Adn

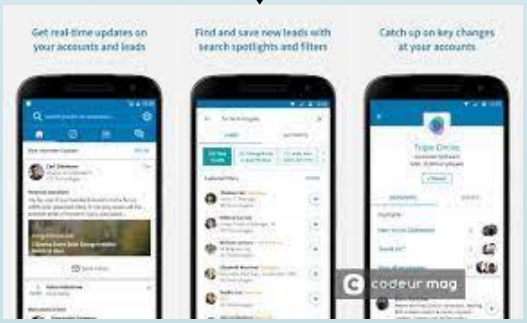
Variables

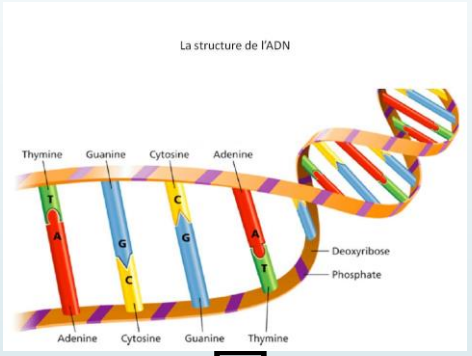
Entrées/Sorties

Tests

Boucles







Plan ou recette d'un algorithme

- Pour réaliser un algorithme, il faut un plan ou recette qui répond d'abord à un besoin, ici on pourrait exprimer **la volonté de manger un gâteau**

- La première étape va consister à réunir les **ingrédients** dont nous aurons besoin selon des quantités précises

Ingrédients :

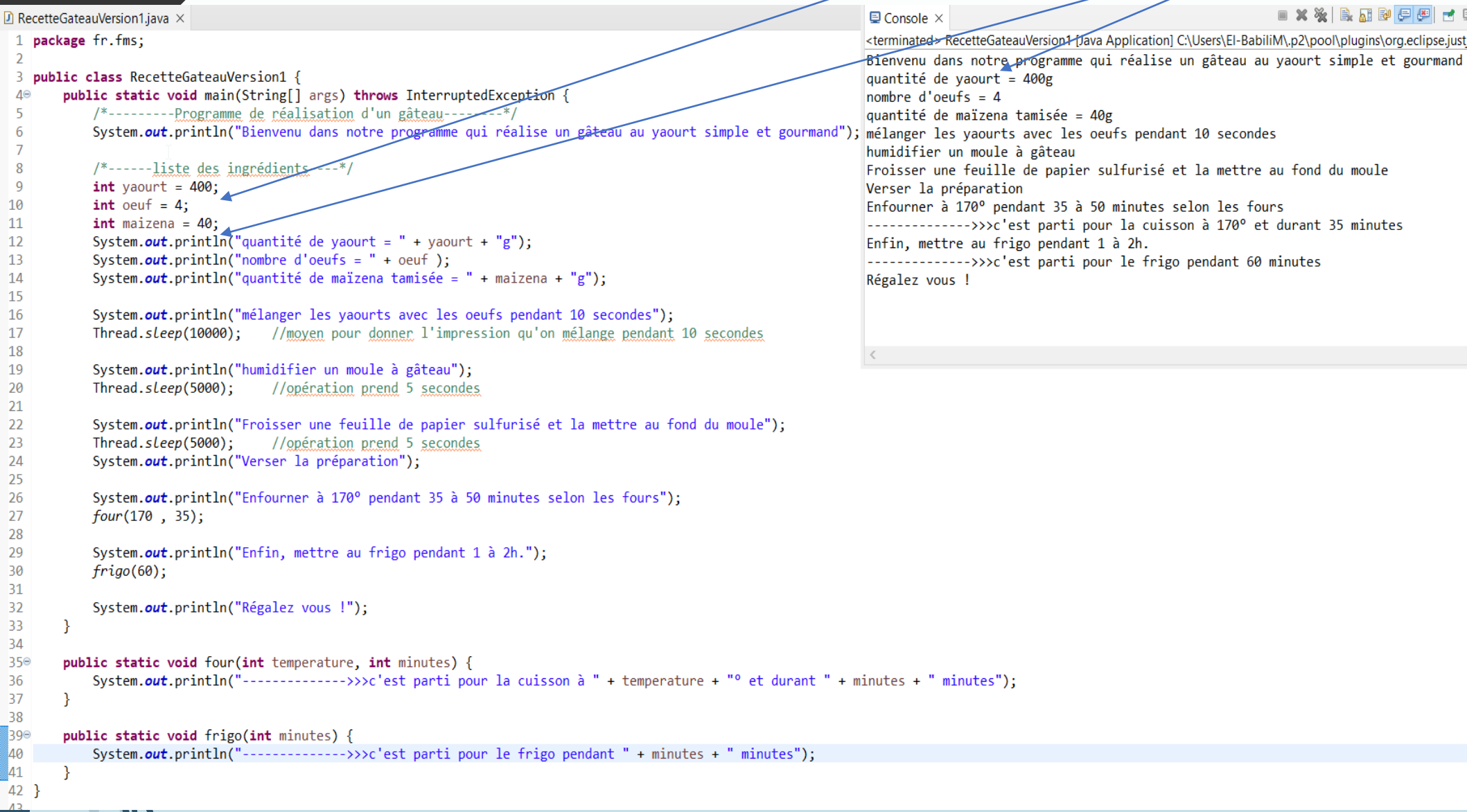
- 400 g de yaourt nature sucré ou yaourt aromatisé
- 4 œufs
- 40 g de Maïzena tamisée

- Ensuite il faut suivre la recette **étape par étape** afin d'arriver au résultat final et à la dégustation

Mélanger le yaourt avec les œufs
Ajouter la Maïzena tamisée
Humidifier un moule à gâteau
Froisser une feuille de papier sulfurisé et la mettre au fond du moule
Verser la préparation
Enfourner à 170 °C pendant environ 35 à 50 min selon les fours
Mettre au frigo 1 à 2 h

Def : un Algorithme est une suite d'instructions compréhensibles, qui une fois exécutée correctement, conduit à un résultat donné

Un programme, c'est du gâteau



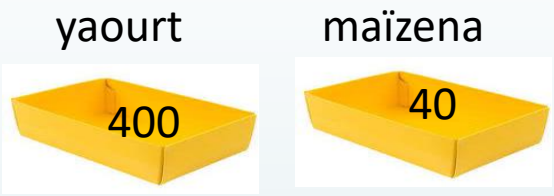

```java
package fr.fms;

public class RecetteGateauVersion1 {
    public static void main(String[] args) throws InterruptedException {
        /*---------Programme de réalisation d'un gâteau---------*/
        System.out.println("Bienvenu dans notre programme qui réalise un gâteau au yaourt simple et gourmand");

        /*------liste des ingrédients ----*/
        int yaourt = 400;
        int oeuf = 4;
        int maizena = 40;
        System.out.println("quantité de yaourt = " + yaourt + "g");
        System.out.println("nombre d'oeufs = " + oeuf );
        System.out.println("quantité de maïzena tamisée = " + maizena + "g");

        System.out.println("mélanger les yaourts avec les oeufs pendant 10 secondes");
        Thread.sleep(10000);    //moyen pour donner l'impression qu'on mélange pendant 10 secondes

        System.out.println("humidifier un moule à gâteau");
        Thread.sleep(5000);     //opération prend 5 secondes

        System.out.println("Froisser une feuille de papier sulfurisé et la mettre au fond du moule");
        Thread.sleep(5000);     //opération prend 5 secondes
        System.out.println("Verser la préparation");

        System.out.println("Enfourner à 170° pendant 35 à 50 minutes selon les fours");
        four(170 , 35);

        System.out.println("Enfin, mettre au frigo pendant 1 à 2h.");
        frigo(60);

        System.out.println("Régalez vous !");
    }

    public static void four(int temperature, int minutes) {
        System.out.println("-------------->>>c'est parti pour la cuisson à " + temperature + "° et durant " + minutes + " minutes");
    }

    public static void frigo(int minutes) {
        System.out.println("-------------->>>c'est parti pour le frigo pendant " + minutes + " minutes");
    }
}
```

Variables et Entrées/Sorties

Variables

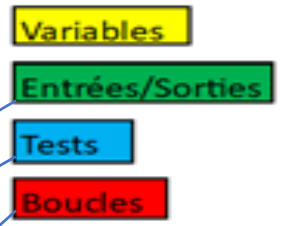
Entrées/Sorties

La notion de **variable** peut être représentée sous forme de boite accessible par son nom, dans laquelle on peut stocker la valeur d'une donnée qui dépend du type de donnée



La notion d'**entrées** dans un programme indique ce qui vient de l'extérieur (saisie clavier) vers notre programme et inversement pour les **sorties** (affiche à l'écran)

Programme complet avec Automatisation du process




```
public class RecetteGateauVersion2 {
    public static final int MIN_YAOURT = 400;
    public static final int MIN_OEUF = 4;
    public static final int MIN_MAIZENA = 40;

    public static void main(String[] args) throws InterruptedException {
        /*---------Programme de réalisation d'un gâteau---------*/
        System.out.println("Bienvenu dans notre programme qui réalise des gâteaux");

        /*------liste des ingrédients----*/
        System.out.println("nous allons vérifier si vous disposez bien des ingrédients en quantité suffisante ");
        Scanner scan = new Scanner(System.in);
        System.out.println("saisissez la quantité de yaourt dont vous disposez ?");
        int yaourt = scan.nextInt();
        if (yaourt >= MIN_YAOURT) { // avec 1000g on peut faire 2 gateaux / avec 200g on peut en faire aucun
            System.out.println("saisissez le nombre d'oeufs dont vous disposez ?");
            int oeuf = scan.nextInt();
            if (oeuf >= MIN_OEUF) { // avec 9oeufs on peut faire 2 gateaux / avec 1 oeuf, on peut en faire aucun
                System.out.println("saisissez la quantité de Maizena dont vous disposez ?");
                int maizena = scan.nextInt();
                if (maizena >= MIN_MAIZENA) {
                    fabriquonsDesGateaux(yaourt, oeuf, maizena);
                } else  System.out.println("vous n'avez pas suffisamment de Maizena");
            } else  System.out.println("vous n'avez pas suffisamment d'oeufs");
        } else  System.out.println("vous n'avez pas suffisamment de yaourt");
    }

    private static void fabriquonsDesGateaux(int yaourt, int oeuf, int maizena) throws InterruptedException {
        //Copmbien de gateaux pouvons fabriquer ?
        int nbGateauYaourt = yaourt / MIN_YAOURT;
        int nbGateauOeuf = oeuf / MIN_OEUF;
        int nbGateauMaizena = maizena / MIN_MAIZENA;

        int nbGateau = 0;
        if(nbGateauYaourt <= nbGateauOeuf) nbGateau = nbGateauYaourt;
        else nbGateau = nbGateauOeuf;
        if(nbGateau >= nbGateauMaizena) nbGateau = nbGateauMaizena;

        System.out.println("le nombre total de gateau que nous pouvons réaliser : " + nbGateau);
        for(int i=0 ; i<nbGateau ; i++) {
            System.out.println("Réalisation du gâteau numéro " + (i+1));
            System.out.println("mélanger les yaourts avec les oeufs pendant 10 secondes");
            Thread.sleep(10000);     //moyen pour donner l'impression qu'on mélange pendant 10 secondes

            System.out.println("humidifier un moule à gâteau");
            Thread.sleep(5000);      //opération prend 5 secondes

            System.out.println("Froisser une feuille de papier sulfurisé et la mettre au fond du moule");
```

NB : Nous avons utilisé dans notre exemple la notion de fonction, qui permet d'apporter une meilleure lisibilité ici en déportant une partie de l'algorithme, noter qu'on aurait pu s'en passer.

Algorithmique : méthodo pour réaliser des algos (1)

- Un **algorithme** sert à répondre à un **besoin** : fabriquer un gâteau
- Il est **constitué** d'un **début** et d'une **fin**
- Il peut contenir : **variables**, **entrées/sorties**, **tests** et **boucles**
- La **qualité** d'un algorithme dépend de notre capacité à gérer correctement le(s) **cas général** et les **cas particuliers** en **anticipant** sur toutes les situations/chemins
  - Tous les ingrédients sont réunis en quantité suffisante, on peut réaliser un ou plusieurs gâteaux
  - Un ou plusieurs ingrédients manque aussi, il faut indiquer à l'utilisateur la raison pour laquelle, on ne pourra pas le satisfaire.
- **La phase de test** permet de s'assurer que notre algorithme fonctionne comme prévue et de le valider par un minimum de **couverture de test** visant à passer par tous les chemins **: jeux d'essai, test unitaire**
- La **simplicité** engendre la **lisibilité** de l'algorithme par des collègues aussi il faut utiliser des **noms explicites**, une **indentation** clair et des **commentaires** précis
- *NB : Lorsque vous sous traitez tout ou partie de votre programme à une ressource externe(ex : Scanner), elle ne doit avoir aucun secret pour vous, ça passe par une lecture précise de la documentation.*

Algorithmique : méthodo pour réaliser des algos (2)

"Tout le monde dans ce pays devrait apprendre à programmer un ordinateur, parce que cela vous apprend à réfléchir" — Steve Jobs

Tous les jours nous rencontrons des problématiques diverses perso ou prof et nous essayons d'y répondre en testant plusieurs options, perdant beaucoup de temps aussi pour être efficace : méthodes + gestion des priorités :

- *"La plus grande erreur que j'observe chez les jeunes programmeurs est de se focaliser sur la syntaxe plutôt que d'apprendre comment résoudre des problèmes"* — V.Anton Spraul

- 1/ Analyser & comprendre les besoins ou la problématique (un minimum)
  - *"Si vous n'arrivez pas à expliquer quelque chose avec des termes simples, c'est que vous ne le comprenez pas" — Richard Feynman*
  - *"La plus grande erreur que j'observe chez les développeurs juniors est de foncer tête baissé sans prendre le temps de la réflexion, le codage pur est pourtant qu'une petite partie du métier "* — M.E
- 2/ Réfléchir et Panifier la solution selon des étapes bien distinctes
  - Commencer par écrire des commentaires //étape 1 : je demande à l'utilisateur de…
- 3/ Diviser pour régner
  - découper un problème complexe en petit pb + simple, programmation modulaire…
- 4/ Syndrome de la page blanche
  - Mieux vaut faire quitte à défaire que de rester à tourner en rond
  - Ecrire en français des commentaires décrivant les étapes puis coder petit à petit

Déboguer : comment résoudre les imprévus ?

- *"L'art du débogage consiste à comprendre exactement ce que vous dites à votre programme plutôt que ce que vous pensez lui avoir dit de faire"* — Andrew Singer

- Avant d'utiliser les outils de débogage super génial qui vous aide à résoudre beaucoup de pb prenez du recul, de la hauteur, un autre point de vue au lieu de rester la tête dans le guidon. Revenir donc à l'essentiel, aux fonctionnalités principales avant de rezoomer sur les secondaires. Reprendre parfois tout depuis le début peut prendre du temps mais peut être indispensable pour comprendre que l'erreur remonte au tout début.
- Avec un déboguer, le mode pas à pas et les points d'arrêts est une manière efficace mais il ne faut se reposer entièrement sur celui-ci.
- S'agissant d'un point spécifique, stackoverflow par ex peut vous aider à condition d'être précis dans votre recherche sans quoi vous risquez de vous noyer dans l'océan d'information, aussi votre recherche doit résulter d'une véritable réflexion en amont :
  - Quelle est le périmètre technique de ce que je recherche ? Sql ou Orm/Jpa/Hybernate/Spring Data
  - Formuler clairement votre recherche : qu'est-ce que je recherche ? Pour répondre à quel besoin ?
  - Il faut comprendre comment fonctionne les moteurs de recherche afin d'utiliser les bons mots clés
  - Vous devez enfin suffisamment connaître votre périmètre technique sans avoir besoin de le maitriser !

➡ ***"Juste au moment où vous pensez avoir franchi un obstacle, un nouveau apparaît. Mais c'est ce qui rend la vie intéressante. […] La vie est un processus consistant à trouver des solutions aux obstacles, les uns après les autres. A chaque fois, vous apprendrez quelque chose. A chaque fois, vous développerez votre force, votre sagesse et votre perspective. A chaque fois, les difficultés tomberont. Jusqu'à ce que tout ce qu'il reste se résume à vous : la meilleure version de vous."* — Ryan Holiday (The Obstacle is the Way)**

QQ critères d'une application de qualité

Efficacité : un maximum de résultat pour un minimum d'effort

Fiabilité : tests unitaires et d'intégration

Robustesse : capacité à résister aux erreurs

Maintenabilité : fermée aux modifications, ouverte aux évolutions

Versionning : notamment lorsqu'on travaille en équipe

Portabilité : gestion de changement de bibliothèques ou base de donnée

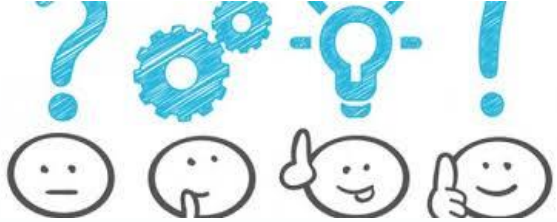
Répéter pour maitriser

- Le secret qui permet de viser l'excellence dans chaque métier reste la répétition des gestes technique
- En effet, qu'est-ce qui fait qu'un hôpital acquière une notoriété dans telle ou telle chirurgie ? Telle opération y est réalisé minimum de nombreuses fois/année
- De même, la répétition dans l'analyse des besoins exprimés ou problématique à solutionner et la réponse ou solution apportée, doit être un processus scrupuleusement suivi et suffisamment reproduit afin d'acquérir les bons réflexes
- Ceci dit, il y a des domaines ou les algo sont + ou – complexes et volumineux nécessitant une double expertise en amont en math/physique par ex :
  - Scientifique (traitement du signal…)
  - Gestion (web…)
  - Industriel (embarqué…)
  - ….

Comment optimiser un algorithme et améliorer la qualité du code ?

- Il faut de l'expérience, ce n'est pas quelque chose qui doit être visé dans un 1er temps puisque ça risque de rendre la tache plus compliquée aussi concentrez vous surtout sur :
  - Est-ce que ça fonctionne ? Est-ce que ça répond aux besoins ?
  - Est-ce que ça gère tous les cas ? Est-ce simple à comprendre ?
- Une fois le 1er cap passé, voici des pistes :
  - Utiliser les boucles uniquement quand c'est indispensable
  - Factoriser quand c'est possible en réduisant les tests par ex, ça réduit le tps de calcul
  - En minimisant l'occupation en mémoire (embarqué , éviter la récursivité…) 
- Il existe des outils pour nous imposer des règles d'écriture de code qui facilite grandement le déploiement d'une application puisque les pbs ont été détectés tôt, ex : sonar…

Abstraction/Imagination et les rythmes d'apprentissage



- Le métier a beaucoup évolué depuis 20 ans, si la pile technologique constituant votre mallette est devenu conséquente, l'autre difficulté vient des concepts abstrait qu'on ne peut se représenter qu'à l'aide de dessins de vulgarisation, de cours de pédagogues, d'imagination, de l'expérience ou du rythme d'apprentissage.

- Raisons pour lesquelles, **la prise de sens** peut arriver à des moments différents selon chacun et c'est tout à fait normal.

Paradigme de programmation

- Un paradigme de programmation fournit (et détermine) la vue qu'a le développeur de l'exécution de son programme. Par exemple, en programmation orientée objet, les développeurs peuvent considérer le programme comme une collection d'objets en interaction, tandis qu'en programmation fonctionnelle un programme peut être vu comme une suite d'évaluations de fonctions sans états.
- **Le problème vient lorsqu'on connait qu'un seul paradigme, on va répondre aux problèmes qu'en fonction de celui-ci. Ce qui peut se traduire vulgairement par l'utilisation d'une porche pour acheter une baguette au bout de la rue.**
- **En réalité, les langages évolués comme Python ou Java nous permettent d'utiliser plusieurs paradigmes. Nous allons utiliser le paradigme procédurale dans un premier temps puis nous basculerons sur celui de la Programmation orientée objet.**

C’est quoi le métier de développeur ?

- Tout le monde peut coder au même titre que tout le monde peut faire des maths à condition de bien lire les leçons et les consignes puis de faire de nombreux exercices.
  - Tous les développeurs ont commencé au niveau 0 un jour, n’est-ce pas ?
- En revanche, tous n’avancent pas au même rythme selon leur parcours et leur motivation
- Softs Skills du dev :
  - **Esprit Logique/Méthodique/Analytique, Bon sens/Intuition, Aimer résoudre des problématiques complexes, échanger et confronter ses idées, Sens de la communication et du service, Capacité de travailler en équipe, respect des consignes/normes, curieux, Agile/Adaptable, innovant, Aime apprendre de nouvelles choses, Esprit synthétique, prise de décision, créativité, sens de l’organisation, Esprit logique de déduction, Savoir quoi chercher comment et où, Motivé, aime les challenges, persévérant, innovant, autonomie, anglophone**
- Comment faire pour les développer ?
  - Il faut se remettre au sport ... Intellectuel
  - La moindre occasion pour coder doit être saisie
  - Il y aura une phase de décrassage, de résistances, de doutes et de détermination
  - Réaliser toutes les demandes du formateur
  - Apprendre à apprendre
  - Privilégier les activités saines pour le cerveau
  - Notre capacité à se concentrer étant limité, parfois il faut savoir switcher pour éviter les migraines

Pour un cerveau en bonne santé

- Alimentation saine (huile d'olive, poissons, fruits, légumes…) vs Sucre (drogues)
- Sport (cœur, cerveau, mémoire) vs apathie
- Méditation/relaxation/silence vs stress
- Qualité du sommeil vs écrans tard le soir
- Jeux de réflexions & jeux vidéos
- Culture (lecture, dessin, peinture, théâtre, cinéma…) vs télévision uniquement
- Multiplier les échanges sociaux et les débats vs les mêmes habitudes
- Jardiner, Cuisiner, Couturer, bricoler…
- Jouer d'un instrument
- Apprendre une langue étrangère
- Apprendre de nouvelles choses chaque jour
- Voyager même à côté (terre, montagne, mer et océan…)
- Avoir un projet professionnel vs aucun projet de vie

- *Motivation : Tout ce que vous donnez à votre corps de bon, reviendra un jour !*

Un dernier mot

- Souvent vous entendrez, les informaticiens sont des fainéants ou il ne faut **pas réinventer la roue** … C'est vrai quand il s'agira de répondre à des besoins de + en + complexe, il faut voir s'il n'existe pas un algo qui fait le job quitte à l'adapter un peu.

- Cela n'est pas valable dans votre contexte d'apprentissage, vous devez réaliser tous les algos afin de **muscler votre raisonnement informatique**.

- Ne jamais hésiter à **ouvrir le capot** pour regarder ce qu'il y a dedans afin de s'approprier les choses, **ne pas avoir peur de se tromper** c'est comme ça aussi qu'on apprend.
- Dans le contexte airbus, ce n'est pas toujours simple mais trouver un moyen de **faire un labo de test** afin de manipuler les choses sans impacter les applis.
- Le métier de développeur est fantastique avec des challenges toujours plus ambitieux, seulement parfois, il y a des taches répétitives simples qu'il faut faire un peu comme le maçon qui monte un mur…

A vous la parole

- Questions / réponses
- Comment améliorer les masterclass ?
- Challenger les stagiaires tous les jeudis
- Rendez-vous les vendredis après midi avec la promo (planning)

Ressources

- https://algo.developpez.com/cours/
- http://cours.pise.info/algo/introduction.htm
- https://medium.com/@rhunold/comment-penser-comme-un-programmateur-le%C3%A7on-pour-pour-r%C3%A9soudre-des-probl%C3%A8mes-1f374a4b399f

Next steps

- Java SE 8
- **Algorithmique**
- Git / GitHub