IPO

Initiation à la
Programmation
Objet

IGI-1202

A3P

Apprentissage
Par
Problème
de la **P**rogrammation

IGI-3007

Responsable : Denis BUREAU

IPO / A3P

Peut-être
les 2h les plus importantes
de toute l'unité !

(ou comment s'y prendre
pour réussir à coup sûr ?)

IPO / A3P

Pourquoi ?

- Pédagogie très différente (moi) ==>
  méthode de travail très différente (vous)
- Facile de réussir (si on suit la méthode)
- Facile de se planter (si on fait comme d'habitude)
- Prévenir les absents ! *(pdf sur le web)*

(Moodle ou **page E1** ou E3ST)

IPO / A3P

Vos objectifs (à court terme) :

- Comprendre et savoir mettre en œuvre les concepts de la programmation Objet.
- Savoir programmer en Java (2e niveau).
- Comprendre et savoir choisir les différentes conceptions possibles d'un programme.
- Être capable d'écrire puis programmer de petits algorithmes, par ex. sur les tableaux.
- Connaître les différences du C p/r Java
- 1ère expérience de programmation en C

IPO / A3P

Vos objectifs (à moyen terme) :

- Ne pas avoir d'handicap dans la suite de vos études (programmation partout).
- Pour pouvoir accéder à toutes les filières.

Vos objectifs (à long terme) :

- Avoir plus opportunités de carrières.
- Innover grâce à l'interdisciplinarité.

IPO / A3P

Mes objectifs :

1. >90% de validation pour tous ceux qui suivent la méthode (+ toutes les séances) **(c'était le cas l'an dernier)**
2. Beaucoup de bonnes notes (69%>12) **(c'était le cas l'an dernier)** (40%>14)
3. Que tout le monde comprenne, en plus de savoir faire.
4. Vous convaincre que la programmation, c'est aussi pour vous !

Maximum de validation ⇒

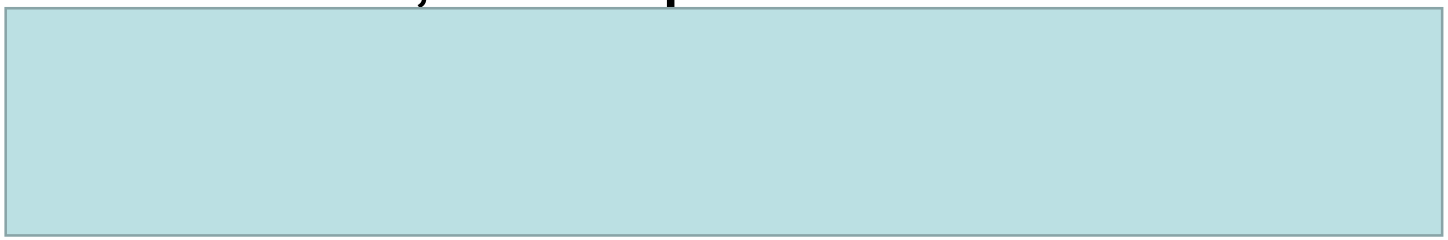
- Une plus grande rigueur de votre part
  (*essayez d'oublier un ; ou de remplacer une majuscule par une minuscule et voyez ce que dit le compilateur …*).

- Mais pas seulement dans le code java, dans les énoncés, les mails, …

- L'exigence de travail personnel est là, mais **l'objectif est atteignable**.

Un maximum de validations

- Ambitieux, mais possible !



- Ne nécessite pas de pré-requis, mais nécessite une bonne entente entre nous.
- Quelle que soit l'impression de **rigueur** que je pourrai vous faire dans les cours ou par mail, soyez assurés de ma **bienveillance** et sachez que je reste à votre **écoute** pour des **questions**, ou en cas de **difficultés**.

Évaluation

- 2 partiels + 1 final C + 1 projet + 1 final Java
- Q.C.M. (2 partiels)
- Écrit (partiel2 et final)
- Questions de cours ouvertes (partiel1, final)
- Projet :
  - Contrôle continu (intermédiaire + C.R.A.)
  - Programme rendu (+ rapport + page web)
  - **Oral (explications)**
- Au cas où : ce n'est pas moi qui corrige …

Cursus informatique ESIEE (e1)

- **E1 :** IPA, **IPO**, IPM + 3 ou 4 ateliers
- **E2 :** Algorithmique, atelier Alg.Linéaire&3D, Ligne de commande + Langage C, (Introduction aux Technologies du Web), (Bases de données), (Python)
- **E3 :** Systèmes d'Exploitation (prog.C/Unix), Intelligence Artificielle, Cyber-Sécurité, Data Sciences, Réseaux, (Prog.C++), Projet de fin de 1ère année Cycle Ingénieur

L'idée générale …

- Un programme qui « marche » ne suffit pas : il doit être « bien programmé ».
  (comprendre par « petites touches »)

- Vous devez **accepter/assumer** les «critiques»
- Le but n'est jamais de vous rabaisser, mais de **vous faire progresser**.
- Essayez toujours de voir l'intérêt de ce qu'on vous dit d'améliorer, et non pas une intention de « vous embêter ».

Méthode d'apprentissage

- Différente de tout ce que vous avez connu avant d'arriver à l'ESIEE *(probablement)*
- Différente de ce que vous connaissez dans la plupart des autres unités ESIEE

- Demande plus d'**implication personnelle**, mais les apprentissages seront **durables**.
- Vous avez toujours le droit de **TOUT** demander
- Vous n'avez **pas** le droit de ne **pas** demander lorsque vous ne comprenez **pas** quelque chose

Ce qu'on attend de vous

- Peut être différent de ce qu'on vous a demandé dans l'unité IPA.
- Écrire des classes complètes, et non plus seulement des petits bouts de code.
- Comprendre tous les concepts et pouvoir être interrogé dessus, y compris sur papier.
- Vous avez toujours le droit de **TOUT** demander, y compris les notions d' IPA !
  (mais ne vous offusquez pas si, avant ou après la réponse, on vous dit que vous auriez dû le savoir …)

Séquencement « inversé » :

HABITUEL :

- Cours (à comprendre) → TD (d'application du cours) → TP (d'application du TD) → projet (d'application)

NOUVEAU :

- TP (de découverte) → TD (de compréhension) → Cours (de restructuration)
- Projet en // mêlant découverte, contraintes, recherche de solution, créativité, …

==> questions, questions, questions
(celui qui n'en pose pas est « suspect »)

Les questions 1/3

- Dès que vous ne comprenez pas qqch dans une lecture ou dans un cours, vous devez poser une question. N'ayez pas peur, si elle « tombe mal », on vous le dira simplement.
- Si la réponse à votre question ne vous suffit pas, **posez une 2ème question**, voire une 3ème.
- Si vous n'osez pas la poser devant tout le monde, venez dans mon bureau, ou envoyez-moi un mail.
- Vous n'avez pas le droit de ne pas demander lorsque vous ne comprenez pas quelque chose

Les questions 2/3

- Ne vous attendez pas à avoir « La solution ».
- Vous obtiendrez souvent une autre question vous permettant **par vous-même** de répondre à votre question initiale.
- Si la réponse à votre question ne vous suffit pas, **posez une 2ème question**, voire une 3ème.
- **Ne demandez jamais la solution** à un camarade, ne regardez jamais son code : demandez-lui de l'aide pour la trouver.
- **Ne donnez jamais la solution** si vous voulez aider un camarade : aidez-le à la trouver !

Les questions 3/3

- Ne vous dites pas « ça n'en finira jamais », il est rare que 2 ou 3 questions ne suffisent pas
- Vous éprouverez une **grande satisfaction** à trouver vous-même la solution, et ces apprentissages sont **durables** & transposables
- Vers la fin de l'unité, vous avez toujours le droit de poser une question sur une notion du début
- Ne vous choquez pas du ton éventuel de la réponse qui laisserait sous-entendre que vous devriez le savoir ; il faut l'**assumer**, **mais vous aurez toujours votre réponse**.

Pas tout, tout seul !

- Contrairement à ce qui est généralement attendu dans les autres unités, ici vous n'êtes pas censé savoir faire tout seul tout ce qu'on vous demande, mais il faut essayer !

- Vous êtes volontairement confrontés à des problèmes que vous ne savez pas résoudre, et grâce à votre réflexion, **votre acharnement**, et à la méthode des questions précédemment exposée, vous trouverez la solution et la retiendrez !

Comment obtenir de l'aide ?

- **Ne pas attendre** la séance suivante !
- La page web liste les intervenants (permanents ou vacataires) de l'unité, et renvoie vers le forum de chaque séance.
- Si vous n'arrivez pas à expliquer votre problème sur le forum, ou par mail, ou de vive voix, un PC sous Linux est disponible dans mon bureau pour vous permettre de vous connecter à votre compte esiee.
- Ressources : **d'abord celles fournies** (résumés, javadoc, liens, polycopié, …)

Le sage a dit :

1. « Si tu donnes un poisson à un homme,
   il mangera **un jour** ; si tu lui apprends
   à pêcher, il mangera **toujours**. »
   [proverbe africain]

   (indices plutôt que solution, et
   où trouver l'info plutôt que l'info)

2. « Celui qui pose une question simple
   prend le risque d'être bête une fois ;
   celui qui ne la pose pas,
   le restera toute sa vie. » [proverbe chinois]
   (ne pas garder d'incompréhensions)

Écrire le « bon » programme

- C'est **quasiment impossible** du 1er coup !
- Il faut accepter de se tromper, corriger, améliorer, corriger, …
- Si vous n'écrivez rien, **vous n'apprenez rien** !
- **Si vous écrivez quelque chose de faux**,
  soit vous vous en apercevez en vous relisant,
  soit l'enseignant vous le signale,
  soit le compilateur vous le signale.
- Si vous ne trouvez pas comment corriger, on vous y aide, et **vous avez appris** à résoudre 1 type de problème qui se reproduira sans doute.

La « page blanche »

- Ne pas attendre de « voir » dans sa tête toutes les instructions nécessaires pour résoudre 100% du problème posé, avant de commencer à écrire.

- **Écrire déjà ce que l'on sait**, puis le cas le plus simple, puis ajouter les cas particuliers (éventuellement avant) ==> écriture non linéaire/séquentielle

Je n'ai jamais programmé avant !

- Ce ne sera jamais une excuse dans cette unité (justement prévue pour ce profil).
- Donc, ne pas regarder ceux qui avancent plus vite et avec facilité : ils ont probablement déjà programmé, ou bien ont une tournure d'esprit qui leur demande moins d'effort en prog^{on}.
- Chacun(e) d'entre-vous peut y arriver avec du travail et en suivant la méthodologie proposée, mais le déclic peut être plus ou moins long à se manifester (parfois en P2 ☺) … (< TP 5 !)

Une contradiction

- L'objectif est bien évidemment d'apprendre à programmer et non pas d'apprendre par cœur des morceaux de programmes tout faits.
- Mais il est nécessaire d'apprendre des **bases par cœur** pour pouvoir raisonner, avancer, et comprendre les explications (vocabulaire, syntaxe, définitions, …). **Surtout au début** (dès la 1ère semaine) **!**
- Bonne nouvelle : il y a beaucoup moins de choses à apprendre par cœur « au total » que dans beaucoup d'autres matières !

Lecture attentive

- Quel que soit le type de séance,
  il est essentiel de **lire très attentivement**
  tous les mots de toutes les phrases,
  et même toutes les lettres !
  (la fonction ≠ les fonctions)
- **Ce n'est pas naturel** pour vous ==>
  penser à faire l'effort à chaque fois.
- Énoncés de TP ou TD, résumés de cours,
  consignes pour le projet, échanges sur le forum
  tout est sur la page web

Un par poste

- Chacun fait 100% des étapes à son rythme
  (et ne se contente pas d'approuver
  ce que fait son binôme).
  Avancer au même rythme que ses 2 voisins est « suspect ».
- Sauter une étape ou
  voir la solution sur l'écran du voisin
  ==> **perdre une occasion d'apprendre**
- En profiter pour apprendre à se servir de Linux,
  surtout en ligne de commandes ==>
  Outils et Moyens Informatiques *(début d'année)*

3 choses avant de partir :

1. Les séquences 1 & 2 sont TRÈS intensives !
   Toutes les séances planifiées sont nécessaires (même non encadrées), et du travail personnel non planifié aussi.
   Le premier contrôle arrive vite !
2. Dans une prochaine séance, vous devrez aussi remplir la Charte de l'unité pour vous engager sur des éléments concrets pour réussir dans cette unité. **C'est obligatoire !**
3. Au programme du prochain cours (0.2) :

Prochaine présentation :

- Détails des différentes sortes de TP
- Détails des séances TD, cours, Résa
- **Objectifs et organisation du projet**
- Travail personnel, conseils
- Introduction à Java
- Introduction à BlueJ
- Bien commencer l'unité
- Avant ça : **Résa 0.1** → mail