Pour résoudre cette tâche, nous allons développer une application en Java qui permet de gérer un répertoire téléphonique. Ce répertoire va permettre à l'utilisateur de rechercher des contacts par nom ou numéro de téléphone, tout en prenant en compte différentes catégories de numéros de téléphone comme MOBILE, BUREAU et DOMICILE.

## Structure du programme

Le programme sera organisé autour des classes suivantes :

1. **Contact** : Représente une entrée dans le répertoire avec les informations sur le prénom, nom, numéros de téléphone et leurs catégories.
2. **Categorie** : Une énumération représentant les différentes catégories de téléphone : MOBILE, BUREAU, DOMICILE.
3. **Directory** : Une classe qui gère l'ajout, la recherche, et la gestion des contacts.
4. **PhoneBookApp** : L'interface principale qui permet de demander des actions à l'utilisateur et d'afficher les résultats.

Nous allons également implémenter la gestion des erreurs et valider le format des données d'entrée provenant du fichier texte.

## Code Java

Voici le code de base pour l'application :

### 1. Enumération Categorie.java

```
public enum Categorie {
    MOBILE, BUREAU, DOMICILE;
}
```

### 2. Classe Contact.java

```
import java.util.ArrayList;
import java.util.List;

public class Contact {
    private String prenom;
    private String nom;
    private List<Telephone> telephones;

    public Contact(String prenom, String nom) {
        this.prenom = prenom;
        this.nom = nom;
        this.telephones = new ArrayList<>();
    }

    public void ajouterTelephone(String numero, Categorie categorie) {
        if (telephones.size() < 5) {
            telephones.add(new Telephone(numero, categorie));
        } else {
```

```
            System.out.println("Le contact a déjà 5 numéros.");
        }
    }

    public String getPrenom() {
        return prenom;
    }

    public String getNom() {
        return nom;
    }

    public List<Telephone> getTelephones() {
        return telephones;
    }
}
```

### 3. Classe Telephone.java

```
public class Telephone {
    private String numero;
    private Categorie categorie;

    public Telephone(String numero, Categorie categorie) {
        this.numero = numero;
        this.categorie = categorie;
    }

    public String getNumero() {
        return numero;
    }

    public Categorie getCategorie() {
        return categorie;
    }
}
```

### 4. Classe Directory.java

```
import java.util.ArrayList;
import java.util.List;

public class Directory {
    private List<Contact> contacts;

    public Directory() {
        this.contacts = new ArrayList<>();
    }

    public void ajouterContact(Contact contact) {
        contacts.add(contact);
    }

    public List<Contact> rechercherParNom(String searchTerm) {
        List<Contact> resultats = new ArrayList<>();
```

```
        for (Contact contact : contacts) {
            if
(contact.getPrenom().toLowerCase().contains(searchTerm.toLowerCase()) ||

contact.getNom().toLowerCase().contains(searchTerm.toLowerCase())) {
                resultats.add(contact);
            }
        }
        return resultats;
    }

    public Contact rechercherParNumero(String numero) {
        for (Contact contact : contacts) {
            for (Telephone tel : contact.getTelephones()) {
                if (tel.getNumero().equals(numero)) {
                    return contact;
                }
            }
        }
        return null;  // Si aucun contact n'est trouvé
    }
}
```

### 5. Classe PhoneBookApp.java

```
import java.io.*;
import java.util.List;
import java.util.Scanner;

public class PhoneBookApp {

    public static void main(String[] args) {
        Directory directory = new Directory();
        chargerDonnees(directory);

        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("Choisissez votre action :");
            System.out.println("1 : Rechercher par nom");
            System.out.println("2 : Recherche par numéro");
            System.out.println("3 : Quitter");
            String choix = scanner.nextLine();

            if (choix.equals("1")) {
                System.out.println("Entrez le prénom ou le nom :");
                String nomRecherche = scanner.nextLine();
                List<Contact> resultats =
directory.rechercherParNom(nomRecherche);
                if (resultats.isEmpty()) {
                    System.out.println("Aucun contact trouvé.");
                } else {
                    for (Contact c : resultats) {
                        System.out.println(c.getPrenom() + " " + c.getNom());
                    }
                }
```

```
            } else if (choix.equals("2")) {
                System.out.println("Entrez le numéro de téléphone :");
                String numeroRecherche = scanner.nextLine();
                Contact contact =
directory.rechercherParNumero(numeroRecherche);
                if (contact == null) {
                    System.out.println("Numéro introuvable.");
                } else {
                    System.out.println(contact.getPrenom() + " " +
contact.getNom());
                }
            } else if (choix.equals("3")) {
                System.out.println("Au revoir !");
                break;
            } else {
                System.out.println("Choix invalide, réessayez.");
            }
        }

        scanner.close();
    }

    private static void chargerDonnees(Directory directory) {
        try (BufferedReader br = new BufferedReader(new
FileReader("data.txt"))) {
            String ligne;
            while ((ligne = br.readLine()) != null) {
                if (ligne.trim().isEmpty()) continue; // ignorer les lignes
vides

                // Lire le prénom et le nom
                String prenom = ligne.split(":")[1].trim();
                String nom = br.readLine().split(":")[1].trim();

                Contact contact = new Contact(prenom, nom);

                // Lire les numéros de téléphone
                while ((ligne = br.readLine()) != null &&
!ligne.trim().equals("---")) {
                    if (ligne.trim().isEmpty()) continue;
                    String numero = ligne.split(":")[1].trim();
                    Categorie categorie =
Categorie.valueOf(br.readLine().trim());

                    contact.ajouterTelephone(numero, categorie);
                }

                directory.ajouterContact(contact);
            }
        } catch (IOException e) {
            System.out.println("Erreur lors de la lecture du fichier.");
            e.printStackTrace();
        }
    }
}
```

## 6. Gestion des erreurs

- Nous avons implémenté une validation de base pour éviter l'ajout de plus de 5 numéros par contact.
- Le programme gère les erreurs lorsque l'utilisateur entre une action invalide.
- En cas de problème lors de la lecture du fichier `data.txt`, une exception est lancée.

## 7. Améliorations possibles

- Utilisation d'un `HashMap` pour stocker les contacts par numéro de téléphone, ce qui permettrait d'améliorer la recherche par numéro de téléphone.
- Ajouter une fonctionnalité de mise à jour ou de suppression des contacts.
- Gérer la saisie d'erreurs plus avancée (ex. : format de numéro de téléphone invalide).

## Conclusion

Le code ci-dessus constitue une version de base de l'application de répertoire téléphonique. Il peut être amélioré pour gérer les erreurs plus en détail et pour optimiser les recherches grâce à des structures de données plus efficaces.