

Intégration Continue

Exercice1

Sur gitHub,

- créer un dossier `.github`
- créer un sous dossier `workflows`
- créer un fichier `salut.yml`

Voici le contenu du fichier:

```
1  name: Bonjour
2
3  on: [push] # Ce workflow se déclenche sur un push dans le dépôt
4
5  jobs:
6    salutations: # Nom du job : Salutations
7      name: salutations
8      runs-on: ubuntu-latest # OS où s'exécute le job
9      steps:
10       - name: Hello #une tâche pour saluer
11         env:
12           personneASaluer: 'Mon voisin Totoro'
13           run: echo "Bonjour ${personneASaluer}"
14       - name: L'heure de salutation # affiche l'heure de la précédente tâche
15         run: echo "L'heure était $(date)."
```

Respecter bien les indentations!

Explications:

```
1  name: Bonjour
```

Cette ligne donne un nom au workflow dans GitHub Actions. Ici, le workflow s'appelle "**Bonjour**".

```
1  on: [push] # Ce workflow se déclenche sur un push dans le dépôt
```

La clé `on` spécifie les événements qui déclenchent l'exécution du workflow. Ici, le workflow se lance à chaque **push** dans le dépôt.

```
1  jobs:
```

Cette clé contient tous les jobs (ou étapes) du workflow. Un workflow peut avoir plusieurs jobs, exécutés en parallèle ou en série selon les besoins.

```
1 | salutations: # Nom du job : salutations
```

Ce nom (`salutations`) est un identifiant du job au sein du fichier YAML. Le nom peut être choisi librement, tant qu'il respecte les conventions YAML.

```
1 | name: salutations
```

Cette ligne donne un nom plus lisible au job, ici **"Salutations"**.

```
1 | runs-on: ubuntu-latest # OS où s'exécute le job
```

La clé `runs-on` spécifie l'environnement d'exécution du job. Ici, il s'exécute sur `ubuntu-latest`, qui est la dernière version d'Ubuntu disponible sur GitHub Actions.

```
1 | steps:
```

La clé `steps` définit les différentes étapes (ou tâches) que le job va exécuter. Chaque étape est exécutée dans l'ordre où elle est listée.

```
1 | - name: Hello #une tâche pour saluer
2 |
```

Cette étape s'appelle **"Hello"** et son but est d'afficher un message de salutation.

```
1 | env:
2 |   personneASaluer: 'Mon voisin Totoro'
```

Cette sous-clé `env` définit des variables d'environnement pour l'étape en cours. Ici, une variable `personneASaluer` est définie avec la valeur **"Mon voisin Totoro"**.

```
1 | run: echo "Bonjour ${personneASaluer}"
```

La commande `run` exécute une commande shell. Ici, elle affiche un message de salutation utilisant la variable `personneASaluer`, qui affiche : **"Bonjour Mon voisin Totoro"**.

```
1 | - name: L'heure de salutation # affiche heure précédente tâche
2 |
```

Cette étape s'appelle **"L'heure de salutation"** et affiche l'heure actuelle.

```
1 | run: echo "L'heure était $(date)."
```

Cette commande affiche l'heure au moment de l'exécution, utilisant `$(date)` pour obtenir l'heure actuelle (template literal)

Le fichier `salut.yml` est un workflow simple qui s'exécute après un `push` et affiche une salutation, suivie de l'heure exacte de l'exécution !

On peut désactiver le workflow dans Actions, cliquez sur le nom du workflow et sélectionner `Disable workflow`

The screenshot shows the GitHub Actions interface. On the left, the 'Actions' tab is active, displaying a list of workflows under 'All workflows'. The 'Bonjour' workflow is selected. The main area shows the 'Bonjour' workflow details, including a search bar for 'Filter workflow runs' and a list of 5 workflow runs. The runs are as follows:

Status	ID	Description	Branch	Time
Failed	Create salut.yml	Bonjour #1: Commit c7d02af pushed by olfabre	main	last week, Failure
Success	Update salut.yml	Bonjour #2: Commit c10136d pushed by olfabre	main	last week, 11s
Success	241024B	Bonjour #3: Commit a84ab87 pushed by olfabre	main	last week, 11s
Success	301024A	Bonjour #4: Commit 0b44fcd pushed by olfabre	main	1 hour ago, 10s
Success	301024B	Bonjour #5: Commit ff79638 pushed by olfabre	main	46 minutes ago, 14s

Exercice 2

Nous devons sélectionner la branche `dev`

Il contient un code `pom.xml` qui est la configuration principale d'un projet Maven, avec des sections pour les dépendances, les propriétés, et les plugins pour la compilation, les tests, l'intégration de qualité de code et la couverture de tests

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/xsd/maven-4.0.0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4
5   <groupId>ceri</groupId>
```

```
6 <artifactId>moduleListeSimple</artifactId>
7 <version>1.0-SNAPSHOT</version>
8 <properties>
9     <maven.compiler.source>21</maven.compiler.source>
10    <maven.compiler.target>21</maven.compiler.target>
11    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
12    <sonar.organization>ceri-m1-devops</sonar.organization>
13    <sonar.host.url>https://sonarcloud.io</sonar.host.url>
14 </properties>
15 <dependencies>
16     <dependency>
17         <groupId>org.junit.jupiter</groupId>
18         <artifactId>junit-jupiter</artifactId>
19         <version>5.10.2</version>
20         <scope>test</scope>
21     </dependency>
22 </dependencies>
23
24 <build>
25     <plugins>
26     <plugin>
27         <groupId>org.sonarsource.scanner.maven</groupId>
28         <artifactId>sonar-maven-plugin</artifactId>
29         <version>4.0.0.4121</version>
30     </plugin>
31     <plugin>
32         <groupId>org.apache.maven.plugins</groupId>
33         <artifactId>maven-compiler-plugin</artifactId>
34         <version>3.8.1</version>
35         <configuration>
36             <release>21</release>
37         </configuration>
38     </plugin>
39     <plugin>
40         <groupId>org.jacoco</groupId>
41         <artifactId>jacoco-maven-plugin</artifactId>
42         <version>0.8.12</version>
43         <executions>
44             <execution>
45                 <goals>
46                     <goal>prepare-agent</goal>
47                 </goals>
48             </execution>
49             <execution>
50                 <id>report</id>
51                 <phase>verify</phase>
52                 <goals>
53                     <goal>report</goal>
54                 </goals>
```

```

55         </execution>
56     </executions>
57 </plugin>
58 <plugin>
59     <groupId>org.apache.maven.plugins</groupId>
60     <artifactId>maven-surefire-plugin</artifactId>
61     <version>3.5.0</version>
62 <!--     <configuration>
63         <argLine>${argLine}</argLine>
64     </configuration>-->
65 </plugin>
66 </plugins>
67 </build>
68 </project>

```

Structure et Métadonnées de Projet

```

1 <project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
2     <modelVersion>4.0.0</modelVersion>

```

- **Déclaration des namespaces** : indique que le projet suit le modèle POM de Maven version 4.0.0.

```

1     <groupId>ceri</groupId>
2     <artifactId>moduleListeSimple</artifactId>
3     <version>1.0-SNAPSHOT</version>

```

- **groupId** : identifie l'organisation ou l'auteur du projet, ici "ceri".
- **artifactId** : le nom unique de l'artifact, ici "moduleListeSimple".
- **version** : version du projet ; "1.0-SNAPSHOT" indique une version en cours de développement.

Propriétés

```

1     <properties>
2         <maven.compiler.source>21</maven.compiler.source>
3         <maven.compiler.target>21</maven.compiler.target>
4         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
5         <sonar.organization>ceri-m1-devops</sonar.organization>
6         <sonar.host.url>https://sonarcloud.io</sonar.host.url>
7     </properties>

```

- **Version Java** : `<maven.compiler.source>` et `<maven.compiler.target>` spécifient la version de Java à utiliser, ici Java 21.
- **Encodage** : `<project.build.sourceEncoding>` indique que le projet utilise l'encodage UTF-8.
- **Configuration Sonar** : les propriétés `sonar.organization` et `sonar.host.url` configurent l'organisation et l'URL de SonarCloud pour l'analyse de la qualité du code.

Dépendances

```
1 <dependencies>
2     <dependency>
3         <groupId>org.junit.jupiter</groupId>
4         <artifactId>junit-jupiter</artifactId>
5         <version>5.10.2</version>
6         <scope>test</scope>
7     </dependency>
8 </dependencies>
```

- **JUnit** : la dépendance `junit-jupiter` version 5.10.2 est ajoutée pour écrire et exécuter des tests unitaires. Le `scope` de cette dépendance est défini sur `"test"`, donc elle sera utilisée uniquement pendant la phase de test

Configuration de Build et Plugins

```
1 <build>
2     <plugins>
```

- **Plugins Maven** : cette section configure les plugins Maven utilisés pour différentes tâches comme la compilation, les tests et l'analyse de couverture.

Plugin Sonar

```
1     <plugin>
2         <groupId>org.sonarsource.scanner.maven</groupId>
3         <artifactId>sonar-maven-plugin</artifactId>
4         <version>4.0.0.4121</version>
5     </plugin>
```

- `sonar-maven-plugin` : configure le plugin Sonar pour l'intégration avec SonarCloud. Cela permet de générer des rapports de qualité de code à chaque build.

Plugin de Compilation

```
1      <plugin>
2          <groupId>org.apache.maven.plugins</groupId>
3          <artifactId>maven-compiler-plugin</artifactId>
4          <version>3.8.1</version>
5          <configuration>
6              <release>21</release>
7          </configuration>
8      </plugin>
```

- **maven-compiler-plugin** : ce plugin compile le code source Java. La version Java est spécifiée avec `<release>21</release>`, donc Java 21 sera utilisé.

Plugin JaCoCo

```
1      <plugin>
2          <groupId>org.jacoco</groupId>
3          <artifactId>jacoco-maven-plugin</artifactId>
4          <version>0.8.12</version>
5          <executions>
6              <execution>
7                  <goals>
8                      <goal>prepare-agent</goal>
9                  </goals>
10             </execution>
11             <execution>
12                 <id>report</id>
13                 <phase>verify</phase>
14                 <goals>
15                     <goal>report</goal>
16                 </goals>
17             </execution>
18         </executions>
19     </plugin>
```

- **jacoco-maven-plugin** : ce plugin est utilisé pour mesurer la couverture des tests.
 - L'exécution `prepare-agent` initialise l'agent JaCoCo pendant la phase de test.
 - L'exécution `report` génère un rapport de couverture pendant la phase `verify`.

Plugin Surefire

```
1         <plugin>
2             <groupId>org.apache.maven.plugins</groupId>
3             <artifactId>maven-surefire-plugin</artifactId>
4             <version>3.5.0</version>
5     <!--             <configuration>
6                     <argLine>${argLine}</argLine>
7             </configuration>-->
8         </plugin>
9     </plugins>
10 </build>
```

- **maven-surefire-plugin** : ce plugin exécute les tests JUnit. La section de configuration commentée (balises `<!-- ... -->`) peut être utilisée pour ajouter des arguments spécifiques lors de l'exécution des tests.

Le fichier `pom.xml` est bien structuré pour un projet Java avec les éléments essentiels pour la compilation, l'analyse de la qualité de code avec Sonar, la couverture de test avec JaCoCo, et les tests avec JUnit. Les versions et les configurations choisies sont récentes et adaptées à un projet Java moderne utilisant Maven.

Exercice 2

On se branche sur `dev`

On créer un dossier `.github`

On créer un dossier `workflows`

on va créer le fichier `buildJava.yml` pour configurer le workflow GitHub Actions en fonction des instructions données:

- Le nom du workflow sera "Java CI with Maven - Test and package"
- déclencher lorsqu'on émet une pull request sur la branche main.
- Il ne contient qu'un seul job
- s'exécute sur la dernière version d'ubuntu.

Il y aura 3 étapes:

- utilisez l'action `actions/checkout@v4` pour se placer dans le dépôt courant (vous pouvez consulter <https://github.com/actions/checkout>)
- utilisez ensuite l'action `actions/setup-java@v4` (vous pouvez consulter <https://github.com/actions/setup-java>). Choisissez la version 21 et la distribution corretto

- enfin faire construire votre package avec maven : pour cela un fichier pom.xml vous est fourni, qui définit les dépendances impliquées dans ce projet. La commande pour construire est mvn -B package, elle lancera les tests avec JUnit.

buildJava.yml

```
1 name: Java CI with Maven - Test and package
2
3 # Déclenchement du workflow: ce workflow s'exécute à chaque pull request vers la
  branche main.
4 on:
5   pull_request:
6     branches:
7       - main
8
9 jobs:
10   build:
11     # Job "build": ce job s'exécute sur la dernière version d'Ubuntu (ubuntu-
latest).
12     name: Build and Test
13     runs-on: ubuntu-latest
14
15     # Etapes
16     steps:
17
18     # Checkout code: utilise l'action actions/checkout@v4 pour cloner le dépôt
actuel et accéder aux fichiers.
19     - name: Checkout code v4
20       uses: actions/checkout@v4
21
22     # Setup Java: utilise l'action actions/setup-java@v4 pour configurer
l'environnement Java. Nous spécifions la version 21 avec la distribution corretto.
23     - name: Set up Java v4
24       uses: actions/setup-java@v4
25       with:
26         java-version: '21'
27         distribution: 'corretto'
28
29
30     # Build and test with Maven: exécute la commande mvn -B package pour
compiler le projet, exécuter les tests JUnit et construire le package en fonction
des configurations du fichier pom.xml
31     - name: Build and test with Maven
32       run: mvn -B package
```

voici le code sans le commentaire

```

1  name: Java CI with Maven - Test and package
2
3  on:
4    pull_request:
5      branches:
6        - main
7
8  jobs:
9    build:
10     name: Build and Test
11     runs-on: ubuntu-latest
12
13     steps:
14       - name: Checkout code
15         uses: actions/checkout@v4
16
17       - name: Set up Java
18         uses: actions/setup-java@v4
19         with:
20           java-version: '21'
21           distribution: 'corretto'
22
23       - name: Build and test with Maven
24         run: mvn -B package
25

```

lors d'un `new pull-request` l'action Java CI with Maven - Test and package s'exécute

← Java CI with Maven - Test and package

✓ Pull request Dev à Main #2

Re-run all jobs ...

Summary

Jobs

✓ Build and Test

Run details

Usage

Workflow file

Triggered via pull request 18 minutes ago

olfabre synchronize #1

dev

Status

Success

Total duration

21s

Artifacts

—

buildJava.yml

on: pull_request

✓ Build and Test

14s

⌵

—

+

Exercice 3 – Qualité du code

Nous allons observer la qualité de notre code en utilisant la plateforme <https://sonarcloud.io> (inscription avec mon gitHub)

et le rapport est sur https://sonarcloud.io/project/configuration?id=CERI-M1-DevOps_tpintegrationcontinue-olfabre

Pour pouvoir lancer l'analyse de la qualité, il faut changer la commande maven pour:

```
1 mvn -B verify org.sonarsource.scanner.maven:sonar-maven-plugin:sonar
2 -Dsonar.projectKey=$(echo ${github.repository} | sed 's-/-_-' )
```

Pour intégrer SonarCloud dans votre workflow GitHub Actions et exécuter l'analyse de la qualité de votre code, vous devrez modifier la commande Maven et configurer le jeton d'authentification (`SONAR_TOKEN`). Voici comment procéder pour que la configuration respecte les nouvelles spécifications.

modification du fichier `buildJava.yml`

```
1 name: Java CI with Maven - Test and package
2
3 on:
4   pull_request:
5     branches:
6       - main
7
8 jobs:
9   build:
10    name: Build and Test
11    runs-on: ubuntu-latest
12
13    steps:
14      - name: Checkout code
15        uses: actions/checkout@v4
16
17      - name: Set up Java
18        uses: actions/setup-java@v4
19        with:
20          java-version: '21'
21          distribution: 'corretto'
22
23      - name: Build, test, and analyze with Maven and SonarCloud
24        env:
25          SONAR_TOKEN: ${secrets.SONAR_TOKEN}
26        run: mvn -B verify org.sonarsource.scanner.maven:sonar-maven-plugin:sonar
              -Dsonar.projectKey=$(echo ${github.repository} | sed 's-/-_-' )
```

The screenshot displays the SonarCloud web interface for a Pull Request. The left sidebar shows the project name 'tpintegrationcontinue-ol fabre' and navigation links for Overview, Main Branch, Pull Requests (1), and Branches (1). The main content area shows the Pull Request summary for '1 - Pull request Dev à Main'. The Quality Gate is 'Passed' with a green checkmark. Below this, there are four metrics: New Issues (49), Accepted Issues (0), Coverage (95.4%), and Duplications (0.0%). Each metric has a small circular progress indicator. The Coverage and Duplications metrics also show the required percentage and the number of new lines covered or duplicated. A warning banner at the top right indicates that the last analysis has warnings.

sonarcloud

My Projects My Issues Explore Q

CERI-M1-DevOps > tpintegrationcontinue-ol fabre > Pull Requests > 1 - Pull request Dev à Main

Summary Issues Security Hotspots Measures Code

The last analysis has warnings. [See details](#)

PR Summary

503 New Lines • dev → main

Quality Gate **Passed** Last analysis 1 minute ago • e3ce1ae8

New Issues 49 No conditions set	Accepted Issues 0 Valid issues that were not fixed	
Coverage 95.4% Required ≥ 80.0% on 104 New Lines to cover 95.4% Estimated after merge	Duplications 0.0% Required ≤ 3.0% on 503 New Lines 0.0% Estimated after merge	Security Hotspots 0 No conditions set

on modifie les fichiers et les codes de façon à obtenir 100%

sonarcloud

tpintegrationcontinue-ol

fabre

PUBLIC

Overview

Main Branch

Pull Requests 1

Branches 1

Information

Collapse

My Projects My Issues Explore

Q

1 – Pull request Dev à Main

The last analysis has warnings. [See details](#)

Summary Issues Security Hotspots Measures Code

```

30 ...      </plugin>
31 ...      <plugin>
32 ...        <groupId>org.apache.maven.plugins</groupId>
33 ...        <artifactId>maven-compiler-plugin</artifactId>
34 ...        <version>3.8.1</version>
35 ...        <configuration>
36 ...          <release>21</release>
37 ...        </configuration>
38 ...      </plugin>
39 ...      <plugin>
40 ...        <groupId>org.jacoco</groupId>
41 ...        <artifactId>jacoco-maven-plugin</artifactId>
42 ...        <version>0.8.12</version>
43 ...        <executions>
44 ...          <execution>
45 ...            <goals>
46 ...              <goal>prepare-agent</goal>
47 ...            </goals>
48 ...          </execution>
49 ...          <execution>
50 ...            <id>report</id>
51 ...            <phase>verify</phase>
52 ...            <goals>
53 ...              <goal>report</goal>
54 ...            </goals>
55 ...          </execution>
56 ...        </executions>
57 ...      </plugin>
58 ...      <plugin>
59 ...        <groupId>org.apache.maven.plugins</groupId>
60 ...        <artifactId>maven-surefire-plugin</artifactId>
61 ...        <version>3.5.0</version>
62 ...        <!--
63 ...          <configuration>
64 ...            <argLine>${argLine}</argLine>
65 ...          </configuration>
66 ...        </plugin>
67 ...      </plugins>
68 ...    </build>
69 ...  </project>

```

© 2018-2024 SonarSource SA. All rights reserved. Terms Pricing Privacy Cookie Policy Security Community Documentation Contact us Status About

Question 4:

Voici le fichier modifié

```

1  name: Java CI with Maven - Test and package
2
3  on:
4    pull_request:
5      branches:
6        - main
7
8  jobs:
9    build:
10     name: Build and Test
11     runs-on: ubuntu-latest
12
13     steps:
14       - name: Checkout code
15         uses: actions/checkout@v4
16
17       - name: Set up Java

```

```

18     uses: actions/setup-java@v4
19     with:
20       java-version: '21'
21       distribution: 'corretto'
22
23   - name: Cache Maven dependencies
24     uses: actions/cache@v4
25     with:
26       path: ~/.m2/repository
27       key: ${ runner.os }-maven-${ hashFiles('**/*.xml') }
28       restore-keys: |
29         ${ runner.os }-maven-
30
31   - name: Cache SonarQube packages
32     uses: actions/cache@v1
33     with:
34       path: ~/.sonar/cache
35       key: ${ runner.os }-sonar
36       restore-keys: ${ runner.os }-sonar
37
38
39   - name: Build, test, and analyze with Maven and SonarCloud
40     env:
41       SONAR_TOKEN: ${ secrets.SONAR_TOKEN }
42       run: mvn -B verify org.sonarsource.scanner.maven:sonar-maven-plugin:sonar
43         -Dsonar.projectKey=$(echo ${ github.repository } | sed 's-/-_-' )
44

```

Explication des modifications :

1. Étape "Cache Maven dependencies" :

- Cette étape utilise l'action `actions/cache@v4` pour mettre en cache le répertoire des dépendances Maven (`~/.m2/repository`), ce qui permet d'éviter de télécharger à chaque build les mêmes dépendances.
- La clé de cache est basée sur un hachage des fichiers `.xml` (comme `pom.xml`), ce qui permet de créer un nouveau cache chaque fois que ces fichiers changent.
- Le paramètre `restore-keys` permet d'utiliser un cache existant s'il existe, même si la clé exacte ne correspond pas.

2. Autres étapes inchangées :

- Vous continuez à configurer Java avec `actions/setup-java@v4` et à exécuter votre build Maven avec SonarCloud comme avant.

Avec cette configuration, à chaque nouveau build, les dépendances Maven seront récupérées depuis le cache si elles sont déjà présentes, ce qui accélérera le processus de build en évitant les téléchargements redondants.

Question 5

Dans un premier temps, je documente quelques fonctions du fichier ListeSimple.java

```
1 package liste;
2
3 public class ListeSimple {
4     private long size;
5     Noeud tete;
6
7     /**
8      * Retourne la taille de la liste.
9      * @return Le nombre de nœuds dans la liste.
10    */
11    public long getSize() {
12        return size;
13    }
14
15    /**
16     * Ajoute un nouvel élément en tête de la liste.
17     * @param element L'élément à ajouter en tête de la liste.
18    */
19    public void ajout(int element) {
20        tete = new Noeud(element, tete);
21        size++;
22    }
23
24    /**
25     * Modifie la première occurrence d'un élément avec une nouvelle valeur.
26     * @param element L'élément à rechercher dans la liste.
27     * @param nouvelleValeur La nouvelle valeur pour remplacer l'élément trouvé.
28    */
29    public void modifiePremier(Object element, Object nouvelleValeur) {
30        Noeud courant = tete;
31        while (courant != null && courant.getElement() != element)
32            courant = courant.getSuivant();
33        if (courant != null)
34            courant.setElement(nouvelleValeur);
35    }
36
37    /**
38     * Modifie toutes les occurrences d'un élément avec une nouvelle valeur.
39     * @param element L'élément à rechercher dans la liste.
```

```

40      * @param nouvelleValeur La nouvelle valeur pour remplacer chaque occurrence
de l'élément trouvé.
41      */
42      public void modifieTous(Object element, Object nouvelleValeur) {
43          Noeud courant = tete;
44          while (courant != null) {
45              if (courant.getElement() == element)
46                  courant.setElement(nouvelleValeur);
47              courant = courant.getSuivant();
48          }
49      }
50
51      /**
52       * Retourne une représentation en chaîne de la liste.
53       * @return Une chaîne représentant la liste.
54       */
55      public String toString() {
56          StringBuilder sb = new StringBuilder("ListeSimple(");
57          Noeud n = tete;
58          while (n != null) {
59              sb.append(n);
60              n = n.getSuivant();
61              if (n != null)
62                  sb.append(", ");
63          }
64          sb.append(")");
65          return sb.toString();
66      }
67
68      /**
69       * Supprime la première occurrence d'un élément dans la liste.
70       * @param element L'élément à supprimer de la liste.
71       */
72      public void supprimePremier(Object element) {
73          if (tete != null) {
74              if (tete.getElement() == element) {
75                  tete = tete.getSuivant();
76                  size--;
77                  return;
78              }
79              Noeud precedent = tete;
80              Noeud courant = tete.getSuivant();
81              while (courant != null && courant.getElement() != element) {
82                  precedent = precedent.getSuivant();
83                  courant = courant.getSuivant();
84              }
85              if (courant != null) {
86                  precedent.setSuivant(courant.getSuivant());
87                  size--;

```



```

88         }
89     }
90 }
91
92 /**
93  * Supprime toutes les occurrences d'un élément dans la liste.
94  * @param element L'élément à supprimer de la liste.
95  */
96 public void supprimeTous(int element) {
97     tete = supprimeTousRekurs(element, tete);
98 }
99
100 /**
101  * Supprime récursivement toutes les occurrences d'un élément dans la sous-
102  * liste à partir d'un nœud donné.
103  * @param element L'élément à supprimer de la sous-liste.
104  * @param tete La tête de la sous-liste.
105  * @return La nouvelle tête de la sous-liste après suppression des
106  * occurrences.
107  */
108 public Noeud supprimeTousRekurs(Object element, Noeud tete) {
109     if (tete != null) {
110         Noeud suiteListe = supprimeTousRekurs(element, tete.getSuivant());
111         if (tete.getElement() == element) {
112             size--;
113             return suiteListe;
114         } else {
115             tete.setSuivant(suiteListe);
116             return tete;
117         }
118     } else return null;
119 }
120
121 /**
122  * Retourne l'avant-dernier nœud de la liste.
123  * @return Le nœud avant le dernier dans la liste, ou null si la liste est
124  * trop courte.
125  */
126 public Noeud getAvantDernier() {
127     if (tete == null || tete.getSuivant() == null)
128         return null;
129     else {
130         Noeud courant = tete;
131         Noeud suivant = courant.getSuivant();
132         while (suivant.getSuivant() != null) {
133             courant = suivant;
134             suivant = suivant.getSuivant();
135         }
136         return courant;
137     }
138 }

```

```

134     }
135 }
136
137 /**
138  * Inverse l'ordre des nœuds dans la liste.
139  */
140 public void inverser() {
141     Noeud precedent = null;
142     Noeud courant = tete;
143     while (courant != null) {
144         Noeud next = courant.getSuivant();
145         courant.setSuivant(precedent);
146         precedent = courant;
147         courant = next;
148     }
149     tete = precedent;
150 }
151
152 /**
153  * Retourne le nœud précédent d'un nœud donné.
154  * @param r Le nœud dont on souhaite obtenir le précédent.
155  * @return Le nœud précédent, ou null si le nœud est la tête de la liste.
156  */
157 public Noeud getPrecedent(Noeud r) {
158     Noeud precedent = tete;
159     Noeud courant = precedent.getSuivant();
160     while (courant != r) {
161         precedent = courant;
162         courant = courant.getSuivant();
163     }
164     return precedent;
165 }
166
167 /**
168  * Échange deux nœuds donnés dans la liste.
169  * @param r1 Le premier nœud à échanger.
170  * @param r2 Le second nœud à échanger.
171  */
172 public void echanger(Noeud r1, Noeud r2) {
173     if (r1 == r2)
174         return;
175     Noeud precedentR1;
176     Noeud precedentR2;
177     if (r1 != tete && r2 != tete) {
178         precedentR1 = getPrecedent(r1);
179         precedentR2 = getPrecedent(r2);
180         precedentR1.setSuivant(r2);
181         precedentR2.setSuivant(r1);
182     } else if (r1 == tete) {

```

```

183         precedentR2 = getPrecedent(r2);
184         precedentR2.setSuivant(tete);
185         tete = r2;
186     }
187     else {
188         precedentR1 = getPrecedent(r1);
189         precedentR1.setSuivant(tete);
190         tete = r1;
191     }
192     Noeud temp = r2.getSuivant();
193     r2.setSuivant(r1.getSuivant());
194     r1.setSuivant(temp);
195 }
196 }
197

```

Ensuite, je désactive `buildJava.yml` sur gitHub.

On crée `documentation.yml`

```

1  name: Documentation
2
3  on:
4    pull_request:
5      types: [closed]
6      branches:
7        - main
8
9  jobs:
10    generate-docs:
11      if: github.event.pull_request.merged == true
12      runs-on: ubuntu-latest
13
14      steps:
15        - name: Checkout repository
16          uses: actions/checkout@v2
17
18        - name: Set up JDK
19          uses: actions/setup-java@v2
20          with:
21            distribution: 'temurin'
22            java-version: '11'
23
24        - name: Generate Javadoc
25          run: mvn -B javadoc:javadoc
26
27        - name: Deploy to GitHub Pages
28          uses: peaceiris/actions-gh-pages@v4
29          with:

```

```
30     github_token: ${ secrets.GITHUB_TOKEN }
31     publish_dir: ./target/site/apidocs
32
```

on va le faire sur un commit et non un close request-merge pour la praticité du tp

```
1  name: Generate and Deploy Documentation
2  permissions:
3    contents: write
4  on:
5    push:
6      branches:
7        - dev
8
9  jobs:
10   generate-doc:
11     runs-on: ubuntu-latest
12     steps:
13       - name: Checkout code
14         uses: actions/checkout@v2
15
16       - name: Set up JDK
17         uses: actions/setup-java@v2
18         with:
19           distribution: 'temurin'
20           java-version: '21'
21
22       - name: Generate Javadoc
23         run: mvn -B javadoc:javadoc
24
25       - name: Deploy to GitHub Pages
26         uses: peaceiris/actions-gh-pages@v4
27         with:
28           github_token: ${ secrets.GITHUB_TOKEN }
29           publish_dir: ./target/reports/apidocs
30
```

la doc est générée dans la branche `gh-pages`

pour obtenir une page à l'adresse <https://ceri-m1-devops.github.io/tpintegrationcontinue-olfabre/>

on va sur la branche `gh-pages`, ensuite on va dans settings / pages /

Dans Build and deployment / source / Deploy from a branch

Dans Branch, je choisie gh-pages et root et SAVE

Fin du TP