2019

Compte rendu de TP : Implantation d'une table de hachage, classe interne

Java avancé – TP n°4

Jonathan CRÉTÉ – Filière Informatique

Exercice 1 - IntHashSet

1) Quels doivent être les champs de la classe Entry correspondant à une case de la table de hachage sachant que l'on veut stocker les collisions dans une liste chaînée.

On cherche à écrire la classe Java Entry correspondante. Quelle doit être la visibilité de cette classe ? Quels doivent être les modificateurs pour chaque champ ?

Écrire la classe Entry dans le package fr.umlv.set.

On cherche à stocker les collisions dans une liste chaînée.

la visibilité de la classe doit être une visibilité package car on est on « maître à bord » dans notre package. Les modificateurs des champs doivent être package - final

2) Comment écrire la fonction de hachage dans le cas où la taille de la table est 2? Pourquoi est-ce que ça n'est pas une bonne idée d'utiliser l'opérateur %? Écrire une version "rapide" de la fonction de hachage

Indice: on peut regarder le bit le plus à droite (celui des unités) pour savoir si l'on doit stocker les éléments dans la case 0 ou 1.

Et si la taille de la table est 8?

En suivant la même idée, modifier votre fonction de hachage dans le cas où la taille de la table est une puissance de 2.

Voir la classe IntHashSet.java

En réalisant l'opération %2.

Ce n'est pas une bonne idée car l'opérateur % une opération lente, d'un point de vue CPU (*divisions successives*).

On effectue une opération bit à bit avec le mask &

3) Écrire la classe fr.umlv.set.IntHashSet et implanter sa méthode add.

Note: si vous ne connaissez pas, allez lire le chapitre 2 du Hacker's Delight.

Écrire également la méthode size avec une implantation en O(1).

```
public void add(int valparam) {
	Objects.requireNonNull(valparam);
	var position = hash(valparam);
	var tmp = this.table[position];
	while(tmp != null) {
		if(tmp.value == valparam) {
			return;
		}
		tmp = tmp.next;
	}
```

```
        table[position] = new Entry(valparam, table[position]);
        counter++;
}
```

4) On souhaite que la classe Entry soit maintenant une classe interne de la classe IntHashSet. Que doit-on faire ?

Attention: Penser à bien supprimer le fichier Entry.java contenant l'ancienne classe.

On déplace la classe Entry dans la classe IntHashSet. On est de visibilité package et est static. C'est une inner-class.

5) On cherche maintenant à implanter la méthode forEach. Quelle doit être la signature de la functional interface prise en paramètre de la méthode forEach?

Quel est le nom de la classe du package java.util.function qui a une méthode ayant la même signature ?

Écrire la méthode forEach.

```
public void forEach(IntConsumer consumer) {
        Objects.requireNonNull(consumer);
        for(var elt: table) {
                for(var entry = elt; entry != null; entry = entry.next) {
                                consumer.accept(entry.value);
                }
        }
}
```

La signature est forEach(IntConsumer consumer) {...}

java.util.function.IntConsumer

6) Écrire la méthode contains.

```
public boolean contains(int value) {
        Objects.requireNonNull(value);
        var position= hash(value);
        for(var tmp = table[position]; tmp != null; tmp = tmp.next) {
                if(tmp.value == value) {
                        return true;
                }

        }
        return false;
}
```

Exercice 2 - DynamicHashSet

1) Avant tout, nous souhaitons généreifier notre table de hachage, pour permettre de ne pas stocker uniquement des entiers mais n'importe quel type de valeur.

Avant de généreifier votre code, quelle est le problème avec la création de tableau ayant pour type un type paramétré ?

Comment fait-on pour résoudre le problème, même si cela lève un warning.

Rappeler pourquoi on a ce warning.

Peut-on supprimer le warning ici, ou est-ce une bêtise?

Comment fait-on pour supprimer le warning?

Reprendre et généreifier le code de l'exercice précédent pour fabriquer une classe de table de hachage générique.

Le problème avec la création d'un hashSet ayant pour type un type paramétré, est que l'on doit stocker nos Objects dans un type Array.

Toutefois, on ne peut pas créer de hashSet de E (par exemple), car le type paramétré n'existe plus à l'exécution. On peut donc créer un hashSet d'Objets et le caster. On a ce warning car on cherche à caster le type Object. On peut supprimer ce warning dans la mesure ou on est sûr que ce cast ne va pas planter.

Pour cela, on utilise l'annotation **@SuppressWarnings("unchecked")**

2) Vérifier la signature de la méthode contains de HashSet et expliquer pourquoi on utilise un type plus général que E.

On utilise comme type plus général, le type de base Object :

- E n'existe plus à l'exécution (La VM ne connaît pas les types)
- La méthode equals(...) prend un objet en argument

3) Modifier le code de la méthode add pour implanter l'algorithme d'agrandissement de la table.

Note: essayer de ne pas mettre tout le code dans la méthode add et de ne pas dupliquer de code.

Voir la classe DynamicSet.

On implémente la méthode *grow()* qui agrandi la taille de l'HashSet si la condition de taille est vérifiée au moment de l'ajout (*méthode privée add(...)*)

```
/**
	* if size of DynamicHash is filled with half, increase size to
keep performance.
	*/
```

```
	@SuppressWarnings("unchecked")
	private void grow() {
		var length= this.hashSet.length * 2;
		var cloneHashSet =  this.hashSet.clone();
		var newHashSet =  (Entry<V>[]) new Entry<?>[length];

		this.hashSet = newHashSet;
		for(int i = 0; i < cloneHashSet.length; i++) {
			for(var entry = cloneHashSet[i]; entry != null; entry = 
entry.next) {
				this.add(entry.value);
			}
		}
	}

	/**
	 * add value to HashSet , private method
	 * @param valparam value to add
	 */
	private void addIntoHashSet(V valparam) {
		var position = hash(valparam);
		var tmp = this.hashSet[position];

		if(this.contains(valparam)) {
			return;
		}

		if(size > hashSet.length/2) {
			this.grow();
		}

		while(tmp != null) {
			if(tmp.value == valparam) {
				return;
			}
			tmp = tmp.next;
		}
		var ent = new Entry<V>(valparam, hashSet[position]);
		this.hashSet[position] = ent;
		size++;
	}
```

4) Si vous avez fait en sorte de ne pas dupliquer de code pour la question précédente, il est probable que vous ayez utilisé une ou plusieurs méthodes publiques de la classe DynamicHashSet à l'intérieur du code de la classe. Rappelez pourquoi c'est une mauvaise idée et proposez une solution pour éviter cela.

On ne doit pas utiliser nos méthodes publiques dans nos méthodes de classes de la classe DynamicSet, car si elles sont redéfinies, cela peut altérer le comportement de nos méthodes.

On modifie ceci pour les méthode add(...) et contains(...)

Exercice 3 - Wild cards (optionel)

1) Écrire une méthode addAll, qui permet de recopier une collection d'éléments dans le DynamicHashSet courant.

```
public void addAll(Collection<? extends V> collection){
	for(V value: collection){
		add(value);
	}
}
```

2) Regarder la signature de la méthode addAll dans java.util.Collection. Avez-vous la même signature ? Modifier votre code si nécessaire.

Expliquer ce que veut dire le '?' dans la signature de addAll

On a la même signature.

? signifie qu'on ne défini pas le type à utiliser. Le mot clé « extends » signifie que ? est **sous-type** de V. On utilise « *? super V* » si on considère les classes mères du type V, où «*? extends V*" pour considérer les classes filles de V.

3) Ne devrait-on pas aussi utiliser un '?' dans la signature de la méthode forEach ?

Nous l'avons déjà fait pour passer les tests Junit, avec la signature Consumer<? super V> consumer. C'est possible.

Conclusion

Durant ce quatrième TP de JAVA avancé, j'ai pu implémenter un structure de donnée sans utiliser de méthode existante

J'ai éprouvé des difficultés sur ce TP à cause des Types paramétrés. J'ai un peu de mal à imaginer que l'on peut généraliser un Type en JAVA. J'ai néanmoins développé le bon réflexe de lire la documentation avant d'implémenter mes idées (signature, type de retour, comportement d'une méthode). Un petit rappel sur les opérations bit à bit ne fait pas de mal. J'ai bien compris qu'il ne faut pas utiliser une ou plusieurs méthodes publiques de la classe actuelle à l'intérieur de celle-ci.

Je dois vraiment travailler sur l'implémentation d'une structure de donnée, de manière à m'entrainer, avec par exemple le TP noté 2017 sur le Container.