NOM :	PRÉNOM :	GROUPE :
Ouazouz	**Sami**	**I1**

Méthode : displayCharString

1. DESCRIPTION DU « QUOI » = COMPRENDRE LE PROBLÈME

Nom de la méthode?

displayCharString

Sa tâche ? *Expliquer brièvement le traitement que doit réaliser la fonction/méthode*

Dans une chaîne de caractères, on veut afficher tel nombre de caractères on fonction du nombre précédent celui-ci.

Préconditions (= ses entrées et les valeurs admises) ? *Lister et nommer les données nécessaires à la réalisation du traitement – ces données peuvent varier d'une exécution à l'autre de la méthode. Dans un contexte orienté objets, listez aussi les champs de l'objet dont la méthode a besoin mais qu'elle ne modifie pas.*

A côté de chaque donnée ou champ de la liste, indiquez les éventuelles conditions qu'il/elle doit remplir pour que la méthode puisse réaliser le traitement demandé.

Une référence non-null vers une chaine de caractères.

Postconditions (= sa sortie et les garanties concernant celle-ci) ? *Indiquer la donnée qui doit être retournée au terme du traitement réalisé par la fonction. Dans un contexte orienté objets, listez et nommez aussi les champs de l'objet que la méthode initialise ou modifie.*

A côté du résultat et des champs initialisés ou modifiés, indiquez les conditions qu'il doit remplir pour être considéré comme correct.

On a affiché la totalité de la chaine et en sortie on a ce qu'on désirait.

Son plan de test = résolution d'au moins six cas concrets : 2 cas « normaux », 1 cas par « branchement » réalisé par la méthode, 1 cas d'erreur géré par la méthode, 1 cas limite (ensemble vide, première ou dernière valeur, zéro ou nul, etc.), éventuellement un cas de dépassement de capacité, 1 à 2 combinaisons de cas

Pour chacun, indiquer 1. le type de cas 2. les entrées spécifiées et les champs utilisés en lecture 3. les effets attendus = le résultat et les valeurs des champs d'objets initialisés ou modifiés

- "10X" ➔ XXXXXXXXXX
- "0X" ➔ **Interdit** parce que X doit être précédé d'un nombre

• "010X" ➔ \nXXXXXXXXXX -> le 0 avant un nombre nous permet de faire un retour à la ligne.

2. FORMALISATION DES TESTS = PRÉPARER LEUR ÉCRITURE EN JAVA

La traduction de chaque test du plan de test

Ecrivez chaque test en respectant le schéma Given-When-Then. Exemple : étant donné un objet dictionnaire contenant les 5 mots arbre, fleur, chien, chat, ordinateur et leurs définitions, quand j'appelle la méthode de recherche avec le mot « arbre » en entrée, alors j'obtiens bien la définition d'un arbre

Etant donné une String 4D, quand j'appelle la fonction avec 4D en argument, alors j'obtiens une chaine « DDDD ».

Etant donné une String 3C06B, quand j'appelle la fonction avec 3C06B en argument, alors j'obtiens « CCC

BBBBBB ».

Etant donné une String 4D0G, quand j'appelle la fonction avec 4D0G en argument, alors j'obtiens une erreur parce qu'après le 0 il n'y a pas de chiffre

3. DESCRIPTION DU « COMMENT » = IMAGINER UN ALGORITHME

Nom de l'algorithme ? *Si vous appliquez un algorithme célèbre, indiquez son nom.*

Boucle avec branchements.

Dessin. Réalisez un dessin de la situation finale de votre problème et déduisez-en une situation plus générale

Ses étapes ? *Répertorier textuellement ou en « pseudocode » ou encore sous forme d'organigramme, dans l'ordre, les étapes principales que la méthode doit appliquer pour résoudre le problème pour toutes les entrées possibles. Vérifiez que votre algorithme « fonctionne » au moins pour tous les cas de tests prévus ! Assurez-vous de la* ***terminaison*** *de votre algorithme*

- D'abord on met tout dans une boucle for, on vérifie si le premier char de la String est un nombre. Pour ça on peut utiliser **if**(!Character.*isDigit*(chaine.charAt(i))).
- Si c'est un 0, le char suivant ne doit pas être un caractère, si oui **if**(i + 1 >= chaine.length() || !Character.*isDigit*(chaine.charAt(i+1))) , print un \n et on incrémente i. si non le programme s'arrête car c'est une erreur.
- Ensuite on stocke le nombre, qui va nous servir de gardien de boucle, vu qu'on va devoir répéter tant de fois le caractère, cmpt = Character.*getNumericValue*(chaine.charAt(i)); et on incrémente i.
- On print tant de fois le caractère.

4. CODAGE = PROGRAMMER ET VALIDER LA SOLUTION EN JAVA