Programmation et Algorithmique II
— Compétences à acquérir —

Pour réussir le cours de Programmation et Algorithmique II, vous devez être capable de

- mettre en oeuvre les compétences acquises au cours de Programmation et Algorithmique I
- écrire correctement, simplement et proprement un algorithme ou un programme; appliquer les règles de bonne pratique telles que choisir des noms de variables et méthodes adéquats; écrire des **commentaires** pertinents
- sur base de son énoncé en français, comprendre un **problème** et concevoir un algorithme ou un programme qui le résolvent
- exécuter mentalement ou sur papier un algorithme sur un exemple; vérifier la terminaison ou la correction d'un algorithme
- évaluer la **complexité** d'un algorithme; comprendre la notion de complexité amortie
- écrire correctement un **algorithme récursif**; utiliser une stratégie **diviser-pour régner** dans un algorithme; mettre en oeuvre la **mémoïsation**
- connaître les principes des **algorithmes de tri** basiques (bulle, insertion, sélection et fusion)
- avoir une maîtrise suffisante du **langage Java**; concevoir ou utiliser une classe ou un ensemble de classes Java; définir les constantes, variables d'instance ou de classe, méthodes d'instance ou de classe, classes internes, constructeur; avoir une connaissance des types de données basiques primitifs (`byte`, `short`, `int`, `long`, `float`, `double`, `char` et `boolean`) et objets (p.ex. `String`); utiliser les structures de contrôle (`if`/`else`, `for`, `while`, `do`...`while`, `switch`); concevoir ou utiliser une *interface*; comprendre la vérification de types statique (à la compilation) et dynamique (à l'exécution)
- utiliser des classes définies dans la **bibliothèque Java** (en ayant la documentation à disposition)
- être capable de déclarer, créer et manipuler un **tableau**; être conscient des différences entre la manipulation des tableaux en Java et en Python (en particulier, le *slicing* n'est pas supporté en Java)
- concevoir et manipuler des **structures de données simples** telles qu'un tableau, une `ArrayList`, une liste simplement ou doublement chaînée, ordonnée ou non, ainsi qu'une table de hachage; sélectionner une structure de données adéquate pour un problème
- comparer des objets entre eux; concevoir ou utiliser une classe `Comparable`, concevoir un `Comparator`
- comprendre et mettre en oeuvre adéquatement les **concepts de programmation orienté-objet**, à savoir : l'encapsulation, l'héritage, les références dynamiques, le polymorphisme, la surcharge et le remplacement de méthodes
- générer / capturer des exceptions; utiliser les **exceptions** à bon escient
- lire des données (évt. structurées) à partir de la console ou d'un fichier; produire des données (évt. structurées) vers la console ou un fichier; manipuler les flux (*streams*); comprendre la différence entre les différentes classes flux de la bibliothèque Java
- utiliser, concevoir et mettre en oeuvre les ***design patterns*** *visiteur* et *itérateur*
- **compiler et exécuter** un programme
- utiliser des classes et méthodes **génériques** (i.e. munies de paramètres de types)