

Tableau à capacité variable



Objectifs

- Présenter la structure de base des tableaux à capacité variable (TCV)
- Notre implantation
- Opérations

Tableau à capacité variable

- En gros, c'est la liste « *List<TypeElement>* » de C# !
- La collection est stockée dans un tableau
- Ce tableau a une capacité fixe, mais un nombre d'éléments variable
- Alors on ajoute un attribut qui permet de connaître le nombre d'éléments dans la collection
 - Le nombre d'éléments est inférieur ou égale à la capacité du tableau
 - Quand le tableau est plein, on crée un nouveau tableau du double de la capacité actuelle (si 0, on la met à 1)

Structure

Minimalemment la structure contient :

- NombreElements : entier ; contient le nombre d'éléments contenus dans la collection
- Capacité : entier ; capacité maximale de la collection
- Donnees : tableau de « *TypeElement* » ; contient l'ensemble des données

Légendes

 Variable locale non identifiée

Nom variable

 Variable locale identifiée

@0

123 @1 Objet avec données

@0
23 13 Tableau référence 0

→ Visualisation référence

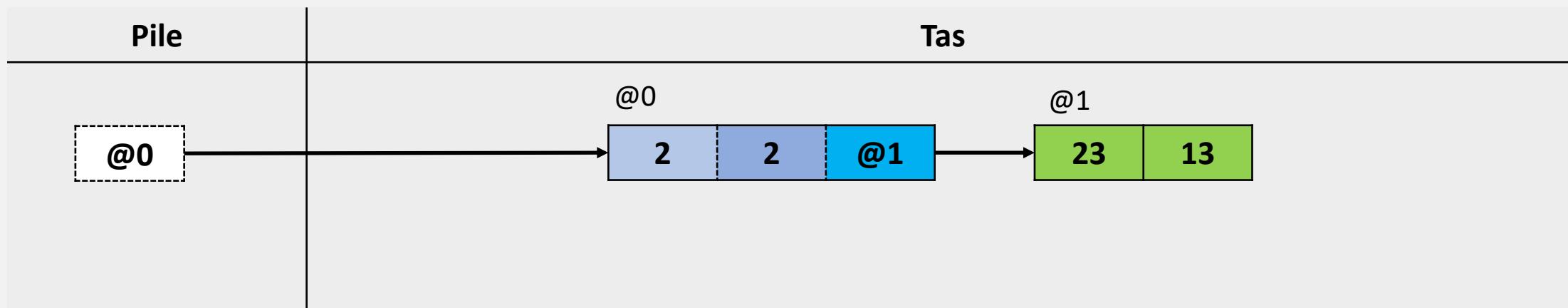
v Case utilisée

? Case non utilisée

n Nombre d'éléments

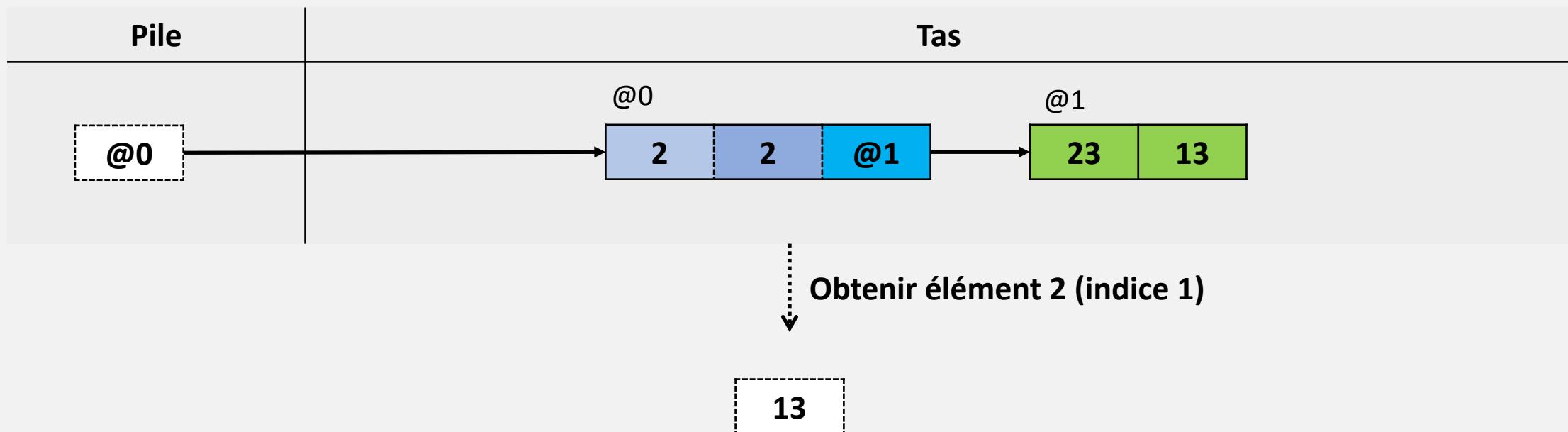
c Capacité

@ref Données



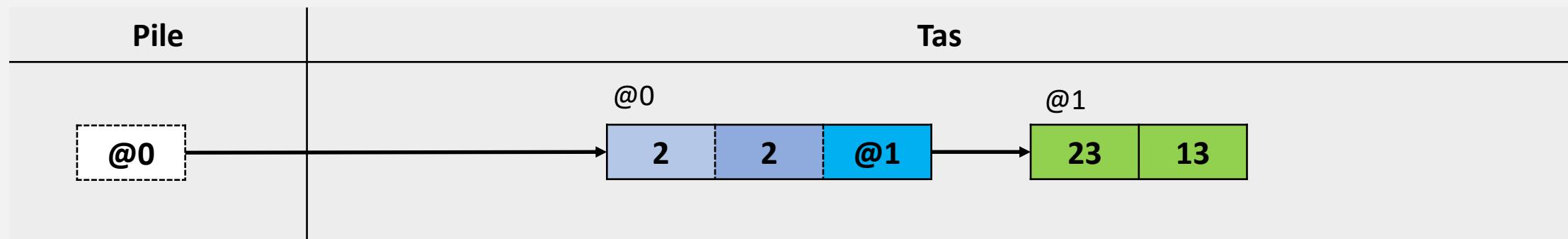
Opération – Accès à un élément – Lecture

- Idée : à partir de la structure, utiliser le tableau de données pour renvoyer l'élément demandé



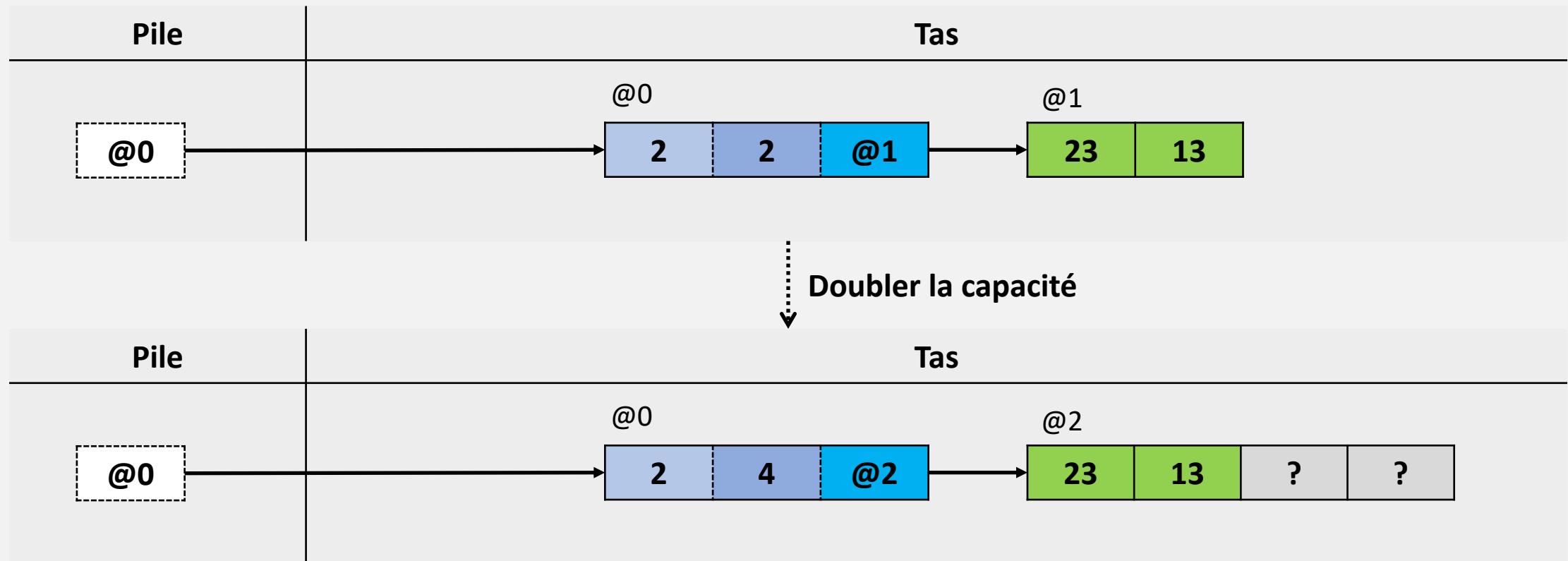
Opération – Accès à un élément – Écriture

- Idée : à partir de la structure, utiliser le tableau de données pour affecter la nouvelle valeur demandée



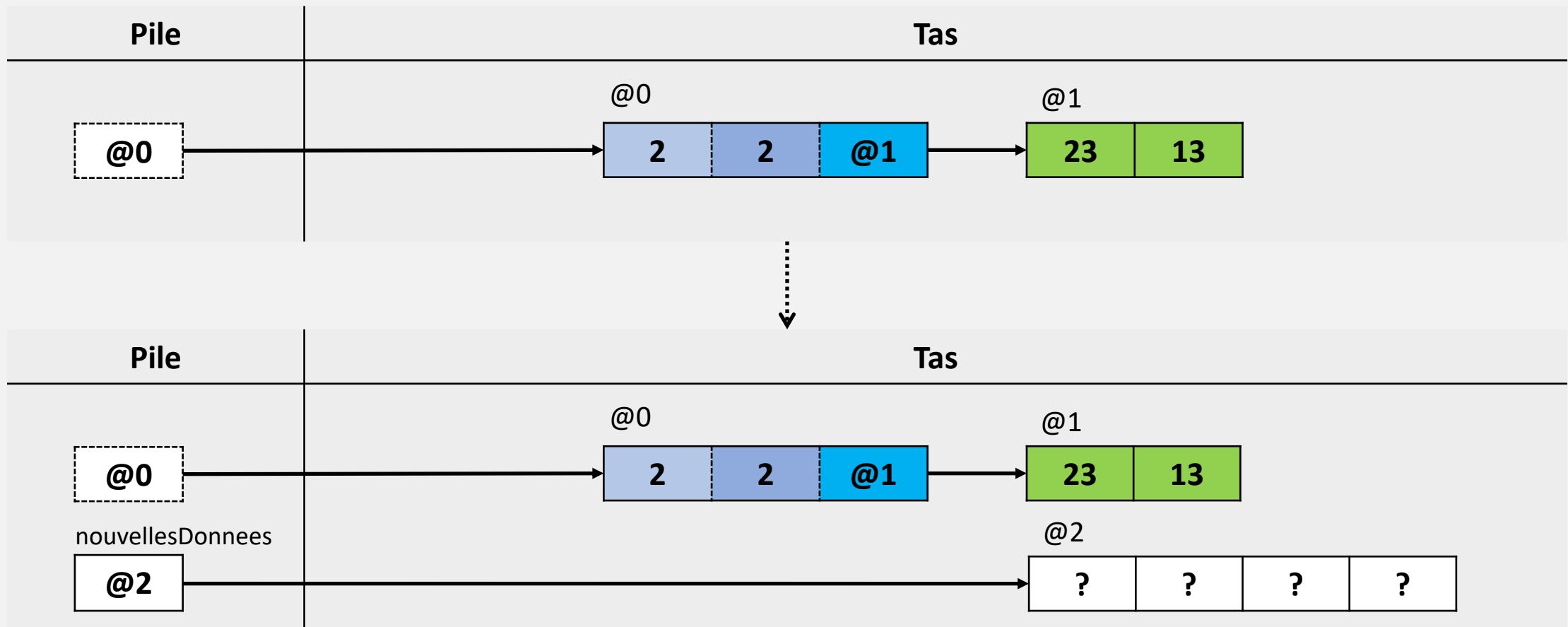
Opération – Augmenter la capacité

- Idée



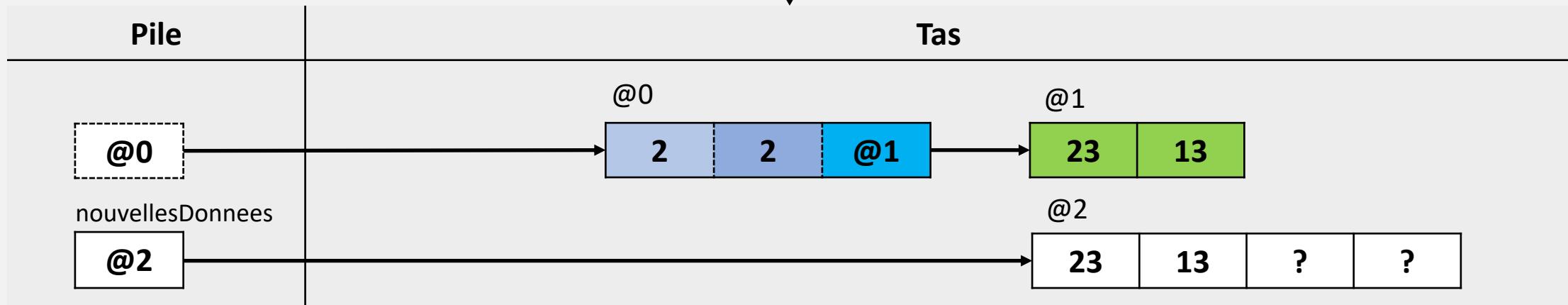
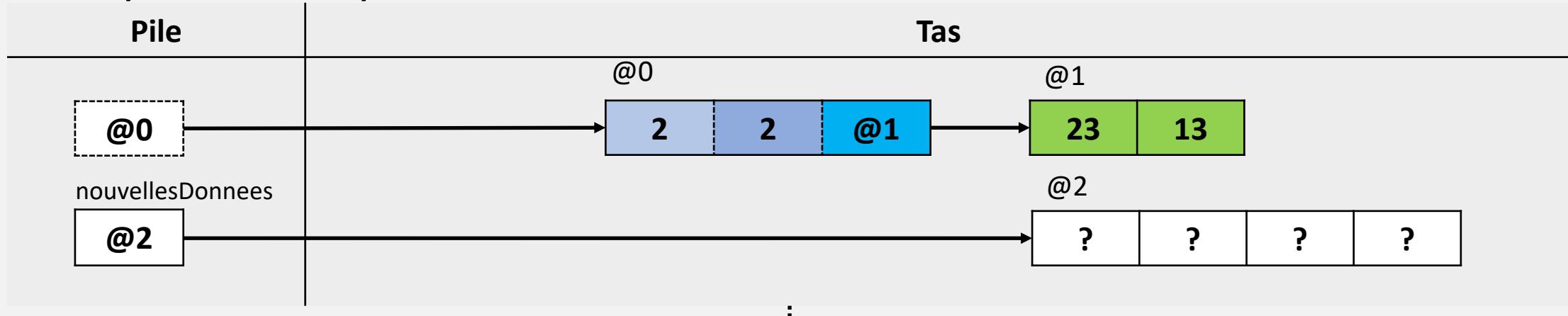
Opération – Augmenter la capacité

- Étape 1 : Créer un nouveau tableau



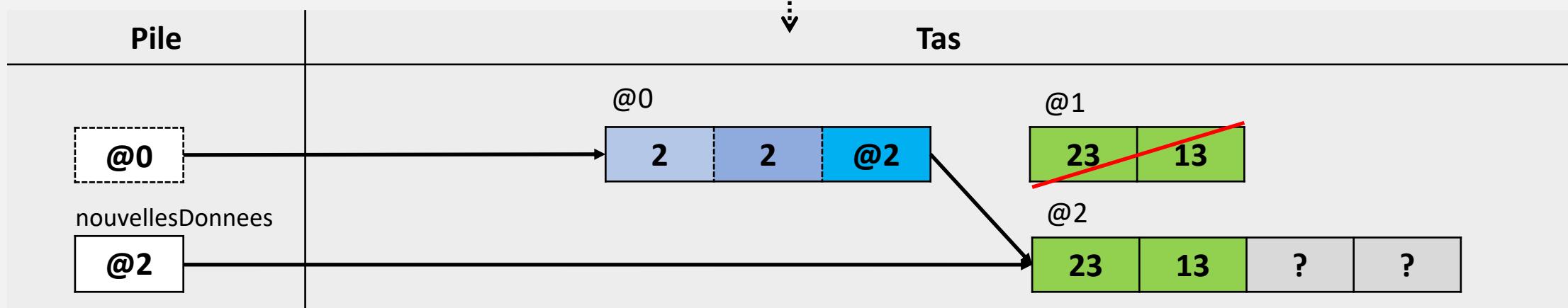
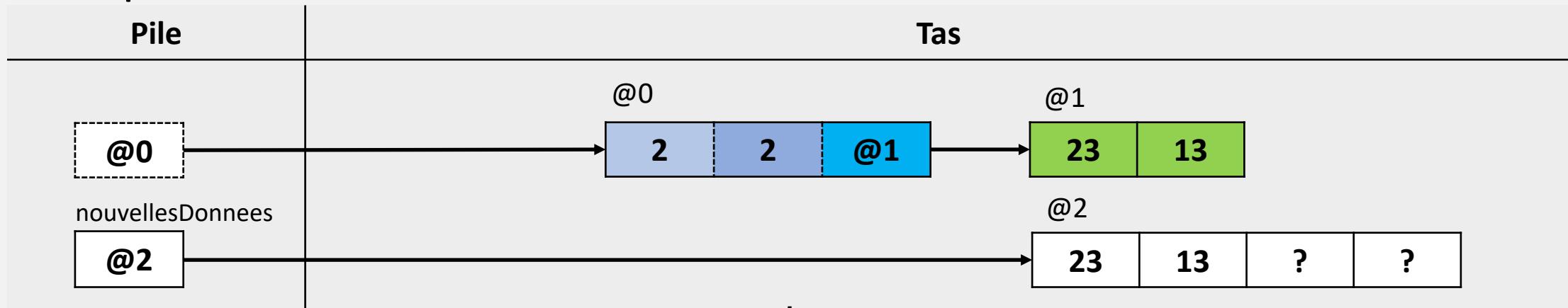
Opération – Augmenter la capacité

- Étape 2 : Recopier les valeurs actuelles



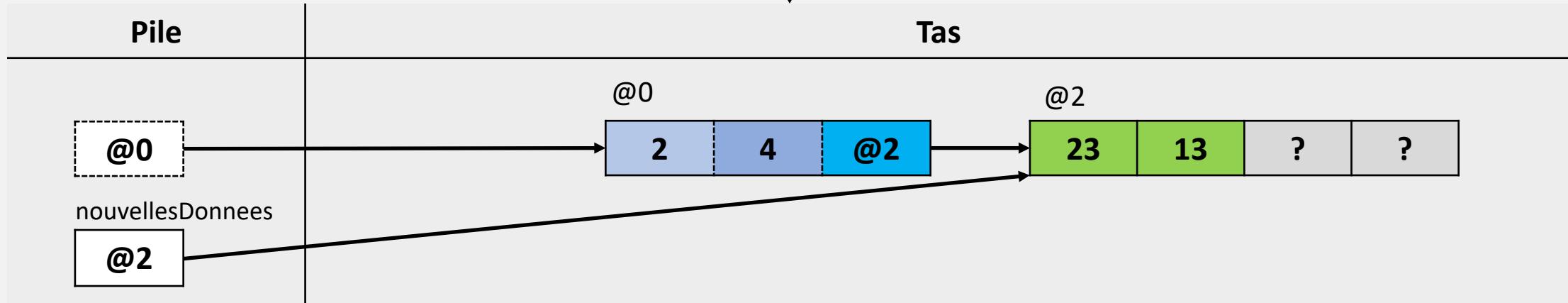
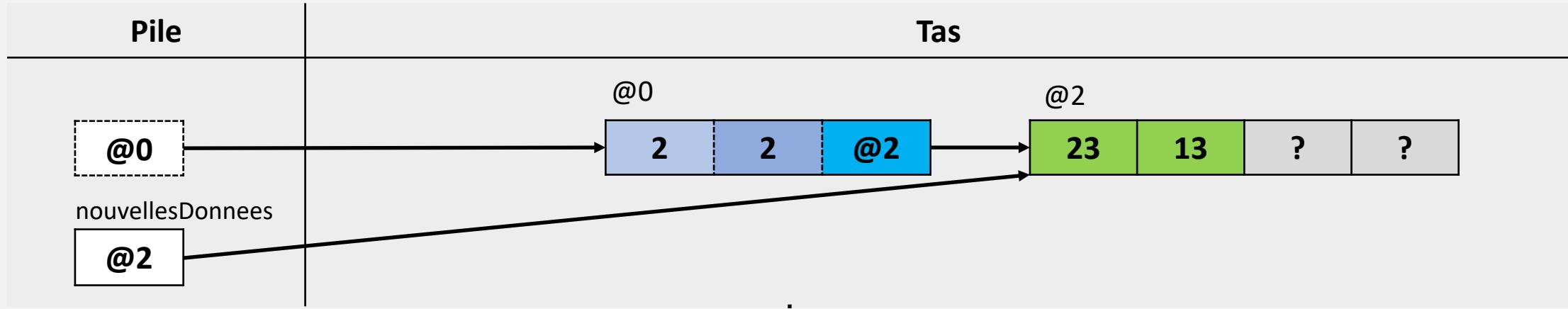
Opération – Augmenter la capacité

- Étape 3 : Modifier la référence du tableau de données



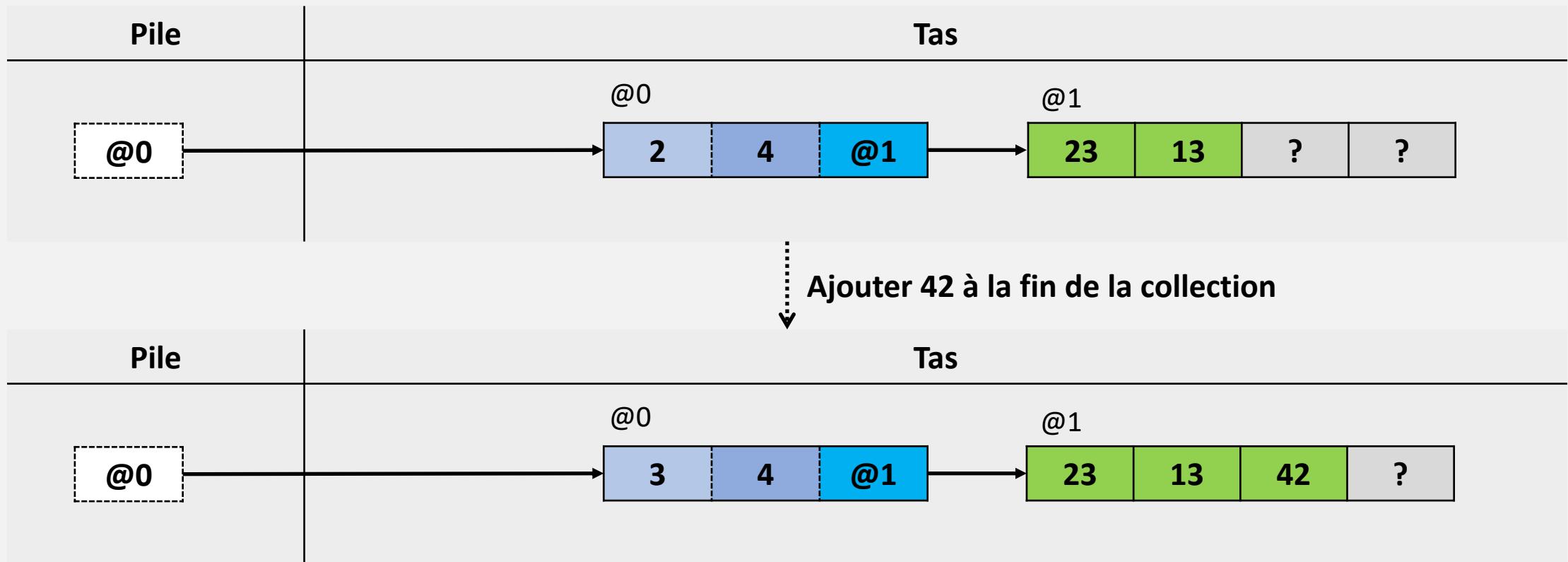
Opération – Augmenter la capacité

- Étape 4 : Modifier la capacité (Optionnel dans notre implantation)



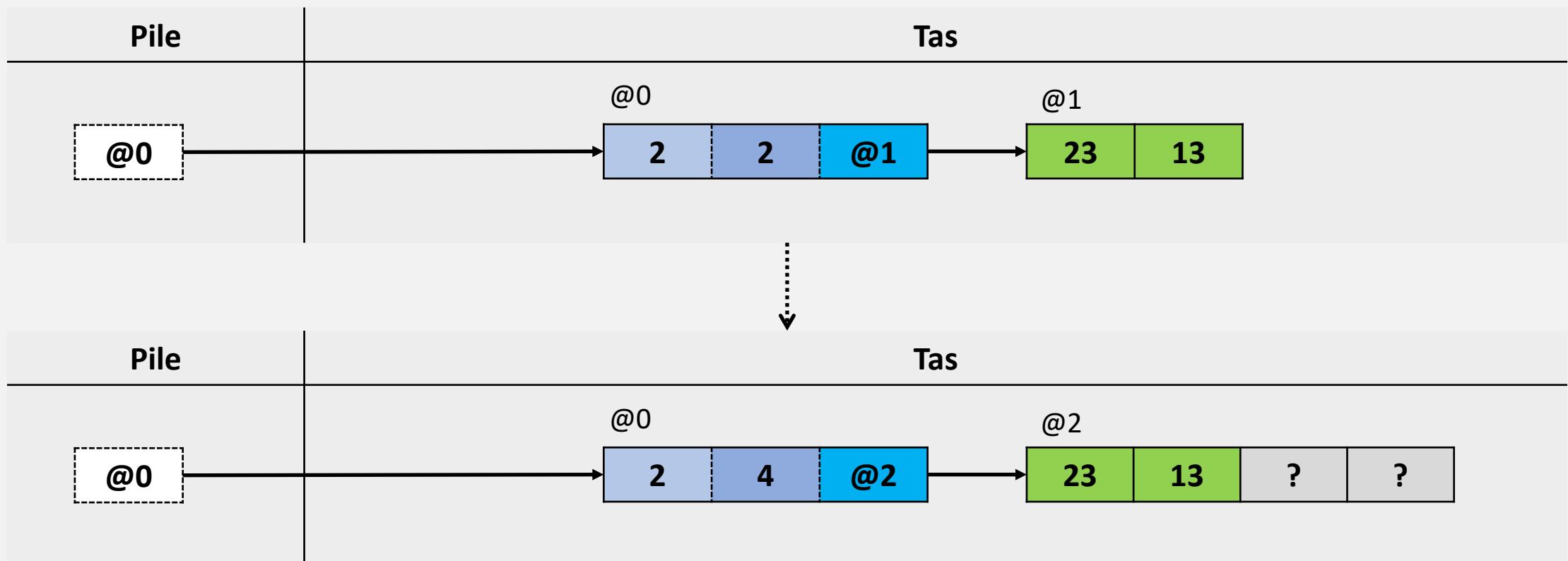
Opération – Ajouter en fin

- Idée



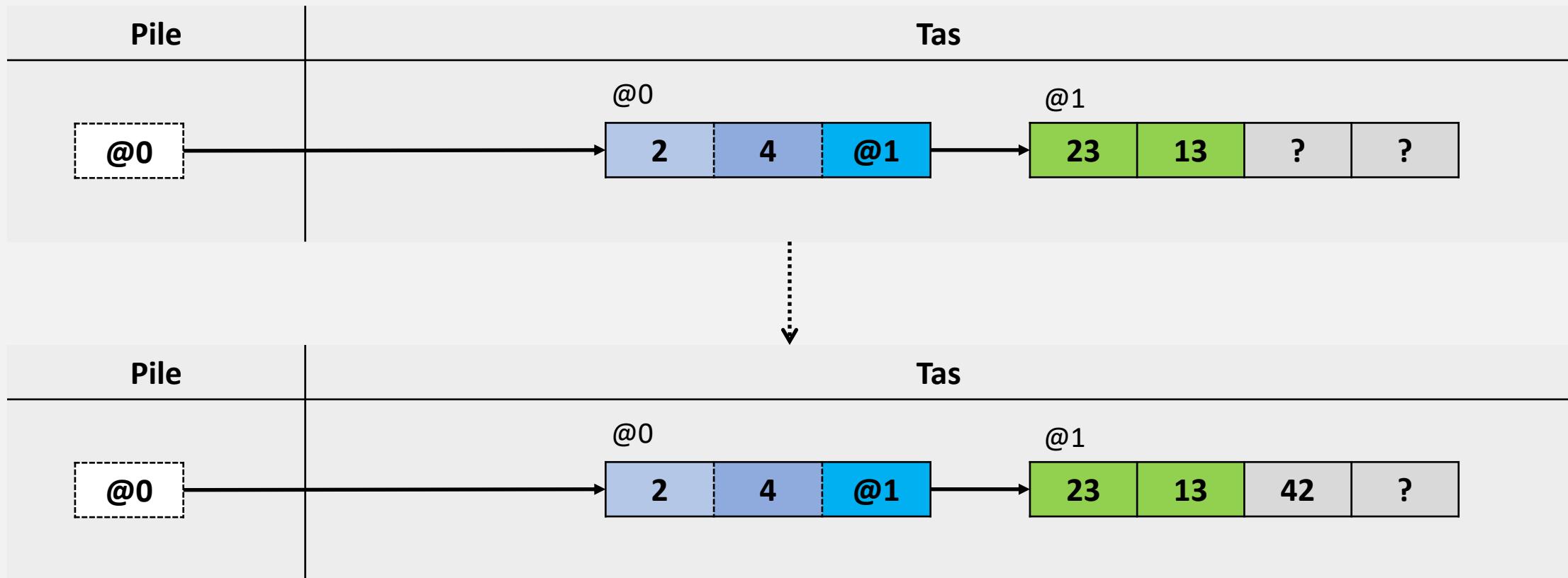
Opération – Ajouter en fin

- Étape 0 : Si capacité atteinte, appliquer l'opération augmenter la capacité par deux



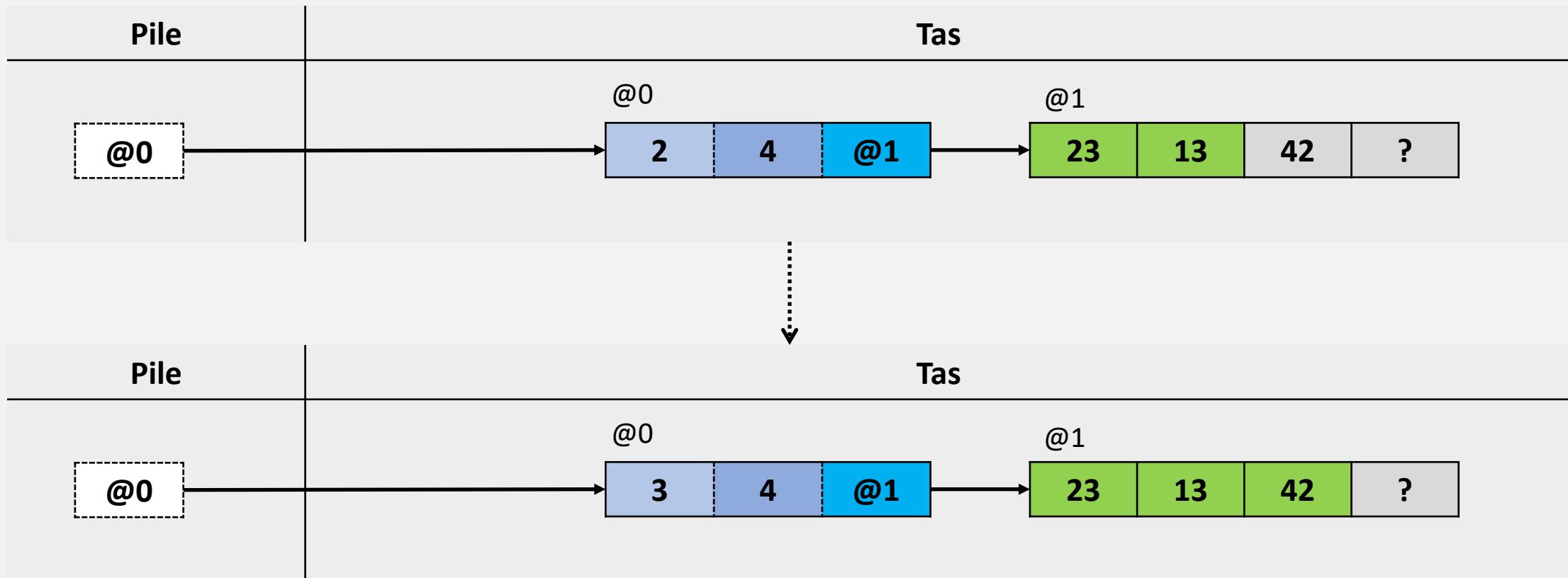
Opération – Ajouter en fin

- Étape 1 : Mettre la valeur à la position « *NombreElements* »



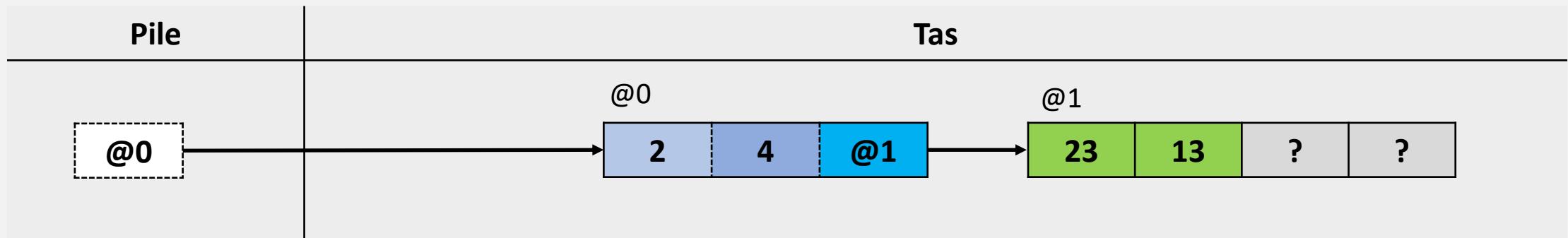
Opération – Ajouter en fin

- Étape 3 : Mettre le nombre d'éléments

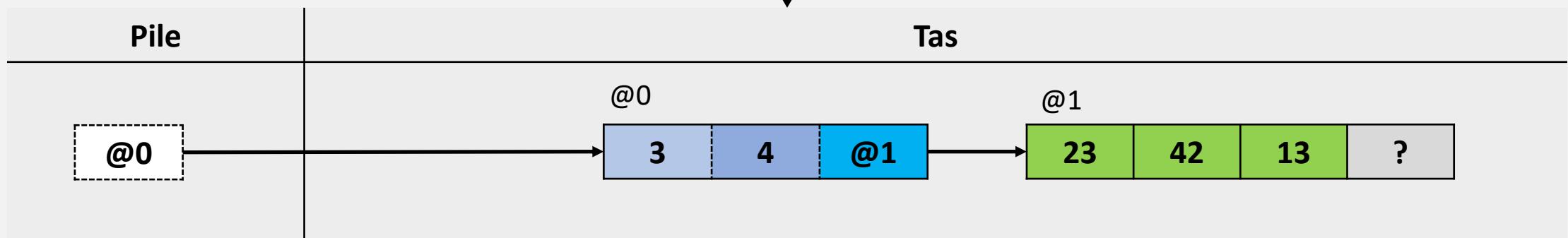


Opération – Insérer

- Idée

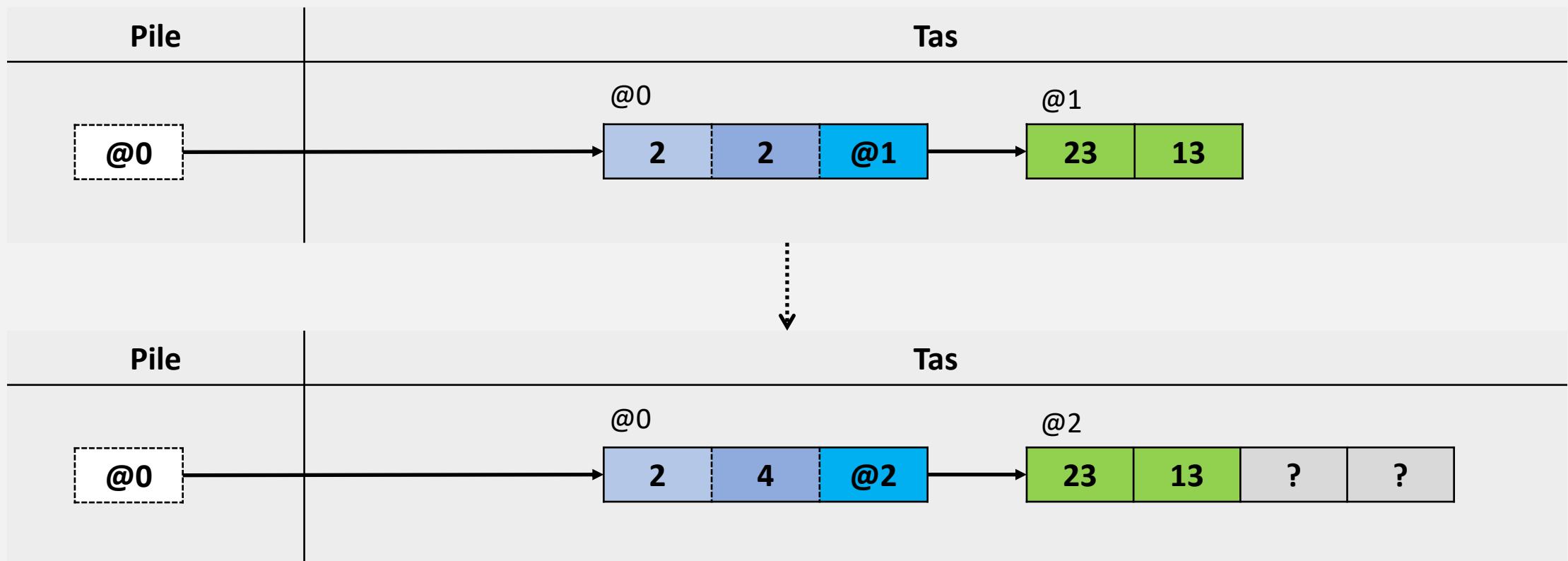


Insérer 42 à la deuxième position (indice 1)



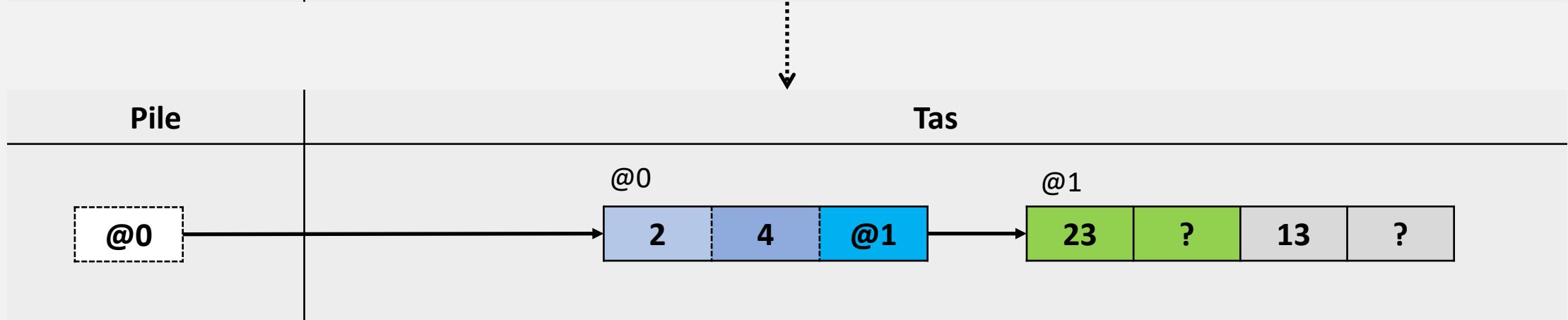
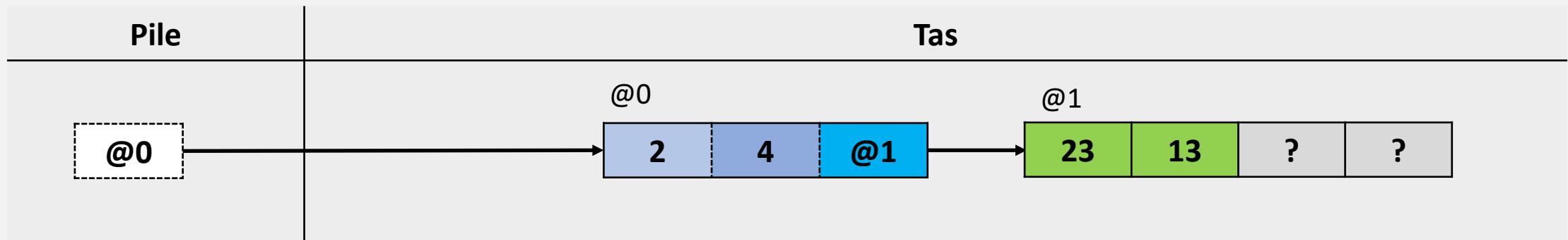
Opération – Insérer

- Étape 0 : Si capacité atteinte, appliquer l'opération augmenter la capacité par deux



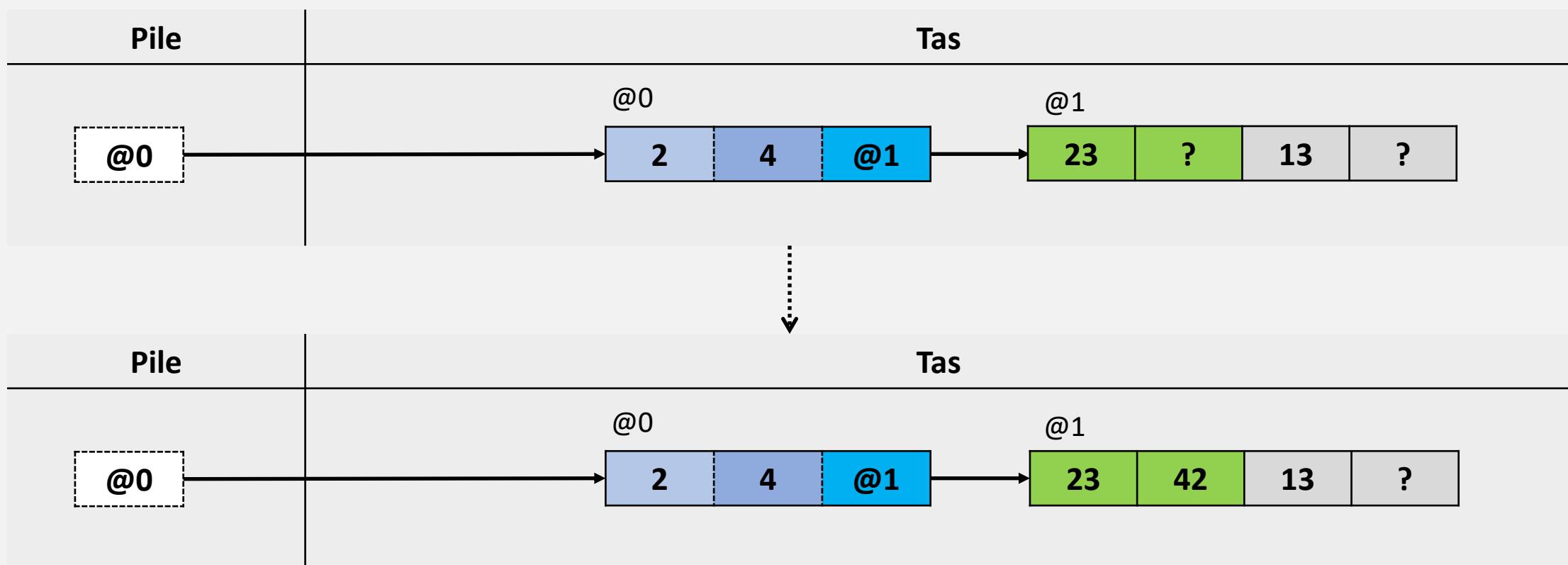
Opération – Insérer

- Étape 1 : Décaler les valeurs présentent de la position voulu au nombre d'éléments vers la droite



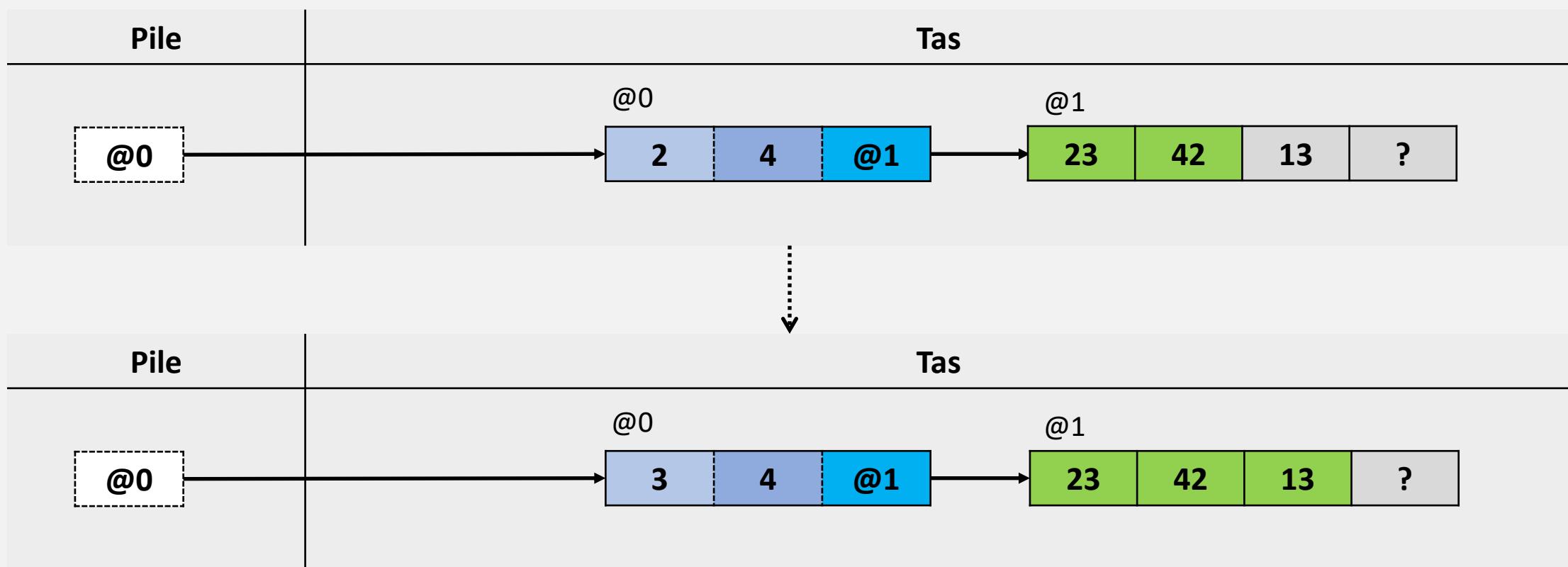
Opération – Insérer

- Étape 2 : Affecter la valeur



Opération – Insérer

- Étape 3 : Mettre à jour le nombre d'éléments



Remarque sur l'implantation C#

- La capacité peut facilement être remplacée par une propriété qui :
 - Get : renvoie la capacité du tableau de données
 - Set : modifie la capacité du tableau de données
- Dans les exemples, on utilise des entiers comme données. Vous devez utiliser les génériques
- Au niveau utilisateur, il est intéressant d'implanter les interfaces :
 - `IList<TypeElement>` : <https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.ilist-1>
 - `IEnumerable<TypeElement>` : <https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.ienumerable-1>
 - `IEnumerator<TypeElement>` : <https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.ienumerator-1>