IPO

Initiation à la
Programmation
Objet

IGI-1104

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 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 (65%>13) **(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 !

Un maximum de validation

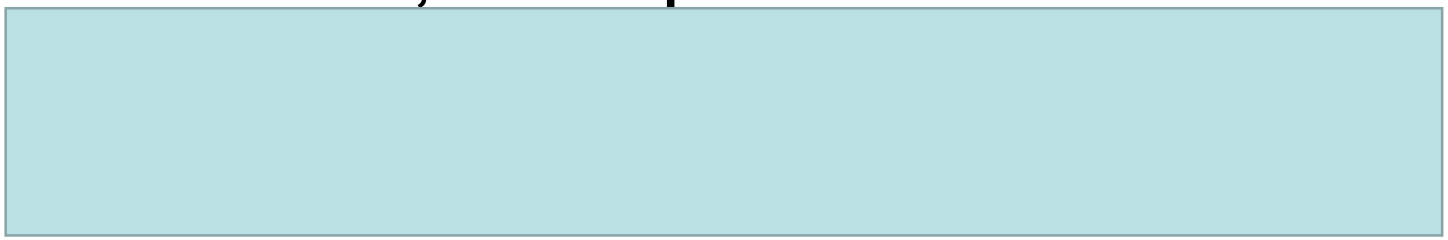
- Mon seul objectif est la réussite du plus grand nombre.

- Je suis persuadé que cela passe par de la rigueur de votre part *(essayez d'oublier un ; ou de remplacer une majuscule par une minuscule et voyez ce que dit le compilateur …)*.

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

Un maximum de validation

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

A quoi sert la programmation ?

- Nouveau slide nécessaire !?
- Domaine ESIEE = S.T.I.C.
  Sciences et Technologies de l'Information et de la Communication
- Nécessaire dans toutes les filières :
  Informatique, Systèmes Embarqués, Data-Science & I.A., Cyber-Sécurité, Bio/e-santé, Systèmes Électroniques, Génie Industriel, Énergie/environnement

Cursus informatique ESIEE (e1)

- **E1 :** [Sciences de l'Ingénieur], IPA, [Ateliers de fin d'année], IPO, IPM
- **E2 :** Langage C, (Intro. aux Technos du Web), Algorithmique, (Bases de données), (Python)
- **E3 :** Systèmes d'Exploitation, Graphes et algorithmes, (2 électives), IA, Cyber-Sécurité, Réseaux, … [Projet de fin de « Tronc Commun »]

Cursus informatique ESIEE E3ST

- **S1P1 :**
  Micro-processeurs, A3P (Java et début projet),
  E3T : Algèbre Linéaire (TP en java)

- **S1P2 :** suite A3P (C et fin du projet),
  Algorithmique et structures de données (C),
  [Atelier algorithmique et résolution de pb]

- **S2 :** Systèmes d'Exploitation (Linux, C),
  Graphes et algorithmes, Réseaux ou BDR,
  [atelier], [projet de fin de « Tronc Commun »]

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 !

- 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 qu'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 ☺) …

Une contradiction

- L'objectif est bien évidemment d'apprendre à programmer et non 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 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
  (parfois, sous les énoncés)

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 Résa sont nécessaires
   et du travail personnel non planifié aussi.
   Le premier contrôle arrive vite !

2. Dans une prochaine séance Résa, 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, pers
- **Objectifs et organisation du projet**
- Travail personnel, conseils
- Introduction à Java
- Introduction à BlueJ
- Usage de Moodle / la page E3ST
- Bien commencer l'unité

Précédente présentation

- Pédagogie ≠ => apprentissage ≠
- Inversion TP -> TD -> cours
- Modalités d'évaluation
- Filières, cursus informatique E1/E3ST
- Implication personnelle
- **Poser des questions**
- Trouver soi-même grâce à une « question »
- Ne pas attendre ni regarder la solution

Rappel :

Nécessité d'une attitude pro-active
dans cette unité.
(signaler dès que qqch ne va pas,
dès qu'il manque une information,
dès que vous avez une question, …)

Se contenter d'assister «passivement»
à chaque séance de TP, TD, cours, et Résa
est une condition nécessaire
mais pas suffisante !

Les séquences

- Elles sont toutes numérotées et détaillées sur la page de l'unité.
- Typiquement : TP*, TD, Résa, Cours
  (* *marqué TDm sur l'emploi du temps*)
- **Chaque séance est « cliquable »** et mène à un article de forum auquel on peut répondre pour poser des questions.
- Lire les échanges fait partie de l'apprentissage.
  Ce n'est pas du spam !

Les TP (TDm)

- TP « *tout court* » (9, énoncé) :
  découverte de nouvelles notions java par la pratique ; chercher à comprendre,
  pas à aller le plus vite possible
  *(pas de rapport à rendre, **un par poste**)*
  mais ne pas perdre de temps :
  profiter au maximum des intervenants !
  Il peut être demandé une lecture préalable.

- TP « de projet » (3, sans énoncé) :
  point d'avancement du projet
  avec une aide technique pour avancer.

Les TD

- TD (3) :

  - écrire un programme différent du TP, mais utilisant les mêmes notions => compréhension

- - écrire du java sur papier => mémorisation, intégration, et **préparation aux contrôles**

- - appeler l'intervenant dès qu'on ne comprend pas qqch ou lorsqu'on a terminé un exercice

Les cours

- Cours « spéciaux » (2 x 1h) :
  - introduction à la pédagogie et à l'unité
  (lecture : concepts « objet »)

- Cours « normaux » (8 x 1h) :
  cours courts !
  - restructuration de ce qui a été vu en TP/TD, voire en projet ==>
  Vous comprenez ce qui se dit ! Écouter 100% ?
  - basé sur un résumé de cours et des Q/R
  (**indispensable** à la véritable compréhension)

Les séances Résa (ex-Pers)

- Résa « dédiées » (3 x 1 ou 2h) :
  salles réservées pour accomplir une tâche,
  comme rendre une version du projet
  ou remplir un formulaire

- Résa « simples » (15 x 1 ou 2h) :
  salles réservées pour avancer le projet
  ou pour terminer un TP

- Tout à fait déraisonnable et non rentable
  de ne pas utiliser ces créneaux !

Travail personnel

- Travail non planifié dans ADE :
- Terminer les TP
- Terminer les TD (et les passer sur machine)
- Avancer le projet
- Relire les « apports de connaissance »
- Formuler et **poser des questions**

- ==> environnement de développement
  sur son ordinateur personnel
  (ou rester tard/week-end à l'ESIEE)

Le projet = un jeu d'aventure

- Le but n'est **pas** de réaliser un jeu vidéo !
- Le but est d'apprendre plus de java, et d'apprendre à utiliser une bonne conception objet, en réalisant un jeu d'aventure dont vous choisissez le scénario *(impossible à apprendre autrement, nécessité de refactoring)*.
- Un but secondaire est d'être fier d'avoir réalisé un jeu tout seul, si possible d'être sélectionné pour le Jour des Projets, et pourquoi pas, de gagner un prix ?

En savoir plus sur le projet 1/2

- Jeu d'aventure, tour par tour
- Projet décrit dans un livre (anglais/français)
- Liste officielle des exercices (oblig./optionnels)
- Créativité, contraintes, créativité
- Versions successives :
  - programmation améliorée
  - fonctionnalités ajoutées
- Explications données dans le livre, et exemples de programmation donnés au fur et à mesure, mais résolution de problèmes !
- **Commencez à réfléchir** à votre histoire …

En savoir plus sur le projet 2/2

- Pris par la main au début
  (ne pas traîner)
- De plus en plus en autonomie
  (ne pas se décourager)
- Découverte de notions nouvelles
- Comprendre le pourquoi du re-factoring
- **Ne pas attendre le prochain TP**
  pour poser des questions sur un exercice
  (utiliser le forum !)
- **Attention au plagiat !**
  (pour le moindre morceau de code)

Le langage Java (1/2)

- Plus facile à apprendre que le C.
  Erreurs moins graves pour un débutant.
- Après le C en E2, vous pourrez apprendre par vous-mêmes le C++ ou le C#.
- Famille de langages la plus utilisée au monde (C : 1972. C++ : 1983. Java : 1995, 21 versions)
- JRE = Java Runtime Environment (exécution)
- JDK = Java Development Kit (compilation)

- **Inutile :** installer le JDK **11** sur son ordinateur personnel (gratuit, open source)

Le langage Java (2/2)

- On ne peut pas apprendre « tout » Java.
- Ni le langage, encore moins l'A.P.I.
  (l'essentiel, javadoc, « pied à l'étrier »)

- Besoin de tout : TP + TD + cours + projet

- Conception Orientée Objet (philosophie)
  petit à petit, surtout par le projet

- Parti pris, vocabulaire, exigences pédagogiques

La Java Virtual Machine

- Processeur virtuel = programme interpréteur d'instructions « assembleur »
- Fourni sur toutes les plateformes (PC, smartphone, box, carte à puce, …)
- Programme compilé tourne tel quel entre Mac / PC / Windows / Linux / …

- Erreurs à la compilation ou à l'exécution
- Dire un maximum de choses au compilateur ! (ne pas économiser des caractères)

Environnement de développement

- BlueJ, prononcez « bloudjè » (IDE)
- Installer v5.0.3 sur son ordinateur personnel (gratuit, open source, inclut JDK 11)
- Plus adapté qu'Eclipse ou NetBeans (pédagogique)
- Peu de menus (tous les regarder ?)
- Schéma des classes / interaction
- Object bench / interaction / inspect
- Debugger
- Tests automatiques
- Code pad

iCampus (Moodle)

- Tout est sur iCampus, ma page perso (E3ST)
  semaine après semaine,
  séance après séance.

- Vous **devez** cliquer sur chaque séance,
  y compris Résa/Pers, pour lire les consignes.

- Toute unité est sur Blackboard (→ Moodle)
- Donc s'inscrire à l'unité sur Moodle
- (puis aussi à l'unité du projet Zuul)

Les bons conseils

- Il n'est pas attendu de vous que vous sachiez résoudre tous les problèmes par vous-mêmes.
- Ne pas accepter de ne pas comprendre qqch (retrouvez cette révolte qui est en vous !)
- **Prévenir** quand vous ne pouvez pas assister à une séance
  (en cas d'imprévu, s'excuser après)
- Et surtout : **rattraper au plus vite** !
  (1h encadrée ==> 2h en autonomie)

L’approche pédagogique

- Elle est évidemment non négociable.
- Toute l’équipe pédagogique est persuadée qu’elle est plus efficace que la « classique ».

- Nous comprenons qu’elle puisse ne pas convenir à une minorité d’étudiants.
- Nous essaierons de les convaincre dans un premier temps, mais ensuite, nous les aiderons à compenser les inconvénients pour eux de la méthode.
- **Votre persévérance est indispensable !**

Les contraintes absolues

1. Lire ses **mails** TOUS LES JOURS ! **[A3P:]**
2. **Toutes les séances sont utiles** : difficile de réussir en manquant un TP, TD, Cours, Résa.
3. Lire tous les mots de toutes les phrases des énoncés, des documents, ET DES MAILS.
   Unité Moodle : IGI-3007=A3P 2022
4. Bien gérer son temps :
   - travailler un peu **(quasiment) tous les jours**
   - ne pas rester bloqué plus d'une demi-heure sans poser de question (10mn en TP)

   (il est anormal d'y passer 20h/semaine !)

Bien commencer

1. L'unité démarre beaucoup plus intensément qu'il n'y paraît : beaucoup de notions nouvelles (même simples) et de **vocabulaire** (indispensable pour comprendre la suite) ; à apprendre/travailler **dès la 1ère semaine**.
2. *Chaque séance utilise ce qui a été vu précédemment. N'accumulez pas de retard !*
3. *Vous pouvez toujours essayer de rattraper un retard de cours peu avant un partiel, mais vous n'aurez pas acquis le savoir-faire nécessaire en programmation …*

Votre réussite est entre vos mains

Votre « pouvoir personnel » est immense,
2 attitudes possibles :

1. Utiliser toute son énergie et son temps à être capable de démontrer pourquoi on a échoué (garder les problèmes en réserve ➔ + de repos)

2. Utiliser toute son énergie et son temps pour essayer de réussir (résoudre les problèmes **dès qu'ils se présentent** ➔ + de travail)

3. Valider en ligne la charte de l'unité ==> **Résa1**

Dès que quelque chose ne va pas :

Dites-le moi au plus vite !

- Vous ne comprenez pas ce qui est demandé
- Vous ne comprenez pas comment faire
- Vous n’arrivez pas à résoudre un problème
- Un lien ne fonctionne pas
- Un document/renseignement manque
- Un intervenant pose problème (son nom !)
- …

Prochaine séance :
Résa 0.1

- Se connecter, puis lancer un navigateur
- **Page E3ST** dans la barre d'adresse
- Se connecter.
- E1 / S.I. / Informatique / IGI-1104=**IPO 2021**
- **M'inscrire**
- Cliquer sur Résa 0.1 => instructions
- Charte de l'unité, puis parcourir ressources.
- (une 2ème et dernière fois **M'inscrire**)

Merci de votre attention

Des questions ?