Laboratoire 2

DURÉE PRÉVUE : 6H00

OBJECTIFS

Au terme de ce laboratoire, l'étudiant sera capable de :

- → Réaliser une phase préparatoire en amont du codage (identifier les informations clés, déduire les étapes de résolution, solutionner des cas concrets et valider les résultats obtenus).
- → Exploiter les variables et les constantes de type primitif et de type java.lang.**String**.
- → Respecter les conventions de nommage pour les variables (*CamelCase*) et les constantes (lettres majuscules et tiret bas) .
- → Effectuer des opérations à l'aide des opérateurs `+`, `-`, `*`, `/`, `%` et `=`.
- → Réaliser des appels de fonction et, le cas échéant, exploiter la valeur retournée.
- → Trouver une information dans la documentation officielle du Java.
- → Réaliser des acquisitions de données (utilisation de la classe *Console* disponible sur *HELMo Learn*) et afficher des informations formatées en console.

AVANT-PROPOS

CRÉER UN NOUVEAU PACKAGE

Créez dans votre espace de travail un dossier nommé `labo2` (en minuscules et sans espace). C'est dans ce dossier que vous enregistrerez tous les programmes de ce 2e labo.

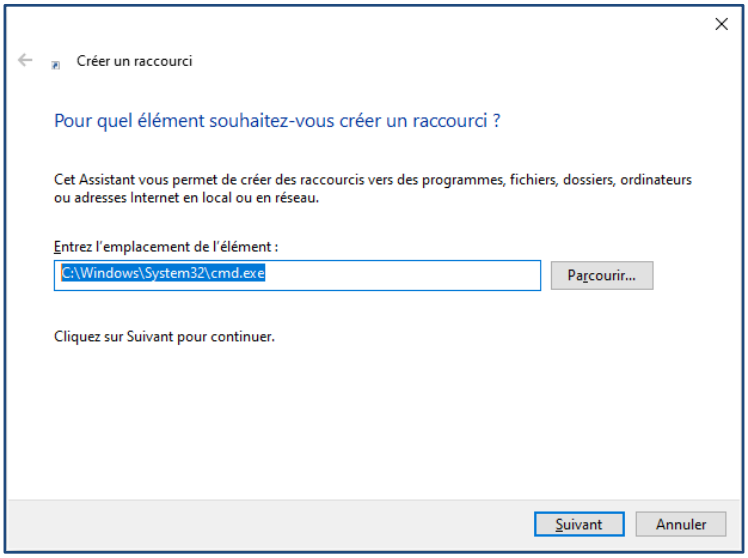
ACCÉDER PLUS RAPIDEMENT À L'ESPACE DE TRAVAIL

Si vous travaillez sous Windows, il est possible de faire en sorte que la console démarre directement dans votre espace de travail `Java`. Ainsi, il n'est plus nécessaire de saisir systématiquement la commande `cd`.

La procédure ci-après ne doit être réalisée qu'une seule fois :

1. Faites un clic droit sur une zone vide du *Bureau Windows* pour faire apparaître le menu contextuel, puis sélectionnez l'option permettant de créer un nouveau raccourci (***Nouveau*** > ***Raccourci***).

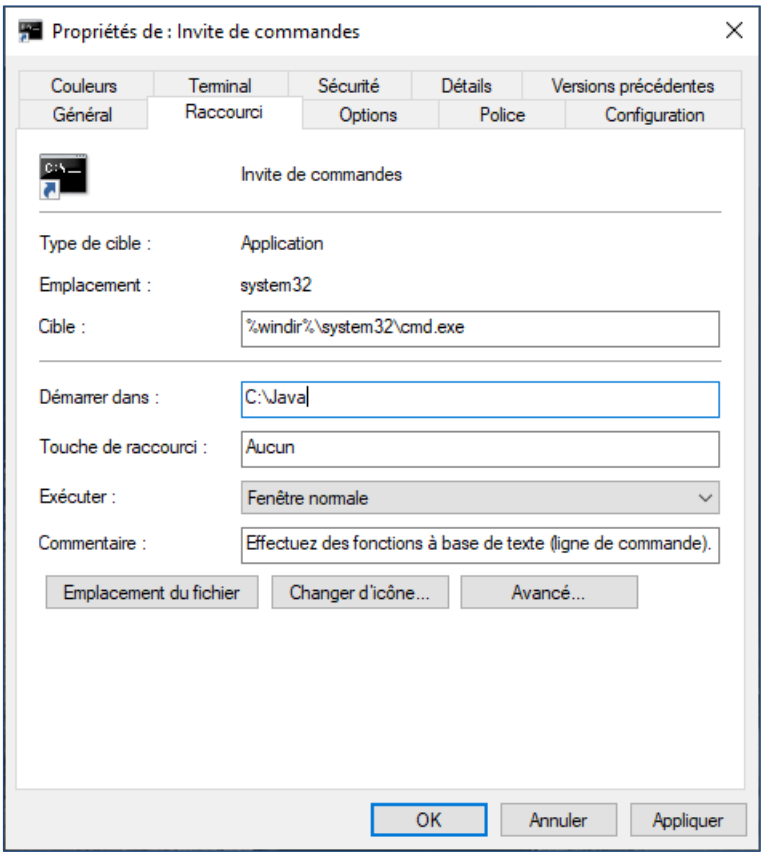
2. Dans la fenêtre qui s'ouvre, spécifiez le chemin d'accès de l'exécutable de la console, à savoir : « ***C:\Windows\System32\cmd.exe*** ». Cliquez sur le bouton ***Suivant***.



3. Spécifiez ensuite un nom à votre raccourci, par exemple : « *Console* ». Cliquez sur le bouton ***Terminer***.

4. De retour sur le *Bureau*, faites un clic droit sur l'icône qui est apparue. Dans le menu contextuel, sélectionnez ***Propriétés***.

5. Dans la fenêtre qui s'affiche, ouvrez l'onglet ***Raccourci***, puis saisissez le chemin de votre espace de travail (par exemple, « ***C:\Java*** ») dans le champ ***Démarrer dans***. Cliquez sur le bouton ***OK***.



6. Si vous le désirez, vous pouvez maintenant épinglez le raccourci à la *Barre des tâches*.

EXERCICE 1 : PHASE PRÉPARATOIRE ET RÉALISATION D'UN PREMIER PROGRAMME

PHASE PRÉPARATOIRE

La phase préparatoire est une phase de réflexion qui **précède** le codage de l'application.

Elle permet entre autres d'anticiper les difficultés, de dégager une méthode de résolution et de valider les résultats obtenus par l'application.

Pour ce premier exercice, la phase préparatoire vous est donnée (voir le document PDF sur *HELMo Learn*). A partir de l'exercice 2, il vous sera demandé de réaliser systématiquement cette phase préparatoire **AVANT** de coder la solution.

DESCRIPTION DU PROGRAMME

Le programme à réaliser doit être capable d'exprimer l'âge d'un chien sur une échelle humaine. En effet, un chien ne vieillit pas au même rythme qu'un être humain.

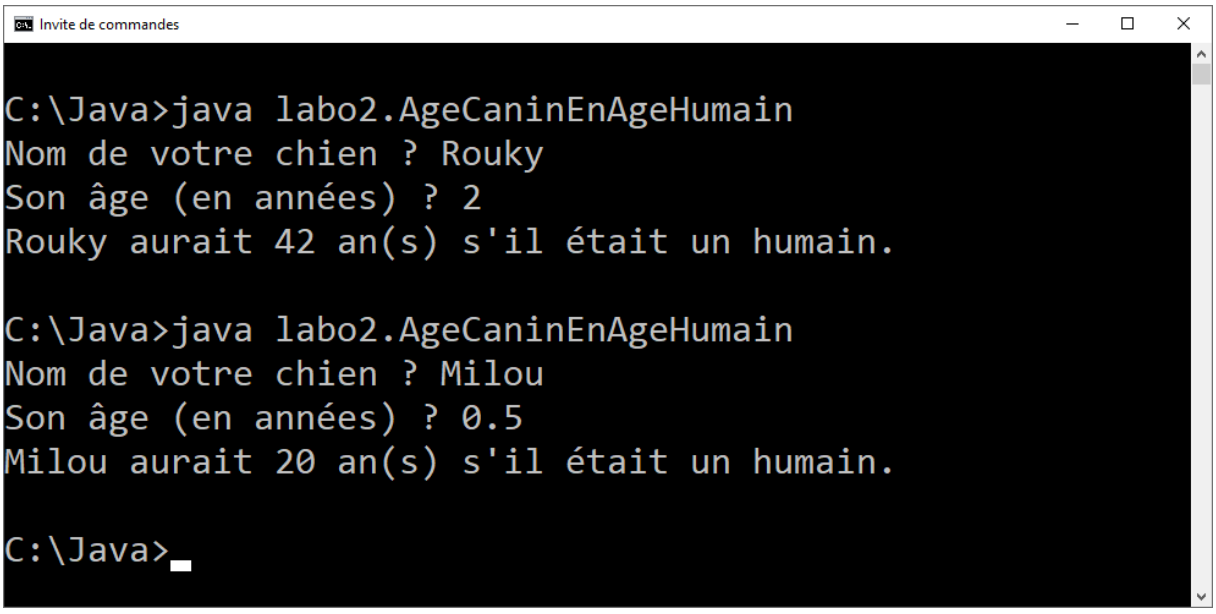
Pour simplifier la détermination de l'âge humain d'un chien, nous utiliserons la formule suivante :

$$ageHumain = 16 \times ln(ageCanin) + 31$$

Les âges sont ici exprimés en années. Il est également à noter que le logarithme ln utilisé est le *logarithme naturel* (ou *népérien*).

Selon cette formule, un chien de 2 ans correspondrait à une personne âgée d'environ 42 ans.

EXEMPLES D'EXÉCUTION



CONSIGNES

1. Téléchargez le document PDF reprenant la phase préparatoire du programme nommé **`AgeCaninEnAgeHumain`**. Lisez ensuite attentivement cette dernière.
2. Téléchargez le fichier **`Console.java`**, puis placez-le dans le dossier **`labo2`**.

   [IMPORTANT] Dans ce fichier, vous devez renommer le package **`io`** en **`labo2`** !

3. Dans le dossier **`labo2`**, créez un fichier nommé **`AgeCaninEnAgeHumain.java`**.
4. Dans ce fichier, déclarez une classe nommée **`AgeCaninEnAgeHumain`** et la fonction principale *main*.
5. En suivant la phase préparatoire (en particulier, la section 3 *« Programmer la solution »*), codez la solution dans la fonction *main*.

   [IMPORTANT] Veillez à obtenir un résultat à l'affichage le plus proche possible de celui montré dans les exemples d'exécution précédents.

   CONSEILS

   Pour réaliser ce programme, trois outils en particulier vous seront utiles :

   - Les fonctions statiques **`Math.log`** pour calculer le logarithme naturel et **`Math.round`** pour effectuer un arrondi.

     Veuillez consulter la documentation officielle à ce sujet :

     https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/lang/Math.html

   - L'opérateur de conversion explicite **`(...)`**.

6. Compilez et exécutez le programme afin de vérifier que le résultat obtenu correspond au résultat attendu (voir les exemples d'exécution précédents).

EXERCICE 2 : TÉLÉCHARGEMENT

PHASE PRÉPARATOIRE

Réalisez une phase préparatoire **AVANT** de coder la solution.

Pour ce faire, téléchargez le document DOCX « *Phase préparatoire à compléter* » situé à la fin de la première section, nommée *Outils et fichiers*, de la page *HELMo Learn* du cours.

Après une dizaine de minutes, l'enseignant réalisera collectivement une correction de la phase préparatoire pour cet exercice.

DESCRIPTION DU PROGRAMME

Le programme à réaliser doit être capable de calculer la durée (en heures, minutes, secondes) nécessaire pour télécharger un fichier sur base :

a. De la taille du fichier exprimée en Mo (mégaoctets) .

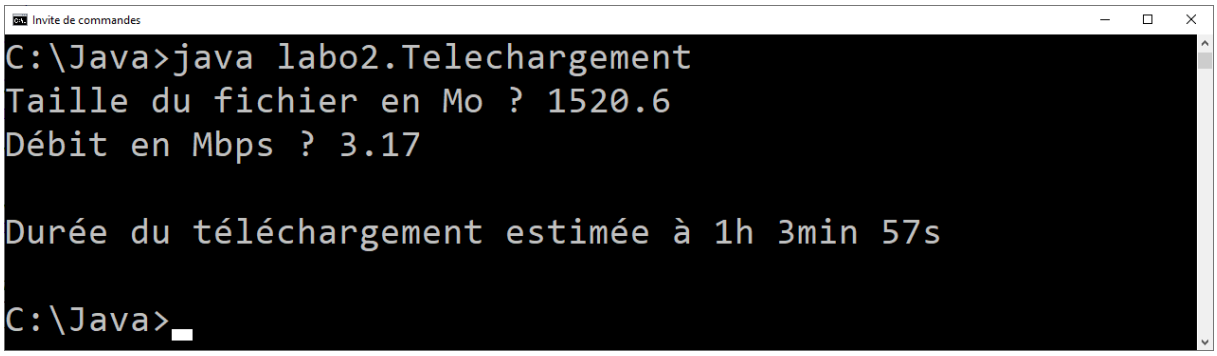
b. Du débit descendant de la ligne d'accès exprimée en Mbps (mégabits par seconde) .

REMARQUE

L'utilisateur peut se rendre sur le site *www.speedtest.net* pour mesurer le débit descendant de sa ligne d'accès.

EXEMPLE D'EXÉCUTION

Afin de vous aider dans cette première phase préparatoire, vous pouvez utiliser le cas concret de l'exemple ci-dessous.



PHASE PRÉPARATOIRE

Quelques conseils pour la réalisation de cette dernière :

1. Pour la résolution du 1er cas concret, choisissez les mêmes valeurs en entrée que dans l'exemple d'exécution précédent, à savoir : un débit descendant de 3,17 Mbps et une taille de fichier de 1520,6 Mo.

   Vous pourrez ainsi confronter vos résultats à ceux de cet exemple.

2. Pour faciliter les calculs à réaliser lors de la résolution des cas concrets, utilisez l'outil ***Numbr*** : https://numbr.dev/.

3. Soyez particulièrement vigilant à une règle de trois qui doit être utilisée lors de la résolution !

PHASE DE RÉALISATION

1. Dans le dossier **labo2**, créez un fichier nommé **Telechargement.java**.
2. Déclarez une classe nommée **Telechargement** et la fonction principale *main*.
3. Codez votre solution sur base de votre phase préparatoire.
4. Compilez, puis testez votre programme afin de vérifier que le résultat obtenu correspond à celui attendu.
5. **[QUALITÉ DU CODE]** Est-ce que vos identifiants respectent les conventions de nommage ? Sont-ils révélateurs des données affectées ? Avez-vous ajouté des commentaires afin d'améliorer la lisibilité du programme ?
6. **[AMÉLIORATION POSSIBLE]** Réalisez l'acquisition du nom du fichier et affichez en fin d'exécution un message de la forme : *« Durée du téléchargement du fichier "Fedora-Xfce-Live-x86_64-32-1.6.iso" estimée à 1h 3min 57sec »*.

EXERCICE 3 : VENTE DE VOITURES

PHASE PRÉPARATOIRE

Dorénavant, vous devrez **réaliser systématiquement** une phase préparatoire avant de coder la solution !

DESCRIPTION DU PROGRAMME

Un concessionnaire automobile souhaite un programme lui facilitant le calcul du montant TVAC qu'un client doit lui payer à l'achat d'une voiture.

Les montants des voitures exposées dans le showroom sont les prix de vente HTVA. Seule cette valeur doit être fournie au début de l'exécution du programme.

La TVA pratiquée à l'achat d'une voiture en Belgique est de 21%.

En raison des conditions salon sur les marques vendues, une réduction de 3% est accordée à chaque achat. Cette réduction porte sur le montant TVAC.

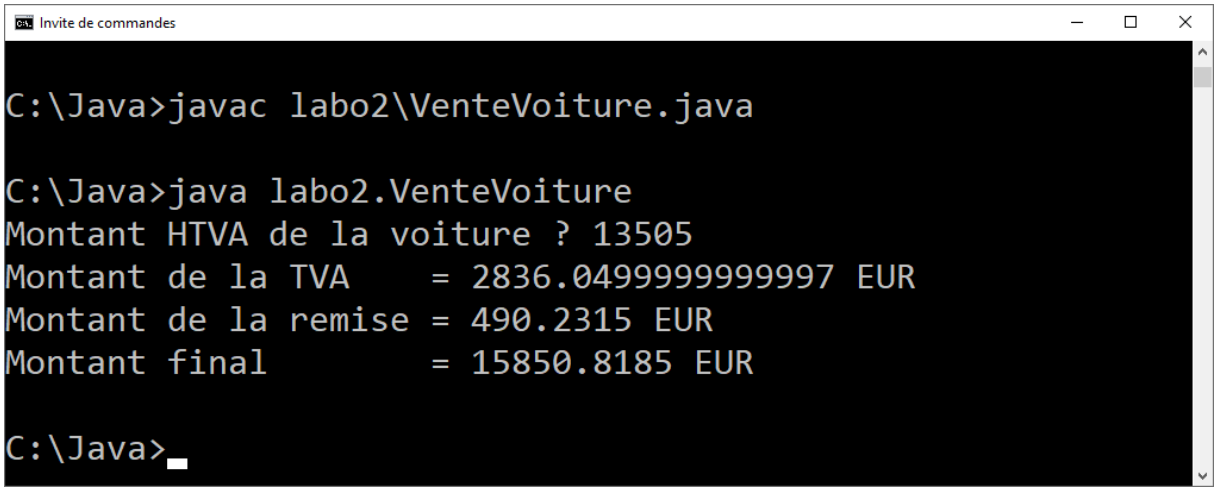
En fin d'exécution, le programme doit afficher un récapitulatif reprenant :

a. Le montant de la TVA.

b. Le montant de la remise.

c. Le montant final que le client doit payer (en incluant la TVA et la remise) .

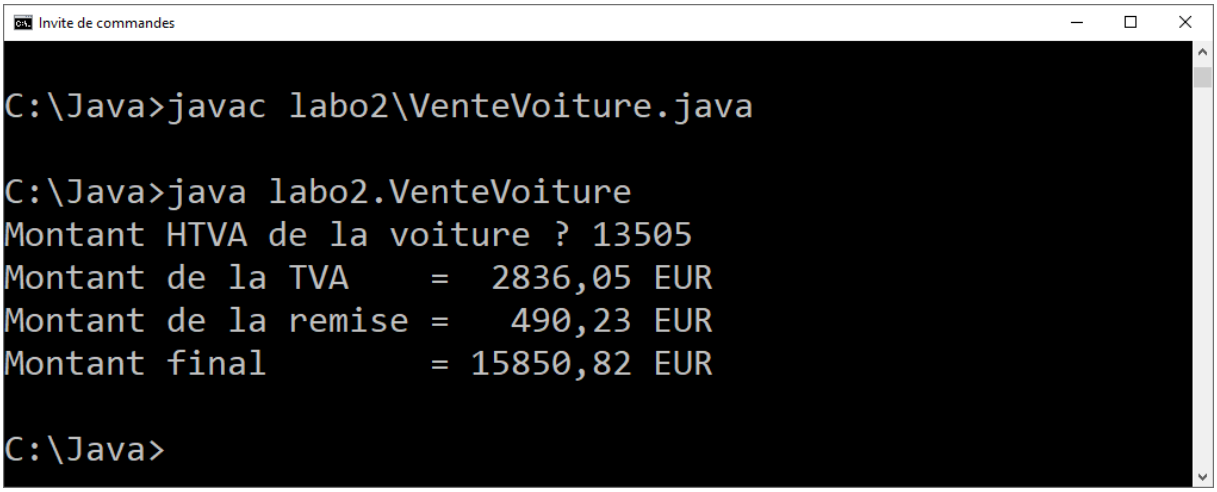
EXEMPLE D'EXÉCUTION N°1



CONCEPTION ET RÉALISATION

1. Comme dans l'exercice 2, effectuez la phase préparatoire avant la phase de programmation.
2. Le programme doit être codé dans un fichier nommé **VenteVoiture.java**.
3. Utilisez deux constantes nommées **TAUX_TVA** et **TAUX_REDUCTION** pour remplacer les littéraux `0.21` et `0.03` apparaissant dans votre traitement.
4. **[AMÉLIORATION POSSIBLE]** Utilisez la fonction `printf` afin d'améliorer la pertinence des résultats affichés (voir l'exemple d'exécution n°2 ci-après).

EXEMPLE D'EXÉCUTION N°2



EXERCICE 4 : HEURES PRESTÉES

DESCRIPTION DU PROGRAMME

Un consultant souhaite un programme lui permettant de calculer la durée totale de ses prestations hebdomadaires et le montant total à facturer en conséquence à son employeur. Sa rémunération actuelle est de 128, 50€/heure.

Afin de comptabiliser le nombre total d'heures prestées durant la semaine (du lundi au vendredi), les heures prestées chaque jour doivent être fournies au début de l'exécution en respectant le format *[h]h:[m]m*. Un chiffre (*h* ou *m*) spécifié entre crochets signifie qu'il n'est pas obligatoire.

Par exemple, pour exprimer « 7:05 », l'utilisateur peut saisir indifféremment un ou deux chiffres pour les heures et les minutes. Les écritures suivantes sont dès lors tolérées : « 07:05 », « 07:5 » et « 7:5 ».

REMARQUE

La difficulté de cet exercice réside dans l'extraction des heures et des minutes à partir des chaînes de caractères saisies par l'utilisateur.

Pour y parvenir, utilisez ces trois fonctions :

→ La méthode d'instance `indexOf` de la classe **java.lang.String**.

→ La méthode d'instance `substring` de la classe **java.lang.String**. N'oubliez pas qu'il existe deux surcharges de cette méthode.

→ La méthode statique *`parseInt`* de la classe **java.lang.Integer**.

Consultez la documentation Java pour comprendre leur fonctionnement :

https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/lang/String.html

EXEMPLE D'EXÉCUTION

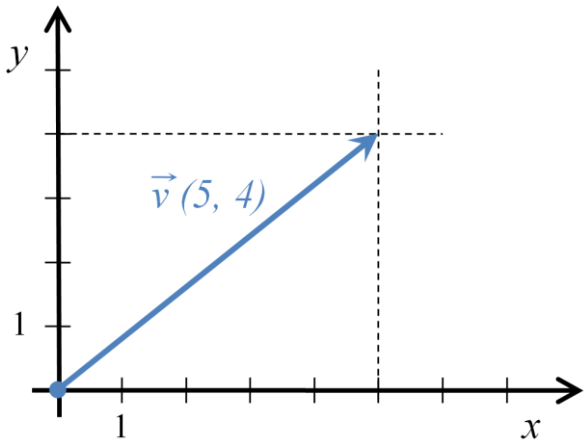
CONCEPTION ET RÉALISATION

1. Effectuez la phase préparatoire avant la phase de programmation.
2. Le programme doit être codé dans un fichier nommé **HeuresPrestees.java**.
3. Utilisez une constante nommée **TARIF_PAR_HEURE** pour remplacer le littéral `128.5` apparaissant dans votre traitement.
4. Lors de la réalisation de ce programme, veillez à structurer au mieux votre code. En effet, le nombre d'instructions nécessaires est conséquent. Pour ce faire :
   - → Soignez la mise en forme du code (tabulations, lignes vides).
   - → Utilisez des variables nommées de manière non ambigüe.
   - → Commentez les étapes du traitement.

EXERCICE 5 [FACULTATIF] : NORME D'UN VECTEUR

DESCRIPTION DU PROGRAMME

Un professeur de mathématiques souhaite un programme capable de calculer la norme d'un vecteur du plan afin d'aider ses élèves. Avant d'être affichée, la valeur de la norme doit être arrondie à 1 chiffre après la virgule.



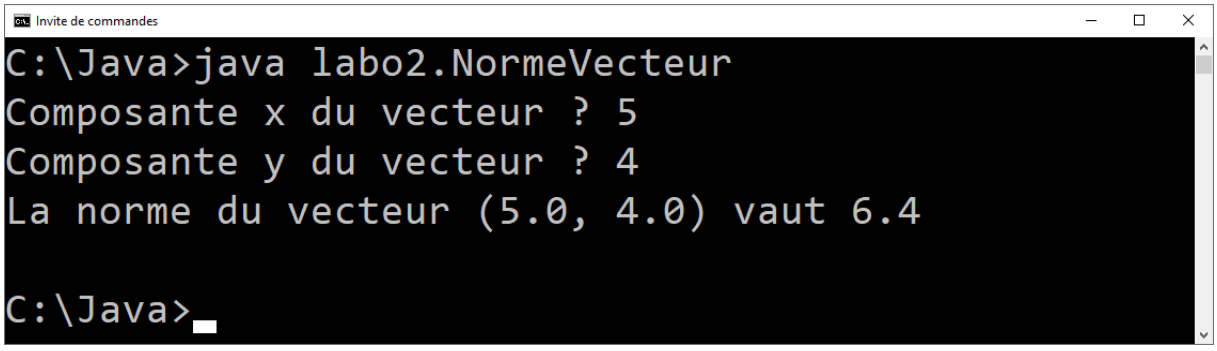
Pour rappel, la norme d'un vecteur est sa longueur.

Si le vecteur $\vec{v}$ a pour composantes (x, y), sa norme s'écrit :

$$\|\vec{v}\| = \sqrt{x^2 + y^2}$$

En fin d'exécution, le programme doit afficher le résultat sous la forme suivante : *« La norme du vecteur (5.0, 4.0) vaut 6.4 »*.

EXEMPLE D'EXÉCUTION



CONCEPTION ET RÉALISATION

1. Effectuez la phase préparatoire avant la phase de programmation.
2. Le programme doit être codé dans un fichier nommé **NormeVecteur.java**.

   [REMARQUE] Les composantes du vecteur saisies par l'utilisateur peuvent être des valeurs réelles.

3. N'oubliez pas de **calculer l'arrondi** de la norme durant l'étape du traitement des données. Les fonctions de formatage, telles que `printf` et *`format`*, ne peuvent donc être utilisées !

   Assurez-vous que l'arrondi obtenu est correct en calculant la norme du vecteur (9, 16). Celle-ci vaut 18,4 une fois arrondie à 1 chiffre après la virgule.

4. Aidez-vous des fonctions statiques **`Math.`***`sqrt`* et **`Math.`***`round`*.

   Veuillez consulter la documentation officielle à ce sujet :

   https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/lang/Math.html