```
import java.util.*;


public class RecipeManagementSystem {

   private static List<User> users = new ArrayList<>();

   private static List<Recipe> recipes = new ArrayList<>();

   private static User currentUser = null;


   public static void main(String[] args) {

      Scanner scanner = new Scanner(System.in);


      // pre-define two recipes for regular users (non-editable)

      recipes.add(new MainDishRecipe("Spaghetti Carbonara", "Spaghetti, Eggs, Bacon,
Parmesan",

            "Boil pasta, cook bacon, mix all with eggs and cheese.", "System"));

      recipes.add(new DessertRecipe("Chocolate Cake", "Flour, Sugar, Cocoa Powder, Eggs",

            "Mix ingredients and bake at 350°F.", "System"));


      while (true) {

         System.out.println("\n--- Recipe Management System ---");

         if (currentUser == null) {

            System.out.println("1. Register");

            System.out.println("2. Login");

            System.out.println("3. Exit");

         } else {

            System.out.println("1. Add Recipe");
```

```
    System.out.println("2. View Recipes");
    System.out.println("3. Logout");
}
System.out.print("Choose an option: ");
int choice = scanner.nextInt();
scanner.nextLine(); // Consume newline

if (currentUser == null) {
    switch (choice) {
        case 1:
            register(scanner);
            break;
        case 2:
            login(scanner);
            break;
        case 3:
            System.out.println("Goodbye!");
            return;
        default:
            System.out.println("Invalid choice. Try again.");
            break;
    }
} else {
    switch (choice) {
        case 1:
            addRecipe(scanner);
```

```
                break;
            case 2:
                viewRecipes(scanner);
                break;
            case 3:
                currentUser = null;
                System.out.println("Logged out.");
                break;
            default:
                System.out.println("Invalid choice. Try again.");
                break;
          }
      }
  }
}

// user registration
private static void register(Scanner scanner) {
  System.out.print("Enter username: ");
  String username = scanner.nextLine();
  System.out.print("Enter password: ");
  String password = scanner.nextLine();
  System.out.print("Enter role (Regular/Premium): ");
  String role = scanner.nextLine();

  User user;
```

```
    if (role.equalsIgnoreCase("Premium")) {
        user = new PremiumUser(username, password);
    } else {
        user = new RegularUser(username, password);
    }
    users.add(user);
    System.out.println("Registration successful!");
}

// user login
private static void login(Scanner scanner) {
    System.out.print("Enter username: ");
    String username = scanner.nextLine();
    System.out.print("Enter password: ");
    String password = scanner.nextLine();

    for (User user : users) {
        if (user.getUsername().equals(username) && user.getPassword().equals(password)) {
            currentUser = user;
            System.out.println("Login successful! Welcome, " + user.getUsername());
            return;
        }
    }
    System.out.println("Invalid credentials. Try again.");
}
```

```
    // add recipe only available for Premium users
    private static void addRecipe(Scanner scanner) {
        if (currentUser instanceof RegularUser) {
            System.out.println("Regular users cannot add or modify recipes.");
            return;
        }

        System.out.print("Enter recipe type (MainDish/Dessert/Beverage): ");
        String type = scanner.nextLine();

        System.out.print("Enter recipe title: ");
        String title = scanner.nextLine();
        System.out.print("Enter ingredients: ");
        String ingredients = scanner.nextLine();
        System.out.print("Enter instructions: ");
        String instructions = scanner.nextLine();

        Recipe recipe;
        switch (type.toLowerCase()) {
            case "dessert":
                recipe = new DessertRecipe(title, ingredients, instructions,
currentUser.getUsername());
                break;
            case "beverage":
                recipe = new BeverageRecipe(title, ingredients, instructions,
currentUser.getUsername());
                break;
        default:
          recipe = new MainDishRecipe(title, ingredients, instructions,
currentUser.getUsername());
          break;
      }

      recipes.add(recipe);
      System.out.println("Recipe added successfully!");
    }

    // view recipes
    private static void viewRecipes(Scanner scanner) {
      System.out.println("\n--- Recipes ---");
      for (int i = 0; i < recipes.size(); i++) {
        Recipe recipe = recipes.get(i);
        if (currentUser instanceof RegularUser) {
          // Regular users can view, but not edit predefined recipes
          if (recipe.getAuthor().equals("System")) {
            System.out.println((i + 1) + ". " + recipe.getTitle() + " (" +
recipe.getClass().getSimpleName() + ")");
          }
        } else {
          // premium users can view all recipes
          System.out.println((i + 1) + ". " + recipe.getTitle() + " (" +
recipe.getClass().getSimpleName() + ")");
        }
      }
```

```
        System.out.print("Enter recipe number to view details or 0 to go back: ");
        int choice = scanner.nextInt();
        scanner.nextLine();

        if (choice > 0 && choice <= recipes.size()) {
            Recipe recipe = recipes.get(choice - 1);
            System.out.println("\nTitle: " + recipe.getTitle());
            System.out.println("Ingredients: " + recipe.getIngredients());
            System.out.println("Instructions: " + recipe.getInstructions());
            System.out.println("Comments:");
            for (String comment : recipe.getComments()) {
                System.out.println("- " + comment);
            }

            System.out.print("Add a comment (leave blank to skip): ");
            String comment = scanner.nextLine();
            if (!comment.isBlank()) {
                recipe.addComment(currentUser.getUsername() + ": " + comment);
                System.out.println("Comment added!");
            }
        }
    }
}
```

```
// base user class
class User {
   private String username;
   private String password;

   public User(String username, String password) {
      this.username = username;
      this.password = password;
   }

   public String getUsername() {
      return username;
   }

   public String getPassword() {
      return password;
   }
}

// regularUser subclass
class RegularUser extends User {
   public RegularUser(String username, String password) {
      super(username, password);
   }
}
```

```
// premiumUser subclass
class PremiumUser extends User {
    public PremiumUser(String username, String password) {
        super(username, password);
    }
}

// base recipe class
class Recipe {
    private String title;
    private String ingredients;
    private String instructions;
    private boolean isSpecial;
    private String author;
    private List<String> comments = new ArrayList<>();

    public Recipe(String title, String ingredients, String instructions, String author) {
        this.title = title;
        this.ingredients = ingredients;
        this.instructions = instructions;
        this.author = author;
    }

    public String getTitle() {
        return title;
    }
```

```
public String getIngredients() {
  return ingredients;
}

public String getInstructions() {
  return instructions;
}

public boolean isSpecial() {
  return isSpecial;
}

public void setSpecial(boolean special) {
  isSpecial = special;
}

public String getAuthor() {
  return author;
}

public List<String> getComments() {
  return comments;
}

public void addComment(String comment) {
        comments.add(comment);
    }
}

// MainDishRecipe Subclass
class MainDishRecipe extends Recipe {
    public MainDishRecipe(String title, String ingredients, String instructions, String author) {
        super(title, ingredients, instructions, author);
    }
}

// DessertRecipe Subclass
class DessertRecipe extends Recipe {
    public DessertRecipe(String title, String ingredients, String instructions, String author) {
        super(title, ingredients, instructions, author);
    }
}

// BeverageRecipe Subclass
class BeverageRecipe extends Recipe {
    public BeverageRecipe(String title, String ingredients, String instructions, String author) {
        super(title, ingredients, instructions, author);
    }
}
```