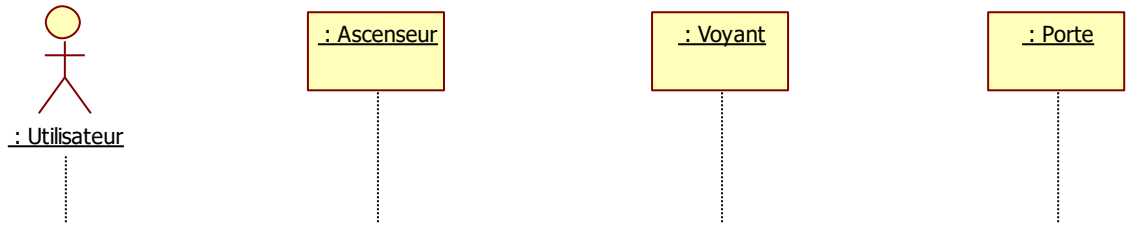
Introduction au diagramme de séquences

Exercice 1 : Un petit échauffement sur le diagramme de séquences...

Un dispositif d'**ascenseur** dispose d'un **voyant** qui est ***éteint*** lorsque l'ascenseur est disponible et ***allumé*** lorsque l'ascenseur est en train de se déplacer. L'ascenseur dispose également d'une **porte** qui permet à un utilisateur d'entrer et de sortir de l'ascenseur : l'ascenseur ne peut se déplacer que lorsque la porte est ***fermée***.
L'ascenseur doit également connaître sa ***position***, c.-à-d. l'étage auquel il se trouve.

1.a Proposez un diagramme de classes afin de modéliser la description précédente.

1.b. Lorsque l'on souhaite utiliser un ascenseur dans 80% des cas, l'ascenseur est disponible et sa porte est fermée. Le scénario suivant va alors se produire. L'utilisateur va appeler l'ascenseur. L'ascenseur va allumer son voyant, puis il va ensuite se déplacer jusqu'à l'étage de l'utilisateur. Arrivé au bon étage, l'ascenseur va ouvrir sa porte afin de laisser entrer l'utilisateur. Une fois dans l'ascenseur, l'utilisateur peut choisir un étage. L'ascenseur va alors fermer la porte, puis se déplacer jusqu'à l'étage souhaité. Une fois arrivé, l'ascenseur ouvrira la porte et éteindra le voyant.
A partir de la description précédente, vous allez devoir représenter le diagramme de séquence correspondant à l'utilisation nominale d'un ascenseur.
Le squelette du diagramme de séquence attendu vous est donné ci-dessous : il ne vous reste plus qu'à rajouter les messages sur ce diagramme.



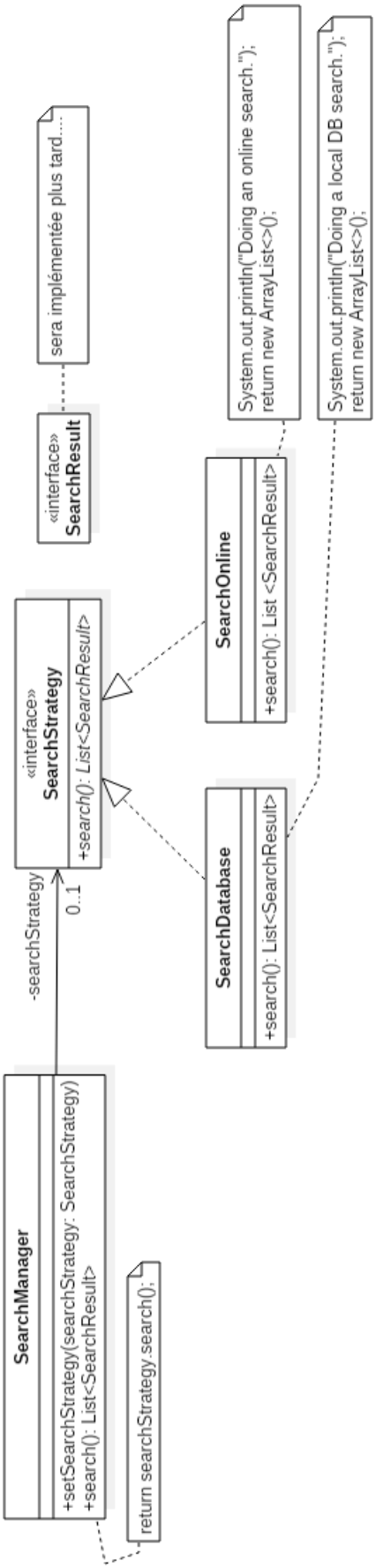
1.c. ***Le diagramme de séquences à la conquête des opérations du diagramme de classes...***
A partir du diagramme de séquences établi dans la question précédente, enrichissez le diagramme de classes construit à la question 1.a

1.d Proposez un ***diagramme de communication*** correspondant au diagramme de séquences obtenu précédemment. Quel est l'intérêt du diagramme de communication en regard du diagramme de classes ?

1.e ***Utilisation de fragment combiné (UML 2.0)***
Transformez votre diagramme de séquence afin de faire apparaître qu'après l'appel de l'utilisateur et l'allumage du voyant, l'ascenseur se déplacera seulement s'il ne se trouve pas au même étage que l'utilisateur...

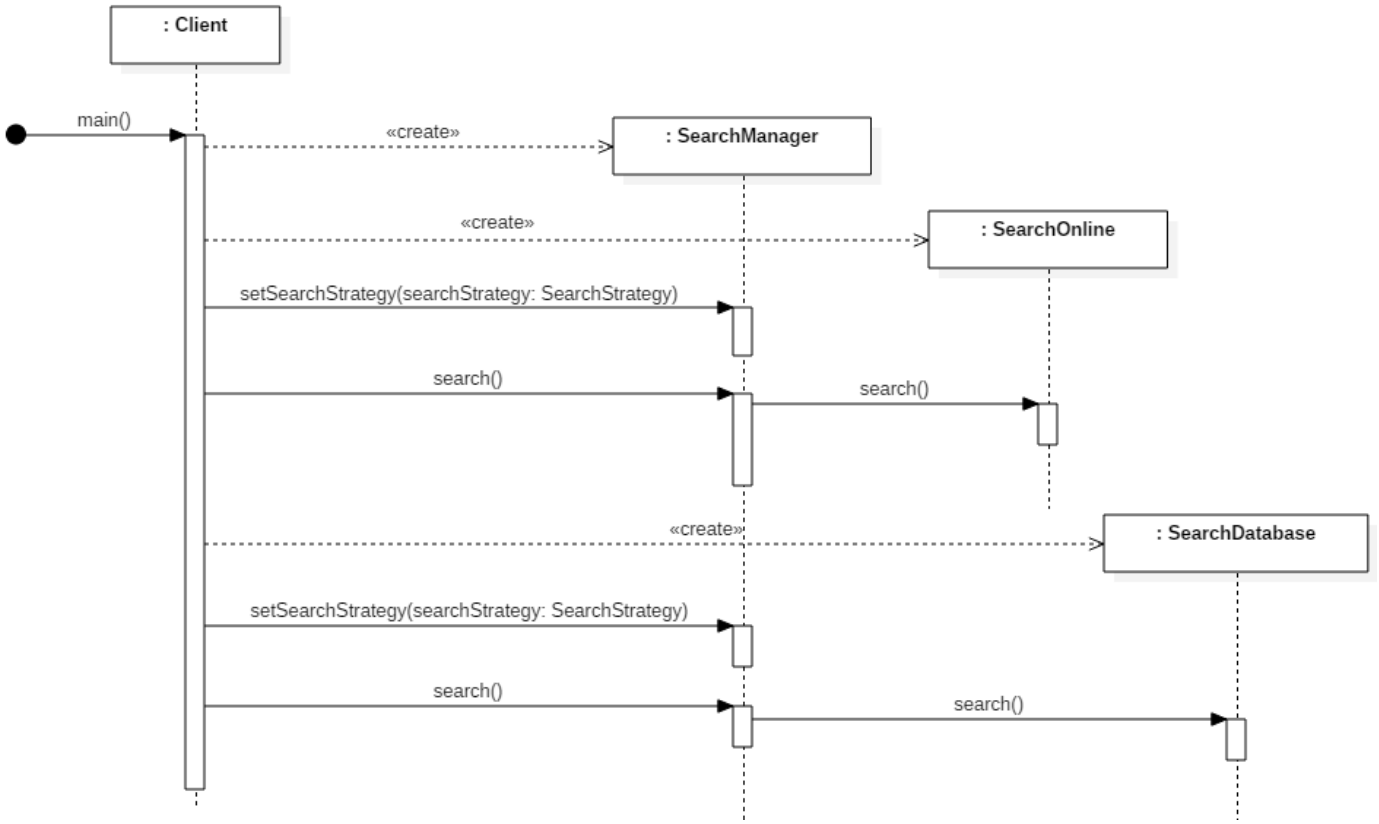
Exercice 2 : Quelle stratégie pour vos recherches

1. Notez, sur le diagramme précédent, la relation appropriée (**EST-UN** ou **A-UN** ou **IMPLÉMENTE(ou REALISE)**) sur chaque flèche du diagramme de classes.



2. Ecrire le **code java** qui devrait être généré à partir de ce diagramme de classes.

3. Le code précédent est testé dans la méthode **main** d'une classe **Client**.
   Le diagramme de séquences suivant modélise les interactions (messages échangés entre les objets) qui apparaissent lors de l'exécution de ce main.



A partir de ce diagramme, écrire ci-dessous, le code java de la méthode **main**.

```
public class Client {

      public static void main(String[] args) {









      }
}
```

Travail à faire en autonomie pour s'entraîner

Exercice provenant d'un ancien contrôle 😉

On considère la classe Java ci-dessous. Le mandataire (`WeatherProxy`) utilise un cache (`LruCache`) pour stocker les résultats des requêtes qu'il fait à un serveur distant au moyen d'un `HttpClient`.

```
public class WeatherProxy {
	private LruCache cache;

	public WeatherProxy() {
		this.cache = new LruCache();
	}

	public WeatherReport getWeather(String city) {
		WeatherReport report = this.cache.get(city);
		if (report == null) {
			report = this.fetchWeather(city);
			this.cache.put(city, report);
		}
		return report;
	}

	private WeatherReport fetchWeather(String city) {
		HttpClient client = new HttpClient();
		String url = "www.meteo.fr/json/";
		return client.request(url + city);
	}
}
```

On suppose qu'il existe par ailleurs les classes `LruCache`, `HttpClient` et `WeatherReport` avec les méthodes utilisées ici.

Un client exécute la méthode `main()` du code suivant :

```
public class WeatherClient {

	public static void main (String[] args) {
		WeatherProxy proxy = new WeatherProxy();
		System.out.println(proxy.getWeather("Limoges"));
	}

}
```

Dessinez le **diagramme de séquences** correspondant au code lancé par le client.
Veillez à bien faire apparaître ***toutes*** les interactions. On ne demande pas les stéréotypes de Jacobson.