L02C\_Types en Java

Table des matières

[1 Questions 2](#_Toc141180651)

[2 Code 3](#_Toc141180652)

[2.1 Classes Operation et L02C(main) 3](#_Toc141180653)

[2.2 Classe OutilsTexte et méthode isPalindrome 4](#_Toc141180654)

[2.3 Méthode StringToInt() 4](#_Toc141180655)

[2.4 Méthode vraiOuFaux() 4](#_Toc141180656)

# Questions

1. Donner la liste des types primitifs en Java :

|  |
| --- |
| byte  short  int  long  float  double  char  boolean |

1. Identifier si le type est primitif ou une classe ou pas un type (invalide):

|  |  |
| --- | --- |
| **Type** | **Primitif ou classe ou invalide** |
| String | Classe |
| Int | invalide |
| Double | Classe |
| char | Primitif |
| short | Primitif |
| Boolean | Classe |
| Integer | Classe |
| long | Primitif |
| Integer | Classe |
| Byte | Classe |
| float | primitif |
| character | invalide |

1. Trouver la méthode d’un type **wrapper** qui permet d’exécuter l’action décrite :

|  |  |
| --- | --- |
| **Action** | **Méthodes** |
| Convertir la donnée en chaine de caractère (String) | toString() |
| Convertir la donnée en entier (int) | intValue() |
| Convertir la donnée en réel (double) | doubleValue() |

1. Trouver la méthode de la classe **String** qui permet d’exécuter l’action décrite

|  |  |
| --- | --- |
| **Action** | **Méthodes** |
| Trouver le nombre de caractères dans le String | length() |
| Comparer 2 chaines de caractères | compareTo() |
| Trouver la position d'un caractère (première occurence) dans le String | indexOf() |
| Trouver le caractère présent à une certaine position | charAt() |
| Chercher si une chaine de caractères est présente dans le String | contains() et aussi matches() |
| Séparer une chaine de caractères en plusieurs chaines selon un caractère séparateur | split() |

1. Quel opérateur en Java permet de trouver le type d’une variable?

|  |
| --- |
| instanceOf() |

# Code

## Classes Operation et L02C(main)

1. Créez un nouveau projet **L02C-TypesEnJava**
2. Créez un nouveau nouveau package **types.acme\_cvm.ca**
3. Créez une classe **Operation** avec :
   1. Aucun champ
   2. Une méthode **addition** :
      * Cette méthode accepte 2 paramètres **n1** et **n2** de type **int** et retourne un **int** égal à la somme de ces 2 paramètres
   3. Une méthode **soustraction** :
      * Cette méthode accepte 2 paramètres **n1** et **n2** de type **Integer** et retourne un **Integer** égale à **n1** -**n2**
   4. Une méthode **multiplication** :
      * Cette méthode accepte 2 paramètres **n1** et **n2** de type **double** et retourne un **double** égal à n1 \* n2
   5. Une méthode **division** :
      * Cette méthode accepte 2 paramètres **n1** et **n2** de type **Double** et retourne un **Double** égal à n1 / n2
4. Dans le même package, créez une classe **L02C** avec :
   1. Une unique méthode main.
   2. Dans le main, écrivez le code selon les consignes ci-dessous :
   3. Essayez d’afficher à l’écran le résultat des 4 opérations (addition, soustraction, multiplication, division) en passant **i1** et **i2** en paramètres à toutes les opérations.
   4. On essaie de passer une variable de type **int** à des paramètres de types différents. Quelle(s) opérations ne fonctionnent pas?

|  |
| --- |
| La division ne fonctionne pas |

* 1. Changez les paramètres pour essayer de passer **wi1** et **wi2** aux 4 opérations.
  2. On essaie de passer une variable de type **Integer** à des paramètres de types différents. Quelle(s) opérations ne fonctionnent pas?

|  |
| --- |
| La division ne fonctionne pas |

* 1. Recommencez en essayant de passer **d1** et **d2** aux 4 opérations.
  2. On essaie de passer une variable de type **double** à des paramètres de types différents. Quelle(s) opérations ne fonctionnent pas?

|  |
| --- |
| L’addition et la soustraction ne fonctionnent pas |

1. Recommencez en essayant de passer **wd1** et **wd2** aux 4 opérations.
2. On essaie de passer une variable de type **Double** à des paramètres de types différents. Quelle(s) opérations ne fonctionnent pas?

|  |
| --- |
| L’addition et la soustraction ne fonctionnent pas |

1. Que concluez-vous au sujet de la conversion automatique de types par Java?

|  |
| --- |
| Java convertit automatiquement un type avec son wrapper correspondant et vice-versa  Java ne convertit pas un réel en entier automatiquement (il pourrait y avoir perte de donnée pour les décimales)  Java convertit des entiers en réels automatiquement (il n’y a pas de perte de donnée) |

1. Utilisez la méthode appropriée du type wrapper Double pour convertir wd1 et wd2 en int et passez les résultats en paramètres aux opérations d’addition et de soustraction.
2. Est-ce que ça fonctionne? Qu’arrive-t-il aux décimales?

|  |
| --- |
| Ça fonctionne pour les 2 méthodes.  Les décimales sont tronquées (pas arrondies) |

## Classe OutilsTexte et méthode isPalindrome

1. Faites une nouvelle classe **OutilsTexte** et ajoutez-y une méthode **isPalindrome** qui permet de vérifier si une chaine de texte est un palindrome en respectant les consignes ci-dessous :

* Un palindrome est du texte qui reste identique lorsqu’écrit à l’envers (exemple : laval)
* Accepte 1 chaine de caractères à vérifier
* Retourne **true** si la chaine est un palindrome et **false** sinon
* Truc : utilisez une boucle qui passe chaque caractère de la chaine en commençant par la fin et stockez le résultat dans une variable locale, puis comparez avec la chaine de caractères originale. On peut ajouter un caractère à une chaine de plusieurs façons, dont la plus facile est simplement avec l’opérateur **+**.

1. Dans le **main**, créez un objet approprié pour pouvoir accéder à la méthode, puis utilisez cette méthode à l’intérieur d’une condition **if** pour produire à la console le texte :
   * La chaine de texte *chainePalindrome* est un palindrome

ou

* + La chaine de texte *chainePalindrome* n’est pas un palindrome.

## Méthode StringToInt()

1. Dans la classe **OutilsTexte**, créez une méthode **stringToInt** qui convertit un chiffre reçu sous forme de texte en **int**. Respectez les consignes suivantes :

* La méthode accepte une chaine de caractères et retourne un **int**
* Truc : utilisez la méthode **valueOf()** de la classe **Integer**
* Comment peut-on gérer les erreurs si le texte reçu en paramètre n’est pas numérique?

1. Dans le **main**, créez une variable locale **chiffreEnTexte** de type String avec une valeur de "201" et utilisez cette méthode pour afficher la somme de 100 et **chiffreEnTexte** à l’écran. Ensuite, essayez avec d’autres valeurs de **chiffreEnTexte**, incluant des valeurs non numériques.

* Note 1: toute donnée entrée dans un formulaire de page web est du texte. Il doit souvent être convertit lorsque la donnée est numérique.
* Note 2 : la méthode **parseInt()** de certains wrappers est souvent utilisée à cet effet.
* Qu’arrive-t-il lorsqu’on assigne du texte non numérique au paramètre de cette méthode?

## Méthode vraiOuFaux()

1. Dans la classe **OutilsTexte**, créez une méthode **vraiOuFaux** qui vérifie si du texte entré indique vrai ou faux en respectant les consignes suivantes :

* La méthode accepte un String et retourne un int
* Si le texte est "v" ou "vrai" indépendamment de la casse, retourne 1
* Si le texte est "f" ou "false" indépendamment de la casse, retourne 0
* Si le texte n’est aucune de ces options, retourne -1

1. Dans le **main**, ajoutez le code requis pour tester votre méthode en affichant un résultat à l’écran.