Fondamentaux du développeur

Séance 5 - TP 1 - Premiers pas avec le débogueur (VSCode)

Philippe Roussille

3iL INGENIEURS
</les sciences informatiques>

1 Objectif du TP

Ce TP a pour but de vous apprendre à utiliser le débogueur intégré de VSCode pour :

- Placer et utiliser des breakpoints.
- Exécuter le programme pas-à-pas.
- Inspecter les variables locales et objets complexes.
- Explorer la pile d'appels (Call Stack).
- Développer une première méthodologie d'analyse d'un bug.

Vous travaillerez sur un projet Java multi-fichier contenant **plusieurs comportements étranges**, signalés par des `// BUG ICI ?`.

Aucun bug n'est garanti : c'est à vous **d'observer**, **de reproduire**, **d'isoler**, et **d'expliquer**.

2 Préparation de l'environnement

1. Créez un nouveau dossier pour le TP :

```
mkdir tp-debug-1
cd tp-debug-1
```

2. Initialisez un dépôt Git :

```
git init
```

3. Créez la structure suivante :

```
src/
 ├── Main.java
 ├── Calculator.java
```

```
└── TextTools.java
```

4. Ouvrez VSCode :

```
code .
```

5. Installez les extensions recommandées :

— **Extension Pack for Java**
— **Debugger for Java**

3 Code fourni

Copiez les fichiers ci-dessous dans `src/`.

3.1 Main.java

```java
public class Main {
    public static void main(String[] args) {
        System.out.println("=== TP Débogage - Partie 1 ===");

        Calculator calc = new Calculator();
        TextTools tools = new TextTools();

        int a = 12;
        int b = 4;

        int resultAdd = calc.add(a, b);      // BUG ICI ?
        int resultMul = calc.multiply(a, b); // BUG ICI ?
        int resultDiv = calc.divide(a, b);   // BUG ICI ?

        System.out.println("Addition : " + resultAdd);
        System.out.println("Multiplication : " + resultMul);
        System.out.println("Division : " + resultDiv);

        String phrase = "Bonjour tout le monde";
        String reversed = tools.reverse(phrase);   // BUG ICI ?
        String upper = tools.toUpper(phrase);      // BUG ICI ?

        System.out.println("Phrase inversée : " + reversed);
        System.out.println("Majuscules : " + upper);

        System.out.println("Programme terminé.");
    }
}
```

3.2 Calculator.java

```java
public class Calculator {

    public int add(int x, int y) {
```

```java
        int r = x + y;
        // BUG ICI ?
        return r;
    }

    public int multiply(int x, int y) {
        int r = 0;

        for (int i = 0; i <= y; i++) {  // BUG ICI ?
            r += x;
        }

        return r;
    }

    public int divide(int x, int y) {
        // BUG ICI ?
        int r = x / y;
        return r;
    }
}
```

3.3 TextTools.java

```java
public class TextTools {

    public String reverse(String input) {
        String out = "";

        for (int i = 0; i < input.length(); i++) { // BUG ICI ?
            out = input.charAt(i) + out;
        }

        return out;
    }

    public String toUpper(String input) {
        String result = "";

        for (char c : input.toCharArray()) {
            // BUG ICI ?
            result += Character.toUpperCase(c);
        }

        return result;
    }
}
```

4 Étapes du TP

4.1 Étape 1 : Exécution simple

— Compilez et exécutez le programme :

```
javac src/*.java
java -cp src Main
```

— Notez les résultats obtenus *sans débogage*.

4.2 Étape 2 : Breakpoints (obligatoire)

— Ajoutez un breakpoint dans **chaque méthode** du projet.
— Relancez en mode débogage.
— Observez **les valeurs des variables locales**.

4.3 Étape 3 : Pas-à-pas

— Exécutez ligne par ligne :

  — `Step Over`
  — `Step Into`
  — `Step Out`

Notez ce qui vous surprend.

4.4 Étape 4 : Pile d'appels

— Ouvrez le panneau "Call Stack".
— Repérez :

  — où vous êtes,
  — qui a appelé quoi,
  — dans quel ordre.

4.5 Étape 5 : Rapport court à rendre

Pour chaque méthode :

1. Décrire le comportement observé.
2. Identifier si un bug existe ou non.
3. Indiquer comment le débogueur a aidé.
4. Proposer une ou plusieurs corrections (sans les appliquer dans Git).