

ALOM - TP - Modern Java

Table of Contents

1. Initialisation du projet	1
2. Records	3
3. Streams et Lambdas	4
4. Multi-Line Strings	6

Dans ce TP, nous allons un peu jouer avec les nouveautés du langage Java :

- records
- streams & lambdas
- multi-line strings



Nous avons bien besoin de Java 17 (minimum) pour ce TP !

1. Initialisation du projet

Initialisez votre projet avec ce lien : <https://gitlab-classrooms.cleverapps.io/assignments/9a50d05f-7a5b-4d26-8756-0ee8e82fac12/accept>

Initialisez un `pom.xml`, ainsi que les répertoires `src/main/java` et `src/test/java`:

```
1 <project>
2   <modelVersion>4.0.0</modelVersion>
3   <groupId>com.miage.alom.tp</groupId>
4   <artifactId>w02-modern-java</artifactId>
5   <version>0.1.0</version>
6   <packaging>jar</packaging>
7
8   <properties>
9     <maven.compiler.source>17</maven.compiler.source> ①
10    <maven.compiler.target>17</maven.compiler.target> ②
11  </properties>
12
13  <dependencies>
14  </dependencies>
15
16 </project>
```

① On indique à maven quelle version de Java utiliser pour les sources !

② On indique à maven quelle version de JVM on cible !



Maven considère par défaut que le code est écrit en Java 7 !

Ajoutez les dépendances JUnit :

pom.xml

```
1 <dependency>
2   <groupId>org.junit.jupiter</groupId>
3   <artifactId>junit-jupiter-api</artifactId>
4   <version>5.10.0</version>
5   <scope>test</scope>
6 </dependency>
7 <dependency>
8   <groupId>org.junit.jupiter</groupId>
9   <artifactId>junit-jupiter-engine</artifactId>
10  <version>5.10.0</version>
11  <scope>test</scope>
12 </dependency>
13 <dependency>
14   <groupId>org.assertj</groupId>
15   <artifactId>assertj-core</artifactId>
16   <version>3.24.2</version>
17   <scope>test</scope>
18 </dependency>
```

Il vous faut également surcharger la version du `maven-surefire-plugin` (qui est le plugin maven qui implémente la phase d'exécution des tests).

pom.xml

```
1 <build>
2   <pluginManagement>
3     <plugins>
4       <plugin>
5         <artifactId>maven-surefire-plugin</artifactId>
6         <version>3.1.2</version> ①
7       </plugin>
8     </plugins>
9   </pluginManagement>
10 </build>
```

① On a besoin de la version 2.22.0 minimum pour JUnit 5 comme indiqué [dans la documentation junit](#)

Pour que votre TP soit évalué automatiquement, ajoutez également le fichier `.gitlab-ci.yml` suivant à la racine de votre projet :

.gitlab-ci.yml

```
1 include:
```

```
2 project: gitlab-classrooms/autograding
3 file: maven-junit.yml
```

2. Records

Dans un premier temps, nous allons manipuler des records.

Créez les records suivants :

- `PokemonType`
 - `id`: `int`
 - `name`: `String`
 - `types`: `List<String>`
- `Pokemon`
 - `type`: `PokemonType`
 - `nickname`: `String`
 - `level`: `int`

Le record `PokemonType` représente les types de Pokemon (Pikachu, Rattata, ...).

Le record `Pokemon` représente un Pokemon particulier (Le Pikachu de Sasha, le Rattata de Julien...).

Importez dans votre code les tests unitaires suivants :

```
1 import org.junit.jupiter.api.Test;
2
3 import java.util.List;
4
5 import static org.assertj.core.api.Assertions.assertThat;
6
7 public class RecordsTest {
8
9     @Test
10     public void testPokemonTypeCreation(){
11         var pikachu = new PokemonType(25, "Pikachu", List.of("electric"));
12
13         assertThat(pikachu.id()).isEqualTo(25);
14         assertThat(pikachu.name()).isEqualTo("Pikachu");
15         assertThat(pikachu.types()).containsOnly("electric");
16     }
17
18     @Test
19     public void testPokemonCreation(){
20         var geodude = new PokemonType(74, "Racaillou", List.of("rock", "ground"));
21         var petersGeodude = new Pokemon(geodude, "Racaillou de Pierre", 12);
22     }
```

```

23     assertThat(petersGeodude.nickname()).isEqualTo("Racaillou de Pierre");
24     assertThat(petersGeodude.type()).isEqualTo(geodude);
25     assertThat(petersGeodude.level()).isEqualTo(12);
26 }
27
28 }

```

3. Streams et Lambdas

Nous allons maintenant charger une liste de types de Pokemons, et la manipuler avec des Streams.

Récupérez le fichier [pokemons.json](#), et placez-le dans le répertoire `src/main/resources` de votre projet.

Pour charger le fichier JSON, nous allons devoir utiliser la librairie `jackson-databind`:

pom.xml

```

1 <dependency>
2     <groupId>com.fasterxml.jackson.core</groupId>
3     <artifactId>jackson-databind</artifactId>
4     <version>2.15.2</version>
5 </dependency>

```

Importez et complétez la classe suivante :

```

1 public class PokemonStreams {
2
3     private Collection<PokemonType> pokemonsTypes;
4
5     public void loadPokemonTypes() throws IOException {
6         var objectMapper = new ObjectMapper();
7         objectMapper.configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES,
8 false);
9
10        this.pokemonsTypes = objectMapper.readValue(new
11        FileInputStream("src/main/resources/pokemons.json"), new
12        TypeReference<Collection<PokemonType>>() {});
13    }
14
15    public List<PokemonType> sortById(){
16        // TODO
17    }
18
19    public Set<PokemonType> findByType(String type) {
20        // TODO
21    }
22 }

```

```

20     public Optional<PokemonType> findFirstByTypes(String... types) {
21         // TODO
22     }
23 }

```

Vous pouvez valider vos développements avec la classe de test suivante :

```

1  public class PokemonStreamsTest {
2
3      PokemonStreams pokemonStreams;
4
5      @BeforeEach
6      void setUp() throws IOException {
7          pokemonStreams = new PokemonStreams();
8          pokemonStreams.loadPokemonTypes();
9      }
10
11     @Test
12     public void testSortById(){
13         assertThat(pokemonStreams.sortById())
14             .extracting(it -> it.id())
15             .isSorted();
16     }
17
18     @Test
19     public void testFindElectricPokemons(){
20         assertThat(pokemonStreams.findByType("electric"))
21             .hasSize(9);
22     }
23
24     @Test
25     public void testFindFirePokemons(){
26         assertThat(pokemonStreams.findByType("fire"))
27             .hasSize(12);
28     }
29
30     @Test
31     public void testFindFirstPsychicPokemon(){
32         assertThat(pokemonStreams.findFirstByTypes("psychic"))
33             .isNotEmpty()
34             .get()
35             .extracting("name")
36             .isEqualTo("abra");
37     }
38
39     @Test
40     public void testFindFirstUnknownPokemon(){
41         assertThat(pokemonStreams.findFirstByTypes("unknown"))
42             .isEmpty();
43     }

```

4. Multi-Line Strings

Importez le test suivant, et faites ce qu'il faut pour qu'il passe !

```

1 public class TextBlockTest {
2
3     ObjectMapper objectMapper;
4
5     @BeforeEach
6     void setUp() {
7         objectMapper = new ObjectMapper();
8         objectMapper.configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES,
9             false);
10    }
11
12    @Test
13    void pokemonAsJsonString() throws JsonProcessingException {
14        var jsonStringOldFashioned = "{\n" +
15            "    \"id\": 47,\n" +
16            "    \"name\": \"parasect\",\n" +
17            "    \"baseExperience\": 142,\n" +
18            "    \"weight\": 295,\n" +
19            "    \"height\": 10,\n" +
20            "    \"types\": [\n" +
21            "        \"grass\",\n" +
22            "        \"bug\"\n" +
23            "    ],\n" +
24            "    \"stats\": {\n" +
25            "        \"speed\": 30,\n" +
26            "        \"attack\": 95,\n" +
27            "        \"defense\": 80,\n" +
28            "        \"hp\": 60\n" +
29            "    },\n" +
30            "    \"sprites\": {\n" +
31            "        \"front_default\":
32            \"https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/47.png\",
33            \"back_default\":
34            \"https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/back/47.png\",
35            }
36            }";
37        var pokemonTypeFromJsonString = objectMapper.
38            readValue(jsonStringOldFashioned, PokemonType.class);
39        System.out.println("pokemonType.toString() = " +
40            pokemonTypeFromJsonString.toString());
41    }
42}

```

```
37
38     // TODO : écrivez un text block ici !
39     String textBlockString = null;
40     var pokemonTypeFromTextBlock = objectMapper.readValue(textBlockString,
    PokemonType.class);
41
42     assertThat(pokemonTypeFromJsonString).isEqualTo(pokemonTypeFromTextBlock);
43 }
44 }
```