CORRIGÉ · JAVA

P1C5 - Lisez un code Java existant, manipulez des données et gérez des erreurs simples

Code final :

```
import java.nio.file.Files;
import java.nio.file.Path;
import java.io.IOException;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        try {
            List<String> lines = Files.readAllLines(Path.of("menu.csv"));

            for (String line : lines) {
                String[] parts = line.split(",");
                String name = parts[0];
                String price = parts[1];
                System.out.println(name + " — " + price + " €");
            }
        } catch (IOException error) {
            System.out.println("Menu introuvable");
        }
    }
}
```

Sortie attendue (fichier présent) :

```
Pizza — 12 €
Salade — 8 €
Burger — 10 €
Tiramisu — 6 €
```

Sortie si le fichier est absent :

```
Menu introuvable
```

Le `try / catch` entoure le code qui peut échouer (ici, la lecture du fichier). Si `Files.readAllLines` ne trouve pas le fichier, Java lève une `IOException` et exécute directement le bloc `catch` au lieu d'afficher une erreur brute au terminal. Le `split(",")` transforme chaque ligne en tableau de textes ; ici, on garde le prix sous forme de `String` puisqu'on ne fait que l'afficher — pas besoin de le convertir en nombre.

> 💡 Dans le projet, vous lirez souvent des données depuis un fichier ou une source externe. Le motif try lecture / catch message clair évite qu'une erreur technique interrompe brutalement l'application côté utilisateur.