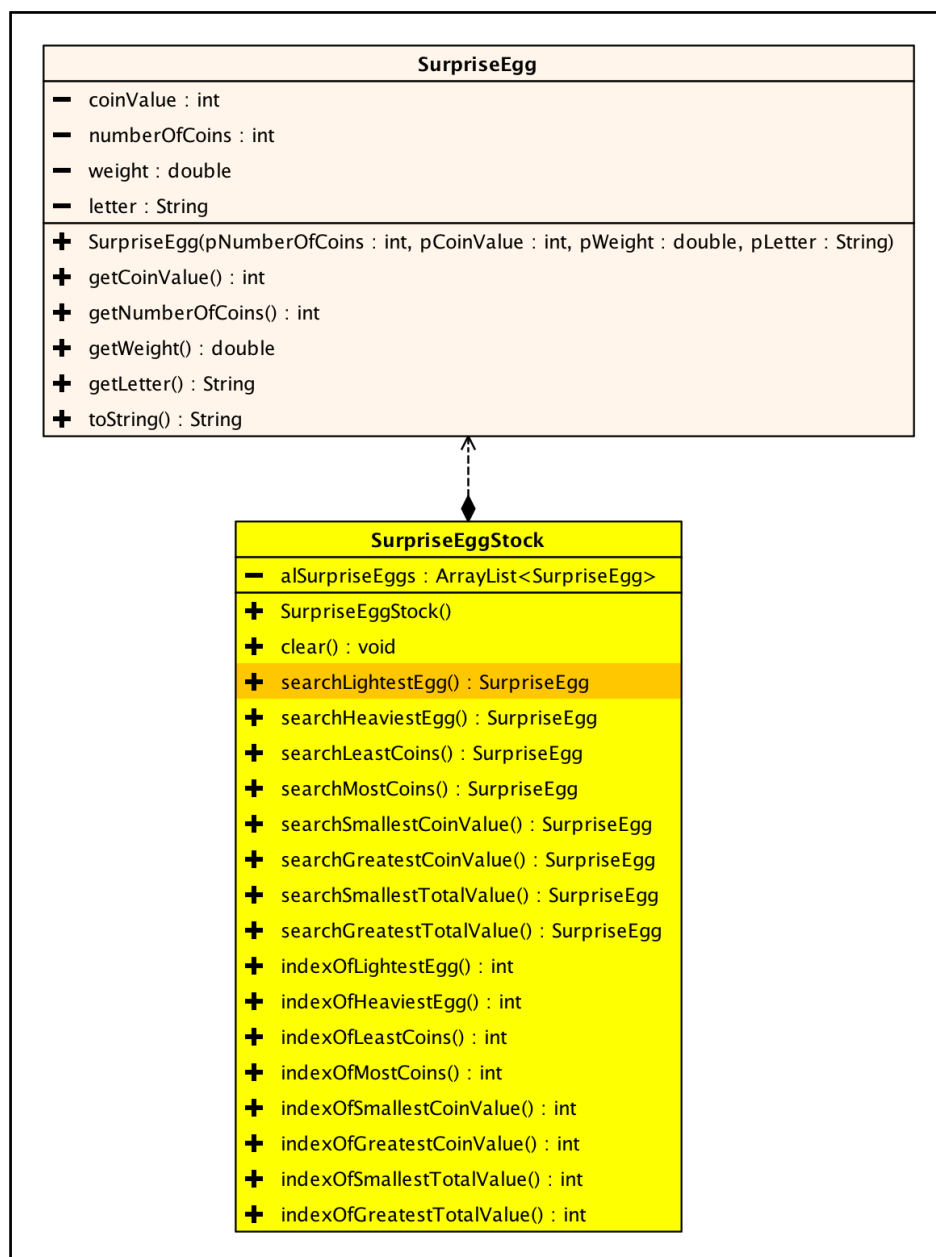
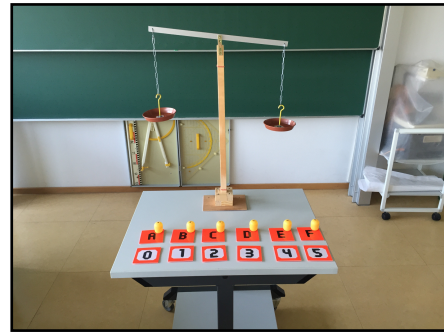


Recherche du minimum

On a une liste (nommée `alSurpriseEggs` dans les exemples qui suivent) avec des œufs surprises (classe `SurpriseEgg`) de poids différents.

Comment peut-on procéder pour déterminer l'œuf le plus léger de la liste `alSurpriseEggs` si on peut uniquement se servir d'une balance qui permet de comparer le poids de deux œufs ? Cette balance n'indique pas le poids en soi.



Code Java

```
public SurpriseEgg searchLightestEgg() {
    if (alSurpriseEggs.isEmpty()) {
        return null;
    }

    SurpriseEgg eggMin = alSurpriseEggs.get(0);

    for (int i = 1; i < alSurpriseEggs.size(); i++) {
        SurpriseEgg egg = alSurpriseEggs.get(i);

        if (egg.getWeight() < eggMin.getWeight()) {
            eggMin = egg;
        }
    }

    return eggMin;
}
```

1. Dans une liste vide il n'y a pas d'œufs. On ne retourne donc aucun œuf le cas échéant. Il faut traiter ce cas séparément pour éviter que l'initialisation de `eggMin` ne produise un message d'erreur.
2. On compare tout œuf de la liste à l'œuf actuellement le plus léger, remplaçant l'œuf le plus léger si on trouve un œuf encore plus léger (voir exemple d'exécution ci-après). Après avoir fait cette comparaison pour l'œuf à chaque index, on est sûr qu'`eggMin` contient l'œuf le plus léger qu'on retourne simplement.

Exemple d'exécution

Soit la liste suivante d'œufs avec les poids indiqués :

Index	0	1	2	3	4	5	6	7
Poids de egg	10	8	13	3	1	15	1	8

Par la suite le déroulement de l'algorithme est détaillé étape par étape.

Initialisation de `eggMin` devant la boucle

Au début on suppose simplement que l'œuf le plus léger est le premier œuf de la liste. C'est parce qu'il est important d'initialiser la variable `eggMin` avant la première comparaison de la boucle.

Les œufs déjà considérés ont un arrière-plan vert.

Index	0	1	2	3	4	5	6	7
Poids de egg	10	8	13	3	1	15	1	8
Poids de eggMin	10							

Déroulement de la boucle

Dans la boucle on compare chacun des œufs avec l'œuf le plus léger actuel `eggMin`. Si on constate que l'œuf `egg` à l'index actuel est plus léger que `eggMin`, `egg` devient `eggMin`.

`i==1`

Index i	0	1	2	3	4	5	6	7
Poids de egg	10	8	13	3	1	15	1	8
Poids de eggMin	8							

`i==2`

Index i	0	1	2	3	4	5	6	7
Poids de egg	10	8	13	3	1	15	1	8
Poids de eggMin	8							

`i==3`

Index i	0	1	2	3	4	5	6	7
Poids de egg	10	8	13	3	1	15	1	8
Poids de eggMin	3							

`i==4`

Index i	0	1	2	3	4	5	6	7
Poids de egg	10	8	13	3	1	15	1	8
Poids de eggMin	1							

`i==5`

Index i	0	1	2	3	4	5	6	7
Poids de egg	10	8	13	3	1	15	1	8
Poids de eggMin	1							

`i==6`

Index i	0	1	2	3	4	5	6	7
Poids de egg	10	8	13	3	1	15	1	8
Poids de eggMin	1							

`i==7`

Index i	0	1	2	3	4	5	6	7
Poids de egg	10	8	13	3	1	15	1	8
Poids de eggMin	1							

Après avoir considéré tous les œufs, `eggMin` correspond à l'œuf le plus léger qu'on peut simplement retourner.

Index du minimum

Souvent on ne désire pas retourner le minimum lui-même, mais plutôt son index.

Le code Java suivant montre comment on peut retourner l'index de l'œuf le plus léger de notre liste `alSurpriseEggs`.

```
public int indexOfLightestEgg() {  
    if (alSurpriseEggs.isEmpty()) {  
        return -1;  
    }  
  
    SurpriseEgg eggMin = alSurpriseEggs.get(0);  
    int iMin = 0;  
  
    for (int i = 1; i < alSurpriseEggs.size(); i++) {  
        SurpriseEgg egg = alSurpriseEggs.get(i);  
  
        if (egg.getWeight() < eggMin.getWeight()) {  
            eggMin = egg;  
            iMin = i;  
        }  
    }  
  
    return iMin;  
}
```

L'algorithme de la recherche de l'index du minimum est le même que celui de la recherche du minimum à quelques détails près qui sont marqués en couleur :

- Lorsque la liste est vide on retourne un index duquel on est sûr qu'il n'existe jamais, pour aucune liste. Ceci nous permet de distinguer le cas d'une liste vide des autres cas. Toute valeur négative satisfait cette condition. On choisit cependant -1, comme c'est une convention respectée par de nombreux développeurs.
- Cette fois-ci il faut aussi mémoriser l'index de l'œuf le plus léger. Initialement cet index vaut 0. Après chaque changement de `eggMin` il faut aussi changer l'index de l'œuf le plus léger `iMin`. Finalement l'index de l'œuf le plus léger est retourné.