TP n° 1

Bases

Exercice 1 : opérations

Écrire un programme qui transforme une durée donnée en secondes en heures, minutes et secondes.

Exercice 2 : somme de deux `byte`

Écrire un programme dans lequel on fait l'addition de deux variables de type `byte` ayant pour valeurs 114 et 25. Interpréter le résultat obtenu.

Exercice 3 : trouver le jour

Soit $\Delta = (j, m, a)$ la représentation d'une date où j est le jour, m le mois et a l'année. On désigne par $\mathrm{E}(x)$ la partie entière d'un réel x. On définit les entiers t, b, c et d par :

$$t = \begin{cases} m+10 & \text{si } m \leq 2 \\ m-2 & \text{sinon} \end{cases} \qquad b = \begin{cases} a-1 & \text{si } m \leq 2 \\ a & \text{sinon} \end{cases}$$

$$c = \mathrm{E}\left(b/100\right) \qquad d = b - 100c$$

Alors, la formule suivante, due à Zeller :

$$w = j + \mathrm{E}(2, 6t - 0, 2) + d + \mathrm{E}(d/4) + \mathrm{E}(c/4) + 5c$$

permet de calculer le jour de la semaine correspondant à la date Δ. En effet, le reste de la division entière de w par 7 est égal au numéro du jour de la semaine (0 pour Dimanche jusqu'à 6 pour Samedi).

Écrire un programme qui à partir du numéro du jour (de 1 à 31), du numéro du mois (de 1 à 12), de l'année (par exemple 1992) affiche le jour de la semaine.

Exercice 4 : la série harmonique

Écrire un programme qui détermine le plus petit entier n tel que

$$1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \geq l$$

On pourra prendre pour $l = 10$

Exercice 5 : entiers parfaits

Un entier est dit parfait s'il est égal à la somme de ses diviseurs autres que lui-même. Par exemple 6 est parfait car $6 = 1 + 2 + 3$. Écrire un programme qui cherche tous les entiers parfaits de 2 à 9000.

Exercice 6 : crible d'Eratosthène

Le crible d'Eratosthène est une méthode qui permet d'obtenir la liste des entiers premiers inférieurs à n. On rappelle qu'un entier est premier s'il n'est divisible que par 1 et par lui-même. On procède de la manière suivante :

- on prend la liste de tous les entiers de 1 à n et on raye le 1 qui n'est pas premier
- on raye tous les multiples de 2 sauf 2
- on recherche, à partir du dernier nombre pour lequel on a rayé les multiples, le premier nombre non rayé k et on raye tous les multiples de k sauf k lui-même
- on répète le traitement jusqu'au dernier entier k vérifiant $k \leq \sqrt{n}$

A la fin, les nombres non rayés sont les nombres premiers inférieurs à n. En utilisant le crible d'Eratosthène, écrire un programme qui affiche la liste de tous les entiers premiers inférieurs à n. On pourra prendre $n = 300$

TP n° 2

Utilisation de classes des API Java

Les API (Application Programming Interfaces) Java contiennent un nombre extrêmement important de classes. Dans les exercices proposés, vous aurez à utiliser des constructeurs, des méthodes, des attributs de quelques-unes de ces classes.

Vous devez avoir ouverte dans un navigateur la documentation de ces API. Elle est consultable à l'URL https://docs.oracle.com/javase/8/docs/api/ et téléchargeable à l'URL
http://www.oracle.com/technetwork/java/javase/documentation/jdk8-doc-downloads-2133158.html

Exercice 1 : chaînes de caractères

On considère la phrase suivante :

Before you start coding, it's a good idea to sit down and think about your problem

1. Créer une instance de la classe `String` avec le contenu ci-dessus.
2. Déterminer le nombre de caractères.
3. Déterminer le premier et le dernier caractère.
4. Écrire la chaine en majuscule.
5. Déterminer la première position du caractère "r" .
6. Déterminer le nombre de caractères "a". On pourra au préalable créer un tableau constitué des caractères de la chaine.

Exercice 2 : classes permettant de représenter des données primitives avec des objets

En plus des types primitifs, il existe des classes qui leur correspondent. Ce sont :

Types primitifs	Classes correspondantes
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

Ces classes enveloppes (appelées aussi wrappers) sont nées d'un besoin de pouvoir encapsuler des types primitifs dans des objets afin de pouvoir par exemple les mettre dans une collection. Les six premières classes enveloppes numériques ont des méthodes semblables qui sont `parseXXX`, `toString` et `valueOf`. A l'aide de la classe `Integer` :

1. À partir de la chaîne de caractère `"1580"`, créer le type primitif entier de même valeur et l'objet de type `Integer` de même valeur
2. À partir de l'entier 45789, construire la chaine de caractères représentant la même valeur
3. Écrire l'entier 14526 sous forme binaire et hexadécimale

Exercice 3 : tableaux dynamiques

La classe `ArrayList` permet de gérer des tableaux dynamiques et possèdent des méthodes pour les manipuler. Les instances de la classe `ArrayList` ne peuvent contenir que des objets. Il faudra donc penser éventuellement à utiliser les `wrappers` vus à l'exercice précédent si l'on manipule des nombres.

1. Créer une instance vide nommée `list1` de la classe `ArrayList` contenant des entiers et lui ajouter successivement les entiers 4, 7 et 3 puis l'afficher.
2. Créer le tableau d'entiers suivants : `int[] tab = {1,6,9,3,5}`. Créer une deuxième instance nommée `list2` de `Arraylist` contenant les éléments du tableau puis l'afficher.
3. Ajouter `list2` à `list1` puis afficher `list1`.
4. Déterminer si `list1` contient l'entier 3 et si c'est le cas, donner l'index de la première occurence.
5. Donner l'élément d'index 5
6. Supprimer le deuxième élément de `list1`
7. Donner la taille de `list1`

8. On souhaite trier `list1` dans l'ordre croissant ou décroissant. Pour cela, on peut utiliser la méthode `sort`. Avec celle-ci, il faut passer un objet d'une classe qui implémente l'interface `Comparator`. Java sait trier des nombres et des chaines de caractères et cette interface `Comparator` possède deux méthodes de classe nommées `naturalOrder()` et `reverseOrder()` qui répondent à notre souhait.

   Trier `list1` dans l'ordre décroissant et l'afficher.

Exercice 4 : dates et heures

La gestion des dates et des heures est un problème difficile dans les langages. Avec Java 8 vient une nouvelle API de gestion du temps nommée `java.time`. Dans l'exercice suivant, vous allez utiliser quelques classes.

1. Créer une instance de la classe `LocalDateTime` représentant le moment présent.
2. À l'aide de la classe `DateTimeFormatter`, créer un format de date et heure sous une forme permettant un affichage comme par exemple «samedi 11 février 2017 14:35» et afficher le moment présent.
3. Ajouter 2 heures et 45 minutes à ce moment présent et afficher le résultat.
4. Créer une instance de la classe `LocalDate` représentant votre date de naissance puis créer un format d'affichage et afficher votre date de naissance
5. À l'aide de la classe `Period`, déterminer votre âge en nombre d'années, de mois et de jours

TP n° 3

Conception : classes, héritage

Exercice 1 : personnes

On souhaite créer une classe `Personne` possédant trois attributs :
— un prénom et un nom qui ne peuvent pas être modifiés
— un surnom qui peut être modifié mais qui peut ne pas exister

Il faudra pouvoir construire une personne avec un prénom et un nom mais également avec un prénom, un nom et un surnom. Il faut pouvoir accéder aux trois attributs et redéfinir la méthode `toString()` en tenant compte de l'existence ou non du surnom.

1. Implémenter la classe `Personne`.
2. Créer une instance de `Personne` avec un prénom et un nom seulement et l'afficher
3. Créer deux instances de `Personne` avec un prénom, un nom et un surnom puis les afficher. Modifier le surnom de l'une d'elles et l'afficher.
4. Créer un tableau contenant les trois personnes et dans une boucle afficher le prénom de chacune.

Exercice 2 : intervalles

On souhaite créer des objets représentant des intervalles fermés $[a, b]$ où a et b sont des entiers. Pour simplifier, on admettra que $a \leqslant b$. Par défaut, on devra avoir $a = 0$ et $b = 1$. Il faut des méthodes qui permettent de :
— obtenir la longueur d'un intervalle
— obtenir le milieu d'un intervalle
— savoir si un réel est ou non dans l'intervalle
— construire le symétrique d'un intervalle par rapport à 0
— afficher l'intervalle sous la forme $[a, b]$

Tester en créant plusieurs intervalles (dont un par défaut) et en appliquant les méthodes.

Exercice 3 : triangles

On souhaite créer des objets représentant des triangles. Un triangle est caractérisé par ses trois longueurs non modifiables qui sont des nombres réels positifs. Pour le(s) constructeur(s), on admettra que le triangle existe, c'est à dire que le plus grand des trois nombres passés en paramètre est plus petit que la somme des deux autres. On souhaite également représenter des triangles rectangles que l'on caractérisera par les longueurs des deux côtés de l'angle droit. L'aire d'un triangle est donné par la formule de Héron (Héron d'Alexandrie, 1er siècle ap JC) dans laquelle a, b et c représentent les longueurs des côtés :

$$p = \frac{a+b+c}{2} \qquad S = \sqrt{p(p-a)(p-b)(p-c)}$$

Il faut des méthodes qui permettent de :

- savoir si un triangle est équilatéral
- savoir si un triangle est rectangle
- obtenir l'aire d'un triangle
- obtenir l'hypoténuse d'un triangle rectangle

1. Implémenter les classes nécessaires.
2. Construire le triangle dont les côtés ont pour longueur 36, 77 et 85. Donner son aire et tester s'il est rectangle.
3. Construire le triangle rectangle dont les côtés de l'angle droit ont pour longueur 28 et 45. Donner la longueur de l'hypoténuse.
4. On souhaiterait créer un triangle connaissant un côté c et les deux angles α et β qui le bordent. Grâce à la formule des sinus, on sait trouver les deux autres côtés :

   $$a = \frac{c\,\sin(\alpha)}{\sin(\alpha+\beta)} \qquad b = \frac{c\,\sin(\beta)}{\sin(\alpha+\beta)}$$

   Comment faire ? Pensez aux dates et heures ?

TP n° 4

Exceptions, adresses IP, en-têtes http

Dans tous les exercices, les exceptions doivent être capturées dans un bloc `try ... catch`

Exercice 1 : capture des exceptions

1. Exécuter le code suivant :

```
int a = 5;
int b = 0;
System.out.println("Le résultat de la division de " + a + " par " +
↪  b + "est : " + a/b);
```

   Vous obtenez alors une exception de type `ArithmeticException`.

2. Modifier le code précédent pou capturer cette exception

Exercice 2 : les adresses IP

Pour les adresses IP, on utilisera la classe `InetAddress` du package `java.net`. Dans cet exercice, vous devez obtenir deux choses :

— les adresses IP d'un nom DNS passé en argument, comme par exemple `google.com`
— l'adresse IP et le nom DNS de votre poste

Exercice 3 : en-têtes d'une réponse http

Pour lire les en-têtes d'une réponse http, il faut utiliser la classe `URLConnection`. Le nom de la méthode a utiliser ensuite est assez facile à trouver. Pour obtenir une instance de `URLConnection`, il faudra d'abord créer un objet de type `URL` et lui appliquer la méthode `openConnection()`. Vous testerez avec diverses URL comme par exemple `http://www.debian.org`

TP n° 5

Datagrammes UDP

Dans tous les exercices, les exceptions doivent être capturées dans un bloc `try ... catch`

Il est nécessaire de relire le chapitre du cours sur les datagrammes UDP.

Exercice 1 :

Un service `echo` UDP écoute sur le port 7. Lorsque un datagramme quelconque est reçu, un datagramme totalement identique est envoyé à l'émetteur.

Créer un client ou émetteur UDP qui enverra une chaine de caractères et afficher la réponse. La classe aura pour nom `ClientUdpEcho` et se lancera en ligne de commande par `java ClientUdpEcho <message> <serveur> <port>`. Pour le serveur, utiliser l'adresse IP `100.64.70.4`

Exercice 2 :

Créer un serveur ou récepteur UDP sur le port 12345. Celui-ci devra retourner la date du jour et l'heure dans un format français. La classe aura pour nom `ServeurUdpDaytime` et devra se lancer en ligne de commande par `java ServeurUdpDaytime <port>`. Afficher sur la console l'adresse du client et le port utilisé par le client. Pour «écouter», appliquer la méthode `receive` dans un boucle infinie `while(true)`. Pour tester, il est possible d'utiliser le client de l'exercice précédent.

TP n° 6

Les sockets TCP

Dans tous les exercices, les exceptions doivent être capturées dans un bloc `try ... catch`

Il est nécessaire de relire le chapitre du cours sur les sockets TCP.

Exercice 1 : serveur echo TCP

Créer un serveur TCP sur le port 45678. Celui-ci devra retourner en majuscule le message reçu du client. La classe aura pour nom `ServeurTcpEcho`.

Exercice 2 : client echo TCP

Créer un client echo TCP qui enverra une chaine de caractères au serveur de l'exercice précédent et affichera la réponse. La classe aura pour nom `ClientTcpEcho` et se lancera en ligne de commande par `java ClientTcpEcho <message> <serveur> <port>`. Pour le serveur, utiliser l'adresse IP `100.64.70.4`