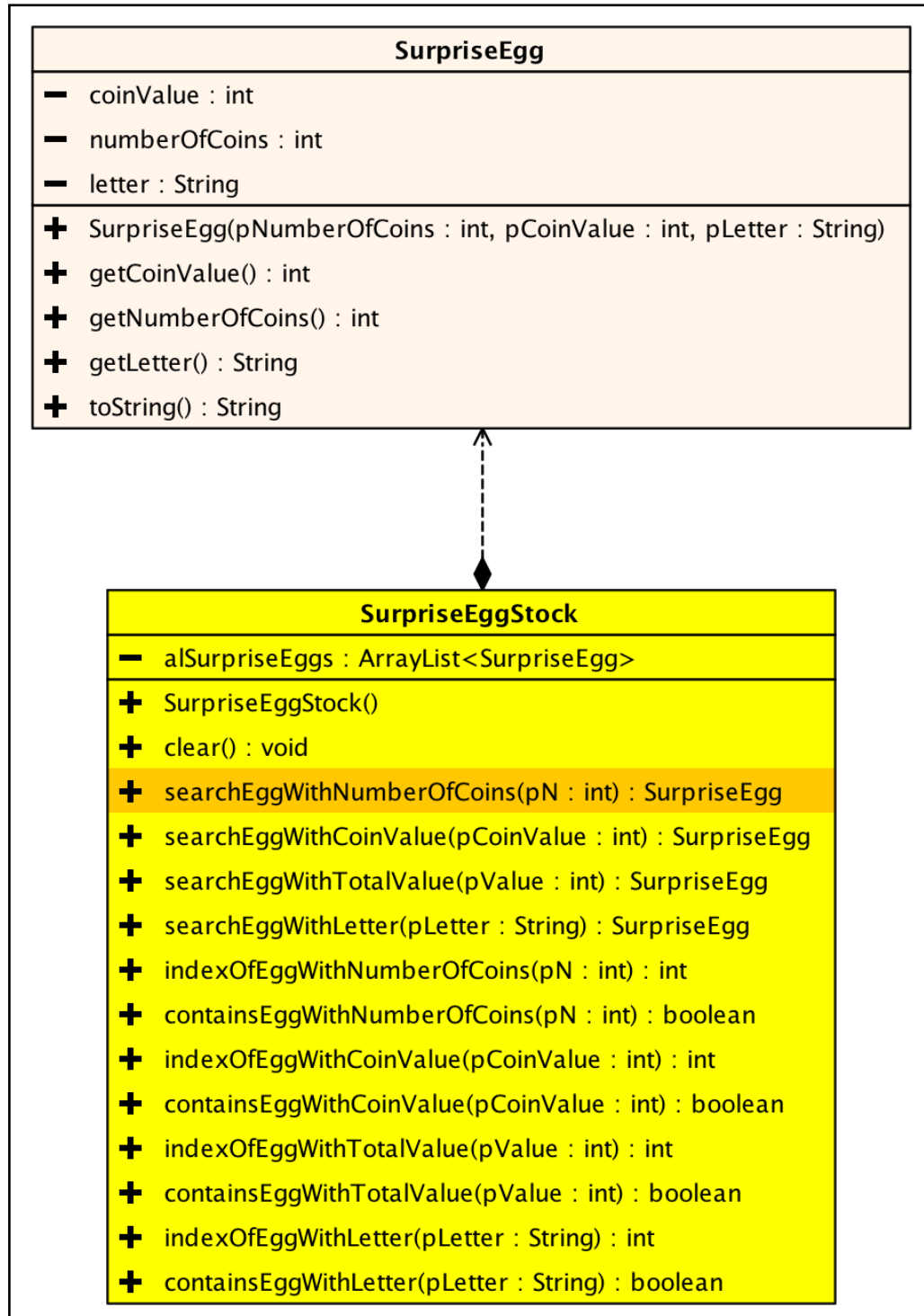


# Recherche linéaire

On a une liste d'œufs surprises qui sont marqués par des lettres et contiennent des pièces de monnaie. D'un œuf à l'autre les pièces de monnaie diffèrent en valeur et en nombre.



## Code Java

```
public SurpriseEgg searchEggWithNumberOfCoins(int pN){

    SurpriseEgg result = null;
    int i = 0;

    while(result == null && i < alSurpriseEggs.size()){
        SurpriseEgg egg = alSurpriseEggs.get(i);
        if (egg.getNumberOfCoins() == pN){
            result = egg;
        }else{
            i++;
        }
    }

    return result;
}
```

Au début on n'a pas encore trouvé d'œuf correspondant au critère de recherche. Pour trouver un œuf contenant pN pièces on vérifie le contenu de chaque œuf en commençant à l'index 0. Tant qu'on n'a pas de résultat et qu'il reste des œufs à vérifier, on accède à l'œuf à l'index actuel et on vérifie s'il contient pN pièces. Si c'est le cas l'œuf actuel constitue le résultat. Sinon il faut procéder avec l'index suivant. Finalement on retourne le résultat.

## Exemple d'exécution

Par la suite le déroulement de l'algorithme est détaillé étape par étape à l'aide d'un exemple concret dans lequel on désire trouver le premier œuf contenant 3 pièces de monnaie.

Soit la liste suivante d'œufs contenant les nombres de pièces de monnaie indiqués :

Index	0	1	2	3	4	5
Nombre de pièces dans o	4	1	5	3	3	10

pN	3
----	---

**i==0** Œufs déjà considérés par la boucle, ne correspondant pas au critère de recherche

Index	0	1	2	3	4	5
Nombre de pièces dans o	4	1	5	3	3	10

pN	3
----	---

**i==1**

Index	0	1	2	3	4	5
Nombre de pièces dans o	4	1	5	3	3	10

pN	3
----	---

i==2

Index	0	1	2	3	4	5
Nombre de pièces dans o	4	1	5	3	3	10

pN	3
----	---

i==3 **Œuf correspondant au critère de recherche**

Index	0	1	2	3	4	5
Nombre de pièces dans o	4	1	5	3	3	10

pN	3
----	---

A l'index 3 on trouve le premier œuf correspondant au critère de recherche. L'exécution de la boucle est donc terminée et on retourne l'œuf à l'index 3.

## Remarques

1. Pour trouver un œuf correspondant à un autre critère de recherche, il faut appeler un autre accesseur **dans la partie bleue** (voir page 2). Si par exemple on veut trouver un œuf contenant des pièces de monnaie d'une valeur donnée, il faut appeler l'accesseur `getCoinValue`.



---

*Pour vérifier l'égalité de deux chaînes de caractères il faut utiliser la méthode `equals` (cf. solution de la méthode `serachEggWithLetter` de l'Exercice M1)*

---

2. Parfois on désire trouver l'index d'un œuf qui correspond à un certain critère de recherche. Le nom de la méthode commence alors généralement par `indexOf`. Par exemple la méthode pour trouver le premier œuf contenant un nombre donné de pièces s'appelle `indexOfEggWithNumberOfCoins`. La structure de l'algorithme reste la même, mais la variable `result` est un entier, initialisé avec la valeur `-1` pour indiquer qu'on n'a pas encore trouvé de résultat. Par conséquent il faut aussi adapter **la première condition de la boucle**. En plus **il faut stocker l'index actuel `i` dans la variable `result` si le critère de recherche est satisfait**.
3. Dans d'autres cas on veut simplement savoir si la liste contient un œuf correspondant à un critère de recherche donné. Le nom de la méthode commence alors généralement par `contains`. Par exemple la méthode pour vérifier si la liste contient un œuf avec un nombre donné de pièces s'appelle `containsEggWithNumberOfCoins`. A nouveau l'algorithme reste le même, mais la variable `result` est un booléen, initialisé avec la valeur `false` pour indiquer qu'on n'a pas encore trouvé de résultat. Par conséquent il faut aussi adapter **la première condition de la boucle**. En plus **il faut stocker la valeur `true` dans la variable `result` si le critère de recherche est satisfait**.

---

*Analysez la solution de l'Exercice M1 et surtout les différences entre les méthodes.*

---

