Langage de programmation orientée objet - Série 1

Objectifs

- Prendre en main l'environnement de développement intégré (IDE)
- Manipuler les notions élémentaires : définir une classe, créer des objets à partir de cette classe et appeler des méthodes sur ces objets

Exercice 1 : Prise en main de l'IDE et entrées-sorties

Objectifs spécifiques

- Installer IntelliJ
- Créer un projet
- Afficher des informations à la console
- Saisir des données introduites par l'utilisateur

Étape 1 : Créer un compte JetBrains

Pour bénéficier de l'utilisation gratuite d'IntelliJ (version Ultimate) pendant un an, créez un compte étudiant JetBrains.

Suggestion

Pour la création d'un compte JetBrains, suivez le guide que vous trouverez sur Moodle.

Étape 2 : Démarrer IntelliJ

Si vous ne trouvez pas IntelliJ dans la liste des programmes installés, cherchez dans `C:\Program Files\JetBrains\IntelliJ.`

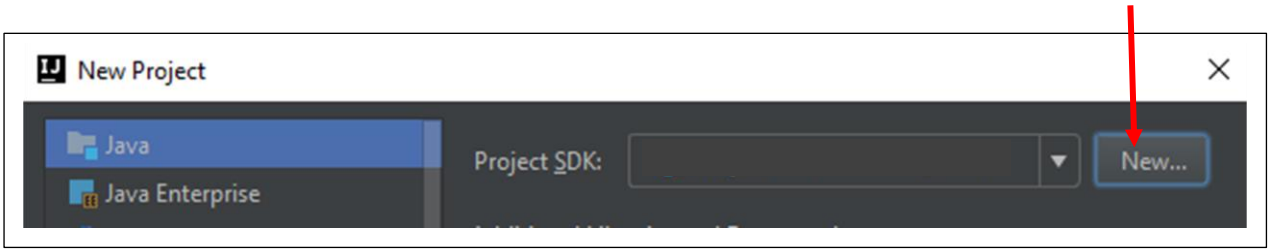
Au lancement d'IntelliJ, validez votre licence avec votre compte JetBrains.

Étape 3 : Créer un projet

Créez un nouveau projet appelé `PremierProjet`.

File ⇨ New ⇨ Project ⇨ Choisir un projet de type Java

Si IntelliJ ne trouve pas de SDK, il vous propose soit de le downloader, soit de lui en associer un déjà installé sur l'ordinateur. Sur les machines de l'IESN, il est déjà installé et se trouve dans C:\Program Files\Java. Pour l'associer à IntelliJ, cliquez sur New et retrouvez-le (ex: C:\Program Files\Java\jdk…).



Donnez un nom à votre projet et une localisation (Project location).

> **Truc**
>
> Si vous souhaitez récupérer votre projet ultérieurement sur une autre machine de l'IESN ou chez vous, vous devrez sauver votre programme à la fin du labo idéalement sur One Drive.

Le projet contient un répertoire appelé `src` qui va contenir le code source du programme.

Étape 4 : Créer un package

En vue de pouvoir réutiliser les classes que l'on écrit, il faut toujours créer un package pour les y placer et donc éviter de travailler dans le package par défaut (default package).

Attention, le nom d'un package commence par une minuscule.

Créez un package appelé `premierPackage`.

Clic droit sur `src` ⇨ New ⇨ Package

Étape 5 : Créer une classe avec la méthode main

Dans le package `premierPackage`, créez une classe appelée `Principale`.

Clic droit sur le package ⇨ New ⇨ Java Class

En tant que classe principale, elle doit contenir une méthode `public static void main(String[] args)` qui sera exécutée automatiquement au lancement du projet (cette méthode joue un rôle identique à celui de la fonction `main` dans un programme C). Dans cette méthode, vous pouvez ajouter du code à la manière d'un langage impératif.

> **Truc**
>
> Dans l'environnement *IntelliJ*, vous pouvez taper `psvm` puis ENTER et le code de cette méthode sera automatiquement généré. Vous utiliserez ainsi des *Live Templates* vous permettant d'insérer dans votre code des constructions fréquemment utilisées, et ce, à partir d'abréviations.

Étape 6 : Variables

Dans la méthode `main`, déclarez deux variables entières `x` et `y` ainsi qu'une variable de type chaîne de caractères (String) appelée `mot`.

> **Camélisation**
>
> En java, les noms de variables ou de méthodes commencent par une minuscule et respectent la notation appelée "camel case" (aussi stylisé en camelCase). Le "camel case" fait par défaut référence au "lower camel case", c'est-à-dire que la première lettre du nom de la variable ou de la méthode est en minuscule. Le "Pascal case" fait lui référence à la version commençant par une majuscule utilisée notamment en C# pour les noms de méthodes. Il convient de respecter la notation "Camel Case" en Java car c'est un des pragmas (bonnes pratiques) du langage.

Ajoutez les instructions permettant d'affecter respectivement à ces variables les valeurs : 7, 3 et "Java".

Étape 7 : Affichage à la console

Ajoutez une instruction permettant d'afficher à la console la phrase ci-dessous (pour afficher un message, utilisez la méthode `System.out.println(…)`).

> **Convention dans les exercices**
>
> Par convention, dans les exemples de sorties des énoncés d'exercices, les parties soulignées et entre parenthèses sont des parties qui devront être remplacées par les valeurs des variables d'instance correspondantes.

Le produit vaut (produit des deux entiers) et le mot est (contenu de la variable mot).

L'opérateur +

L'opérateur + appliqué à deux variables de type numérique est l'opérateur d'addition.

L'opérateur + appliqué à deux objets de type String est l'opérateur de concaténation ; il en va de même si l'opérateur + est appliqué entre un objet de type String et une variable numérique.

Exécutez ensuite le projet :

Première exécution : Clic droit sur la classe contenant la méthode main ⇨ Run

Exécutions suivantes : Flèche verte (réexécute la dernière méthode main)

Étape 8 : Saisie au clavier

Ajoutez la déclaration

```
Scanner clavier = new Scanner(System.in);
```

Puis examinez l'erreur rapportée par l'IDE et résolvez-la (autorisez-le à ajouter la ligne d'importation nécessaire pour pouvoir utiliser la classe `Scanner` : `java.util.Scanner` ou `java.util.*` ; vous pouvez aussi ajouter manuellement `import java.util.Scanner;` au début du fichier *Principal.java*).

Import de package

L'instruction d'import se place après la première ligne reprenant le nom du package et avant la déclaration de la classe.

```
package premierPackage;
import java.util.Scanner;
public class Principal {
```

Remplacez l'affectation `x = 7;` par les lignes ci-dessous puis exécutez à nouveau le projet.

```
System.out.print("Entrez un entier : ");
x = clavier.nextInt();
```

De même, remplacez l'affectation `mot = "Java";` par les lignes ci-dessous puis exécutez à nouveau le projet.

```
System.out.print("Entrez un texte : ");
mot = clavier.next();
```

Notons qu'il existe d'autres méthodes que l'on peut appeler sur l'objet `clavier` selon le type de valeurs que l'on veut obtenir (`nextFloat()`, `nextDouble()`, `nextBoolean()`…).

Désormais, vous savez comment effectuer des entrées et des sorties en Java. Il est temps de passer à l'orienté objet !

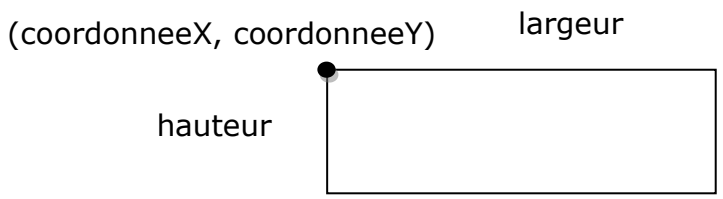
Exercice 2 : Classe Rectangle

Objectifs spécifiques
- Créer des objets à partir d'une classe
- Appeler des méthodes sur des objets

Étape 1 : Créer une classe

Toujours dans `PremierProjet` et dans le même package, créez une nouvelle classe (un fichier par classe Java) appelée `Rectangle` permettant de gérer des rectangles : *Clic droit sur premierPackage ⇨ New ⇨ Java Class*

Tout rectangle est décrit par les coordonnées X et Y de son point d'ancrage (le coin supérieur gauche) ainsi que par sa largeur et sa hauteur.



Représentation UML :

Rectangle
coordonneeX coordonneeY largeur hauteur
perimetre() surface() modifierLargeur(int) modifierHauteur(int) deplacerEn(int,int)

Suggestion

Pour le code de la classe `Rectangle`, vous pouvez vous inspirer du syllabus (cf. point 2.4).

Étape 2 : Déclarer les variables d'instance (ou attributs)

La classe comportera les variables d'instance suivantes de type entier (avec la protection par défaut ne nécessitant aucun mot clé : visibilité package) :

- coordonneeX
- coordonneeY
- largeur
- hauteur

Étape 3 : Constructeur

Créez un constructeur pour la classe `Rectangle` ; ce constructeur reçoit en arguments les coordonnées x et y, ainsi que la largeur et la hauteur du rectangle à créer.

Étape 4 : Création d'un objet de la classe

Pour tester votre code créé jusqu'ici, dans la classe `Principale`, remplacez le contenu de la méthode `main` par :

- la déclaration d'un objet appelé `premierRectangle` de type `Rectangle` ;
- l'instanciation de `premierRectangle` en utilisant le constructeur. Les valeurs à transmettre lors de cette instanciation sont les suivantes : 1 pour la coordonnée X, 2 pour la coordonnée y, 10 pour la largeur et 4 pour la hauteur.

Étape 5 : Accès aux variables d'instance d'un objet

Dans la méthode `main` :

- affichez (via `System.out.println`) les coordonnées x et y, la largeur et la hauteur de l'objet `premierRectangle` ;
- modifiez les variables d'instance de l'objet `premierRectangle` : 0 pour la coordonnée X, 3 pour la coordonnée y, 5 pour la largeur et 6 pour la hauteur ;
- affichez de nouveau les coordonnées x et y, la largeur et la hauteur de l'objet `premierRectangle`.

Étape 6 : Définition de méthodes

Ajoutez dans la classe `Rectangle` :

- une méthode appelée `modifierLargeur` qui reçoit en argument un entier (positif ou négatif) et qui ajoute cette valeur à la largeur du rectangle (*méthode sans type de retour*) ;
- une méthode appelée `modifierHauteur` qui reçoit en argument un entier (positif ou négatif) et qui ajoute cette valeur à la hauteur du rectangle (*méthode sans type de retour*) ;
- une méthode appelée `perimetre` *(sans arguments)* qui retourne le périmètre calculé du rectangle ;
- une méthode appelée `surface` *(sans arguments)* qui retourne la surface calculée du rectangle ;
- une méthode appelée `deplacerEn` qui reçoit en argument deux entiers : les coordonnées X et Y du *nouveau* point d'ancrage et qui remplace les coordonnées X et Y de l'ancien point d'ancrage par les valeurs reçues en entrée (*méthode sans type de retour*).

Étape 7 : Appel de méthode sur des objets

Dans la classe `Principale`, appelez ces différentes méthodes sur l'objet `premierRectangle` :

- affichez à la console le périmètre et la surface de l'objet `premierRectangle` ;
- élargissez et agrandissez `premierRectangle` en appelant les méthodes adéquates ;
- déplacez `premierRectangle` en appelant la méthode `deplacerEn` ;
- affichez ensuite de nouveau la largeur, la hauteur et les coordonnées x et y de `premierRectangle`.

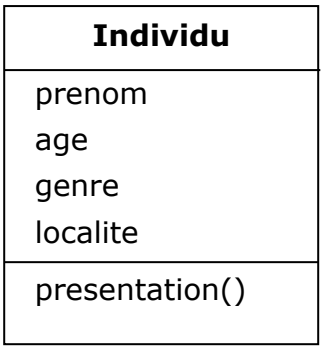
Créez d'autres objets de la classe `Rectangle` avec d'autres valeurs de point d'ancrage, de largeur et de hauteur et testez-les en appelant différentes méthodes de la classe `Rectangle` et en affichant ensuite leurs point d'ancrage, hauteur, largeur, surface et périmètre.

Exercice 3 : Classe Individu

Étape 1 : Classe et variables d'instance

Toujours dans `PremierProjet` et dans le même package, créez une nouvelle classe appelée `Individu`.

Représentation UML :



La classe comportera les variables d'instance suivantes déclarées avec la protection par défaut (de type package) :

- un prénom (`String`) ;
- un âge (`int`) ;
- un genre (`char`) : valeurs 'M', 'F' ou 'X' ;
- une localité (`String`).

Étape 2 : Constructeur

Créez un constructeur pour la classe `Individu` ; ce constructeur recevra en arguments le prénom, l'âge, le genre et la localité d'un individu.

Étape 3 : Instanciation et accès aux variables d'instance

Dans la méthode `main` de la classe `Principale`, ajoutez :

- la déclaration d'un objet appelé `moi` de type `Individu` ;
- l'instanciation de l'objet `moi` en utilisant le constructeur (utilisez les valeurs qui vous décrivent ou d'autres, au choix) ;
- l'affichage du prénom, du genre et de l'âge de l'objet `moi` ;
- une instruction modifiant le genre de `moi` en 'W' et l'âge de `moi` en -124 ;
- à nouveau la même instruction d'affichage que ci-dessus.

Étape 4 : Méthode

Ajoutez à la classe `Individu` la méthode `presentation` qui **va retourner la chaîne de caractères** respectant la structure ci-dessous.

Je m'appelle (prenom) et je suis âgé(e) de (age) an(s).

Je réside à (localite)

Pour rappel, les valeurs entre parenthèses et soulignées devront être remplacées par les valeurs des variables d'instance correspondantes de l'objet de type `Individu` sur lequel on appelera la méthode `presentation`.

Tant qu'à faire, arrangez-vous pour que le "s" de "an(s)" n'apparaisse que si nécessaire (opérateur conditionnel ?) et que le "e" de "âgé(e)" lui aussi ne soit là que si nécessaire.

Adaptez la méthode `main` de la classe `Principale` :

- **afficher à la console la chaîne de caractères** présentant l'objet `moi` qui vous correspond ;
- **afficher à la console la chaîne de caractères** présentant l'objet `moi` après modification du genre en 'W' et de l'âge en -124.

Bonne pratique : introduction à la séparation des couches vue et modèle

On privilégiera la réutilisation des classes dans des applications différentes. Or, les applications peuvent proposer des interfaces utilisateurs différents : affichage à la console, dans des pages HTML, dans des écrans en Android…

On évitera donc de prévoir dans les classes dites "modèles" (classes à partir desquelles on créera des objets) des méthodes qui fixent (hardcodent) les canaux de communication avec l'utilisateur, comme par exemple les entrées et sorties sur la console. On évitera donc les instructions de type `System.out.println` ou d'utiliser la classe `Scanner` dans les méthodes des classes modèles.

On préférera donc la méthode `presentation` qui retourne une chaîne de caractères plutôt que la méthode `presentation` qui contiendrait l'instruction `System.out.println`. Dans notre exercice, c'est la méthode `main` qui appellera la méthode `presentation` et qui se chargera d'afficher à la console le résultat de la méthode.

Exercice 4 : Billets pour spectacle

Objectifs spécifiques
- Organiser les classes en packages
- Déclarer des variables d'instance de différents types
- Appeler au sein d'une méthode une autre méthode de la même classe

Étape 1 : Créer un autre package

Créez un nouveau package intitulé `packageLoisir` dans le projet `PremierProjet`.

Étape 2 : Créer la classe BilletSpectacle

Créez dans `packageLoisir` la classe `BilletSpectacle`, avec les variables d'instance suivantes :

- un intitulé de spectacle (`String`)
- une date de spectacle (`String`)
- une catégorie (`char`)
- un cout de base (`double`)
- avec ou sans carte d'étudiant (`boolean`)

BilletSpectacle
intituleSpectacle dateSpectacle categorie coutDeBase avecCarteEtudiant
prixBillet() descriptionBillet()

Étape 3 : Constructeur

Créez un constructeur pour la classe `BilletSpectacle` ; ce constructeur devra permettre d'initialiser toutes les variables d'instance.

Étape 4 : Méthodes

Prévoyez dans la classe `BilletSpectacle` la méthode `prixBillet` qui va retourner le prix final du billet sur base des règles suivantes :

- En catégorie 'A', le prix du billet est égal au coût de base augmenté de 10% ; en catégorie 'B', le prix du billet est égal au coût de base diminué de 20% ; dans tous les autres cas, le prix du billet est égal au coût de base.
- Si l'acheteur du billet dispose d'une carte d'étudiant, celui-ci paiera la moitié du prix ainsi calculé.

Prévoyez également la méthode `descriptionBillet` qui retourne la chaîne de caractères décrivant le billet en respectant la structure ci-dessous. Les valeurs entre parenthèses et soulignées doivent être remplacées par les valeurs correspondantes de l'objet de type `BilletSpectacle`.

Si l'acheteur du billet dispose d'une carte d'étudiant :

Billet pour le spectacle intitulé (intituleSpectacle) du (dateSpectacle)
en catégorie (categorie) avec carte étudiant
pour un total de ... euros
↳ *Appel à la méthode* ***prixBillet***

Si l'acheteur du billet ne dispose pas de carte d'étudiant :

Billet pour le spectacle intitulé (intituleSpectacle) du (dateSpectacle)
en catégorie (categorie)
pour un total de ... euros
↳ *Appel à la méthode* ***prixBillet***

Étape 5 : Création d'objets et appels de méthode

Dans le package `packageLoisir`, créez la classe `Principale` contenant la méthode `main`.

Dans cette méthode `main` :

- créez et initialisez plusieurs objets de type `BilletSpectacle` ;
- affichez à la console pour chacun d'eux le résultat de l'appel à la méthode `prixBillet` ;
- affichez à la console pour chacun d'eux leur description (`descriptionBillet`).

Exercice 5 : Séjours à Disneyland

Objectif spécifique
- Appeler au sein d'une méthode d'autres méthodes de la même classe

Étape 1 : Créer la classe SejourDisney

Créez dans `packageLoisir` la classe `SejourDisney`, avec les variables d'instance suivantes :

- un nombre d'enfants (`int`)
- un nombre d'adultes (`int`)
- un nombre de jours (`int`)
- un nombre de places de parking par jour (`int`)
- avec ou sans personne à mobilité réduite (`boolean`)

SejourDisney
nbEnfants nbAdultes nbJours nbVehiculesParkin g avecPMR
estLongSejour() coutEntreeParc() nbNuiteesEnfantsGratuites() coutHotel() coutParking() coutTotal() aAccesPrioritaire() resumeSejour()

Étape 2 : Constructeur

Créez un constructeur pour la classe `SejourDisney`; ce constructeur doit permettre d'initialiser toutes les variables d'instance.

Étape 3 : Méthodes

On part du principe que le nombre de nuits du séjour sera égal au nombre de jours - 1.

Prévoyez dans la classe `SejourDisney` les méthodes suivantes :

- La méthode `estLongSejour` retourne un booléen : à partir de 3 nuits, un séjour est considéré comme étant un long séjour.
- La méthode `coutEntreeParc` calcule le coût total des entrées au parc d'attraction Disney comme suit : 19,9 euros par enfant et par jour, 45 euros par adulte et par jour, avec le dernier jour gratuit pour les longs séjours

(**faites appel à la méthode** `estLongSejour` au sein de la méthode `coutEntreeParc`*).*

- La méthode `nbNuiteesEnfantsGratuites` qui va retourner le nombre de nuits d'hôtel pour enfant gratuites calculé comme suit : deux nuits pour enfant payantes donnent droit à une nuit pour enfant gratuite.
- La méthode `coutHotel` calcule le coût total des nuits à l'hôtel Disney comme suit : 49,9 euros par enfant et par nuit, 65,5 euros par adulte et par nuit, sans oublier de déduire les nuitées gratuites (**faites appel à la méthode** `nbNuiteesEnfantsGratuites` au sein de la méthode `coutHotel`*).*
- La méthode `coutParking` calcule le coût total des places de parking comme suit: 8 euros par véhicule et par jour mais gratuit si accompagné d'une personne à mobilité réduite.
- La méthode `coutTotal` calcule le coût total du séjour : **faites appel aux méthodes** `coutEntreeParc`**,** `coutHotel` ***et*** `coutParking` au sein de la méthode `coutTotal`*.*
- La méthode `aAccesPrioritaire` retourne un booléen : la présence d'une personne à mobilité réduite donne droit à un accès prioritaire dans les files d'attente du parc d'attraction.
- La méthode `resumeSejour` retourne la chaîne de caractères décrivant le séjour en respectant la structure ci-dessous (à vous de déduire les valeurs qui doivent remplacer les pointillés).

  S'il y a présence d'une personne à mobilité réduite :

  Le séjour chez Disney de … jour(s) et … nuit(s)
  pour … enfant(s) et … adulte(s) avec … véhicule(s)
  pour un montant total de … euros
  avec accès prioritaire ↳ *Appel à la méthode* ***coutTotal***

  S'il n'y a pas présence d'une personne à mobilité réduite :

  Le séjour chez Disney de … jour(s) et … nuit(s)
  pour … enfant(s) et … adulte(s) avec … véhicule(s)
  pour un montant total de … euros

Étape 4 : Création d'objets et appels de méthode

Dans la méthode `main` de la classe `Principale` :

- créez et initialisez un objet appelé `sejour` de type `SejourDisney` *;*
- affichez à la console son résumé (`resumeSejour`*) ;*

- appelez la méthode `estLongSejour` sur cet objet `sejour` : si c’est un long séjour affichez à la console "Vous avez droit à un bon de réduction de 20% sur votre prochain séjour !", sinon affichez "Merci d’avoir choisi un séjour chez Disney!".
- créez d’autres objets de type `SejourDisney` et affichez leur résumé.