Compléments de la POO
Cours 1

2021-2022 Université de Paris – Campus Grands Moulins
Licence 3 d'Informatique
Eugène Asarin

Merci à Aldric Degorre pour ses supports de cours!!!

Aspects pratiques 1

- Prof du cours Eugène Asarin
- Profs de TP : Isabelle Fagnot, Yan Jurski, Ni Ayna Andriambolamalala, Wael Boutglay
- Les 6 cours : 17/9, 1/10, 15/10, 29/10, 12/11, 26/11
- Les TP (chaque semaine) commencent le 20/10
- Conseil: venir en TP avec le JDK (version 15+) + IDE favori (Eclipse, IntelliJ, NetBeans, vi ☺) installés sur vos portables.
- Nous contacter: asarin@irif.fr, fagnot@univ-mlv.fr, jurski@irif.fr, ny-aina.andriambolamalala@irif.fr, btwael@gmail.com

Aspects pratiques 2

- MCC
  - S1: 40% projet + 30 % examen (papier) + 30% CC (QCM/devoirs etc)
  - S2: 100% examen (papier)
- Covid-19
  - Protégez-vous, protégez-nous: gestes barrières (obligatoires), vaccins (recommandés)
  - Malade, contact, quarantaine, symptômes, doutes - consultez, testez-vous, ne venez pas en cours, prévenez la scolarité et nous
  - On adaptera les MCC/échéances aux cas particuliers
  - Et on fera notre mieux pour s'adapter aux changements de situation

Aspects pratiques 3 – deux comptes nécessaires

- Sur U-Paris
  - Moodle indispensable (supports, annonces, rendus), inscrivez-vous si ce n'est pas fait
- Pour les machines de l'UFR d'info
  - L2 chez nous ou redoublants => vous l'avez déjà
  - Nouveaux => vous avez reçu un SMS.

    Confusions possible 0-O (zéro-oh) et 1-I-l (un – i majuscule – el)
  - Pas de compte, mdp oublié, autre problèmes – demandez au prof de TP

Sources d'info (Moodle + bibliothèque+ Internet)

- Au lieu d'un poly de cours:
  les 565 slides très détaillés d'Aldric Degorre
- Notes de cours et de TP, exemples de code, annales – sur Moodle
- Livres (excellents mais épais)
  - *Effective Java, 3rd edition* (Joshua Bloch)
  - *Java Concurrency in Practice* (Brian Goetz)
  - *Design Patterns : Elements of Reusable Object-Oriented Software* (Erich Gamma, Richard Helm, Ralph Johnson et John Vlissides a.k.a. "the Gang of Four")
- Site Java d'Oracle (y compris API doc) et des millions d'autres

Ce cours dans le curriculum

- *Secret pédagogique: théorie en séquence, programmation en couches*

- Prérequis: programmation de base, savoir programmer en Java et comprendre les objets (POOIG+projet de L2)
- Après ce cours: C++, Android, concurrence en M1, POCA en M2
- Et aussi lié à : Génie logiciel, prog. fonctionnelle

Principe

- On fait du Java (mais pourquoi?)
- On revient aux thèmes classiques (classes-objets-constructeurs, encapsulation-héritage-polymorphisme, généricité, exceptions):
  - révisions (pour approfondir voir le poly)
  - problèmes, limitations et dangers
  - nouvelles techniques, souvent design patterns, un peu de style
- On approfondit beaucoup Lambdas/Streams (si vous voulez Map-Reduce)
- On entame (bien) la programmation concurrente multithread
- …et voilà Noël et les vacances

Objectifs

- Académique: mieux maitriser les principes de POO
  - Ça se teste en examen et en QCM
- Professionnalisant : évoluer vers le code professionnel (applications et bibliothèques) – clair, réutilisable, évolutif, sûr/fiable/sécure
- Et aussi: apprendre des nouvelles techniques et approches
  - Ça se teste en TP, mais surtout en projet
  - Le projet devrait
    - bien fonctionner (10 points)
    - être bien programmé! (4 points)
    - utiliser les nouvelles techniques! (6 points)

Allez, on commence

Rappel: Histoire des langages impératifs +

- Au débuts: code machine + schéma bloc à côté
- Assembleur: même chose mais avec des noms/adresses symboliques
- Langages haut niveau: Fortran, Algol
- Programmation structurée => Pascal, Ada, C
- POO: Simula, Smalltalk, C++, Java

// à chaque étape des ordres de grandeur gagnés pour la taille des projets etc.

- Et puis tous les langages impératifs sont devenu structurés, OO, un peu fonctionnels… Python, Scala, Kotlin, C#
- En même temps autres paradigmes existent: ML (fonctionnel), SQL (relationnel), Prolog (logique),…

Rappel – caractéristiques du Java (volé chez Aldric D) vous devez comprendre chaque mot ci-dessous

Java (Java SE) est en réalité une plateforme de programmation caractérisée par :

- le langage de programmation Java
  - orienté objet à classes,
  - à la syntaxe inspirée de C
  - au typage statique,
  - à gestion automatique de la mémoire, via son garbage collector).
- sa machine virtuelle permettant aux programmes Java d'être multi-plateforme (le code source se compile en code octet pour JVM, laquelle est implémentée pour nombreux types de machines physiques).
- les bibliothèques officielles du JDK (fournissant l'API 3 Java), très nombreuses et bien documentées (+ nombreuses bibliothèques de tierces parties)

Conseils de style (documents Sun + bon sens)

- Pensez à la lisibilité, respectez les règles de votre entreprise
- Commentaires, javadoc pour l'API, indentation
- Structurez en modules(?)-packages-classes-méthodes de bonne taille
- Noms (majuscules/minuscules etc.)
  - Types : Person, TriplyLinkedList
  - Variables : x, ind, student, firstName (court seulement si privé et très local)
  - Constante GRID_SIZE
  - Packages: fr.uparis.specialutils

Style: + sur les noms (English!)

- Variables : noms (student) ou participes passés (finished)
- Types: noms singulier (Student), nom pluriels pour classes-outils (Collections), mais aussi intreface Printable
- Méthodes : print, sort ; getX, setX; contains, isOK; List.of(2,3,4)
  - x.sort() vs y=x.sorted() (effet de bord vs transformation)
- Privilégiez l'anglais, évitez les accents

Rappel – classes et objets

- Vision OO: un graphe dynamique d'objets qui naissent, meurent, changent d'état, se parlent
- Un objet Java: attributs, méthodes, on y accède par références.
- Une classe Java: définit un type d'objets (et aussi des fonctions statiques, constantes, membres statiques etc)
- On instancie des objets (alloue mémoire, crée, récupère référence) grâce aux constructeurs
- Mais je ne veux pas vous répéter les cours de L1-L2!!! Voyons un exemple

Et maintenant

- On appliquera notre méthode à la première étape de la vie d'objet: l'instanciation
- Notre méthode:
  - Se rappeler l'approche de base de Java, vue en L2
  - Voire ses limitations/risques/dangers
  - Étudier des techniques avancées, voir leurs avantages et inconvénients
  - Souvent il s'agit de *design patterns*

Rappel – constructeurs

- On peut écrire des constructeurs dans la classe (sinon il y a un par défaut).
- On instancie (ça veut dire?) un objet en appelant un constructeur

Personne directeur=new Personne("Delporte"," Carole")

Limitation des constructeurs

- Même nom pour tous
- Un seul constructeur de même signature
- Pas de contrôle d’instance (toujours une nouvelle instance est créée)
- Toujours instance directe de la même classe retournée
- Pénible si beaucoup de paramètres facultatifs….

Les fabriques statiques sont souvent meilleures

- On peut en avoir plusieurs pour la même signature, noms parlants
- On peut faire une instance unique, éviter des doublons etc
- On peut renvoyer un objet de sous-classe, même cela
- Noms typiques:
  - from, of, valueOf, instance ou getInstance, create ou newInstance, getType
- Exemples connus
  - List.of(12,54,31)
  - Integer.valueOf("011111",2)
  - BufferedReader br = Files.newBufferedReader(path);
  - Calendar cal= Calendar.getInstance()
- Exemples vus en cours (à voir sur Moodle):
  - Personne.java (plusieurs fabriques de même signature)
  - Johnny.java (singleton – une seule instance possible, avec une fabrique…)

Cas particulier : patron Monteur (Builder)

en fait c'est une fabrique dynamique…

Exemple (Effective Java – Nutritional Facts)

Décrit la quantité de produit par portion, nombre de portions, et peut-être calories, lipides, sodium, glucide… Certains paramètres peuvent manquer. *(On voudrait aussi contrôler que calories>=9*lipides+4*glucides)*

3 solutions (Voir fichiers d'exemples sur Moodle):

- Constructeur téléscopant
- Constructeur JavaBeans
- Builder – solution recommandée (à maitriser)!!!