Oussama Kraiem
Compte Rendu Tp Java

I. Titre de la Session : Exploration et Mise en Œuvre de la Sérialisation d'Objets Java

I.1. Objectif de la Session : Cette session visait à approfondir la compréhension et l'application des principes de sérialisation d'objets Java. L'objectif était de savoir comment stocker des objets Java dans un fichier pour une utilisation ultérieure.

I.2. Contexte de la Session : La manipulation et la sérialisation d'objets Java sont des compétences fondamentales dans le développement logiciel. La sérialisation permet de convertir un objet Java en un flux d'octets, facilitant son stockage dans un fichier ou sa transmission sur le réseau. L'accent a été mis sur la sérialisation d'objets **`Employee`** afin de comprendre pleinement ce processus.

II. Solution Proposée pour la Sérialisation des Objets Java : La solution proposée implique plusieurs étapes cruciales :

II.1. Définition de la Classe Employee : La classe **`Employee`** a été créée, implémentant l'interface **`Serializable`** pour signaler qu'elle est prête à être sérialisée. Elle contient des attributs tels que l'identifiant, le prénom et le nom de l'employé.

II.2. Méthode addToObjFile : Cette méthode prend en entrée les informations d'un employé sous forme de chaîne de caractères, ainsi qu'un délimiteur pour les séparer. Elle instancie ensuite un nouvel objet **`Employee`** avec les informations fournies, puis le sérialise en le stockant dans un fichier.

II.3. Validation des Entrées : Avant la sérialisation, un contrôle est effectué pour vérifier la validité des informations fournies. Si le nombre de jetons n'est pas conforme (trois sont attendus), un message d'erreur est affiché, signalant un format d'entrée incorrect.

III. Programmation :

```
import java.io.*;
import java.util.Locale;

class Employee implements Serializable {
    private static final long serialVersionUID = 1L;
    private String id;
    private String firstName;
    private String lastName;
```

```
    public Employee(String id, String firstName, String lastName) {
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
    }
}

public class AddToObjFile {
    public static void addToObjFile(String employeeInfo, String
delimiter, String filename) {
        try {
            String[] tokens = employeeInfo.split(delimiter);
            if (tokens.length != 3) {
                System.out.println("Erreur : Format d'entrée invalide.
Trois jetons attendus.");
                return;
            }

            Employee employee = new Employee(tokens[0], tokens[1],
tokens[2]);

            FileOutputStream fileOutputStream = new
FileOutputStream(filename, true);
            ObjectOutputStream objectOutputStream = new
ObjectOutputStream(fileOutputStream);

            objectOutputStream.writeObject(employee);
            System.out.println("Objet Employé sérialisé et ajouté au
fichier avec succès.");

            objectOutputStream.close();
            fileOutputStream.close();
        } catch (IOException e) {
            System.out.println("Erreur lors de l'écriture de l'objet
dans le fichier : " + e.getMessage());
        }
    }
```

```
    public static void main(String[] args) {
        if (args.length != 3) {
            System.out.println("Utilisation : java AddToObjFile
<informations_employé> <délimiteur> <nom_fichier>");
            System.exit(1);
        }

        addToObjFile(args[0], args[1], args[2]);
    }
}
```

IV. Testing et Validation : Pour assurer le bon fonctionnement de la solution, des tests variés peuvent être réalisés en fournissant différentes entrées pour vérifier la sérialisation correcte des objets **`Employee`**. Des tests supplémentaires peuvent être effectués pour confirmer la désérialisation et la récupération réussie des objets à partir du fichier.

Ce document présente une synthèse de la session pratique sur la sérialisation d'objets Java, en mettant en lumière les étapes essentielles et le code impliqué dans le processus.