



POA - Labo 2

Squadrons

Nicolas Crausaz & Maxime Scharwath

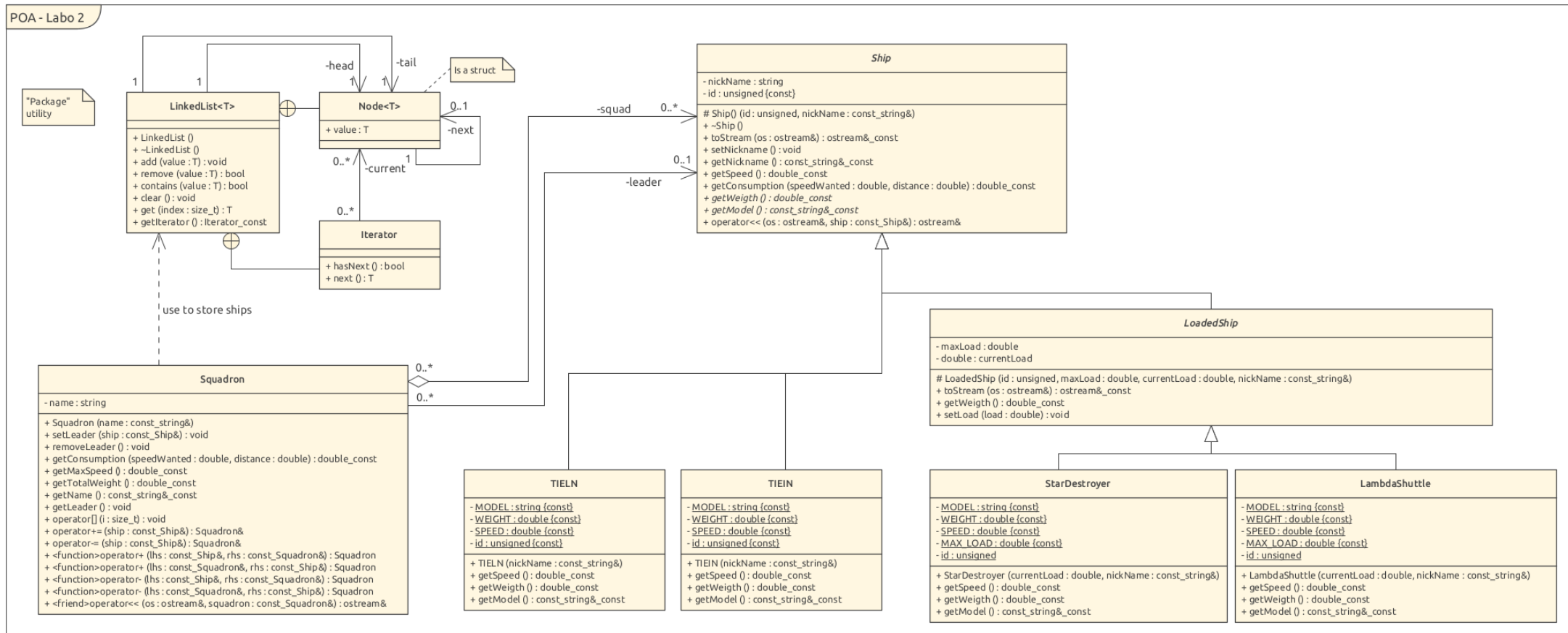
13.04.2022

Table des matières

1. Conception	3
1.1. Diagramme	3
2. Implémentation	4
2.1. Choix	4
3. Tests	5

1. Conception

1.1. Diagramme



2. Implémentation

2.1. Choix

Vaisseaux

Nous avons implémenté les vaisseaux en distinguant deux types principaux, les vaisseaux (*Ship*) et les vaisseaux avec chargement (*LoadedShip*).

Nous avons dû faire un choix de conception au niveau de la classe *LoadedShip*, en effet nous avions pour idée de définir une méthode *getMaxLoad()* dans les sous vaisseaux, qui renverrait la valeur de l'attribut statique *MAX_LOAD*. Mais lorsqu'on souhaitait appeler cette méthode dans le constructeur de *LoadedShip* pour valider *currentLoad*, nous rencontrions une segmentation fault (probablement dû à l'appel d'une méthode virtuelle dans un constructeur). Nous avons donc contourné le problème en passant le chargement maximum dans le constructeur de *LoadedShip*.

Squadron / LinkedList

Pour stocker les vaisseaux dans les squadrons, nous avons implémenté notre propre version d'une liste simplement chaînée. Cela nous permet d'ajouter / supprimer efficacement des vaisseaux par rapport à l'utilisation d'un tableau standard dont on devrait allouer plus d'espace mémoire lors d'ajouts. Notre implémentation de cette LinkedList reprend la plupart des concepts de la librairie standard et propose des méthodes proposant: ajout, suppression, vérification d'appartenance, vider, accès et itérateurs.

Dans cette liste nous stockons une tête et une queue, cela nous permet d'insérer en fin de liste en temps constant. Nous avons également proposé un itérateur sur cette liste.

3. Tests

Les numéros des tests sont référencés dans le fichier du processus de test (test/test.cpp).

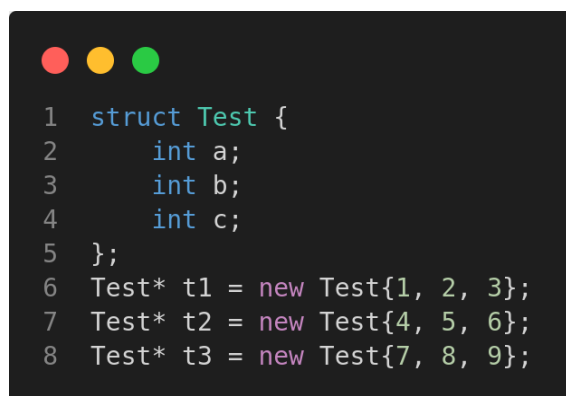
Nous avons testé notre implémentation de liste chaînée avec plusieurs type de données (primitif, pointeurs, objets (références constantes))

LinkedList<int>

No	Description	Méthodes	Ok
1	Ajouter et récupérer des éléments	add(item), get(index)	Ok
2	Supprimer un élément	remove(item)	Ok
3	Vider la liste	clear()	Ok
4	Vérifier si un élément est dans la liste	contains(item)	Ok

LinkedList<Test*>

Cette série de test utilise la structure suivante:



```

1 struct Test {
2     int a;
3     int b;
4     int c;
5 };
6 Test* t1 = new Test{1, 2, 3};
7 Test* t2 = new Test{4, 5, 6};
8 Test* t3 = new Test{7, 8, 9};

```

No	Description	Méthodes	Ok ?
1	Ajouter et récupérer des éléments	add(item), get(index)	Ok
2	Supprimer un élément	remove(item)	Ok
3	Vider la liste	clear()	Ok
4	Vérifier si un élément est dans la liste	contains(item)	Ok

LinkedList<const string&>

No	Description	Méthodes	Ok ?
1	Ajouter et récupérer des éléments	add(item), get(index)	Ok
2	Supprimer un élément	remove(item)	Ok
3	Vider la liste	clear()	Ok
4	Vérifier si un élément est dans la liste	contains(item)	Ok

LinkedList::Iterator

No	Description	Comportement attendu	Ok ?
1	Récupérer	Itérateur sur le premier élément	Ok
2	A un suivant	Vrai	Ok
3	Itère vers le suivant	Itérateur sur élément suivant	Ok
4	N'a pas de suivant	faux	Ok

Ships

No	Description	Comportement attendu	Ok ?
1	Construction d'un LambdaShuttle avec chargement > que le max autorisé	std::runtime_exception	Ok
2	Construction d'un StarDestroyer avec chargement > que le max autorisé	std::runtime_exception	Ok
3	Récupérer le nom (non défini)	""	Ok
4	Changer le nom	setNickname("Test")	Ok
3, 4	Récupérer le nom	"Test"	Ok
5	Récupérer la vitesse	TIELN -> 100 TIEIN -> 110 LambdaShuttle -> 54 StarDestroyer -> 40	Ok
6	Récupérer le poids (sans chargement)	TIELN -> 6 TIEIN -> 5 LambdaShuttle -> 360 StarDestroyer -> 9000000000	Ok
7	Récupérer le poids (avec chargement)	LambdaShuttle -> 360 + load	Ok

		StarDestroyer -> 90000000000 + load	
8	Afficher un vaisseau	Affichage selon type	Ok

Squadron

No	Description	Comportement attendu	Ok ?
1	Création	Instance de squadron	Ok
2	Ajout du leader	Deviens leader + ajout vaisseau	Ok
3	Ajout du leader (déjà présent)	Deviens leader	Ok
4	Suppression du leader	Plus de leader, vaisseau toujours dans le squadron	Ok
5	Ajout vaisseau (+=)	Vaisseau ajouté	Ok
6	Ajout vaisseau (+)	Vaisseau ajouté	Ok
7	Suppression vaisseau (-=)	Vaisseau retiré	Ok
8	Suppression vaisseau (-)	Vaisseau retiré	Ok
9	Suppression d'un vaisseau non membre	Aucun	Ok
10	Récupérer à index	Récupère le bon vaisseau	Ok
11	Récupérer à index invalide	std::out_of_range	Ok
12	Vitesse max	Vitesse du vaisseau le plus lent	Ok
13	Poids total	Poids vaisseau + chargements	Ok