SC2002

SDDA : GROUP 3

AU YI TENG, CHUA WEN JUN, HASINI LAGATAPATHI, KIERAN MAK, RAJKUMAR SAANVI

DESIGN CONSIDERATIONS

Overview

- Built using Model-View-Controller (MVC) for separation of concerns.
- Model layer: User, Project, Application, Enquiry classes.
- View layer: CLI interfaces; Controller logic in utility/menu handlers.
- Design guided by OOP and SOLID principles for modularity.
- Abstract User class and polymorphism used for role management.
- Interfaces and enums not used, but system is designed to support them.

DESIGN CONSIDERATIONS

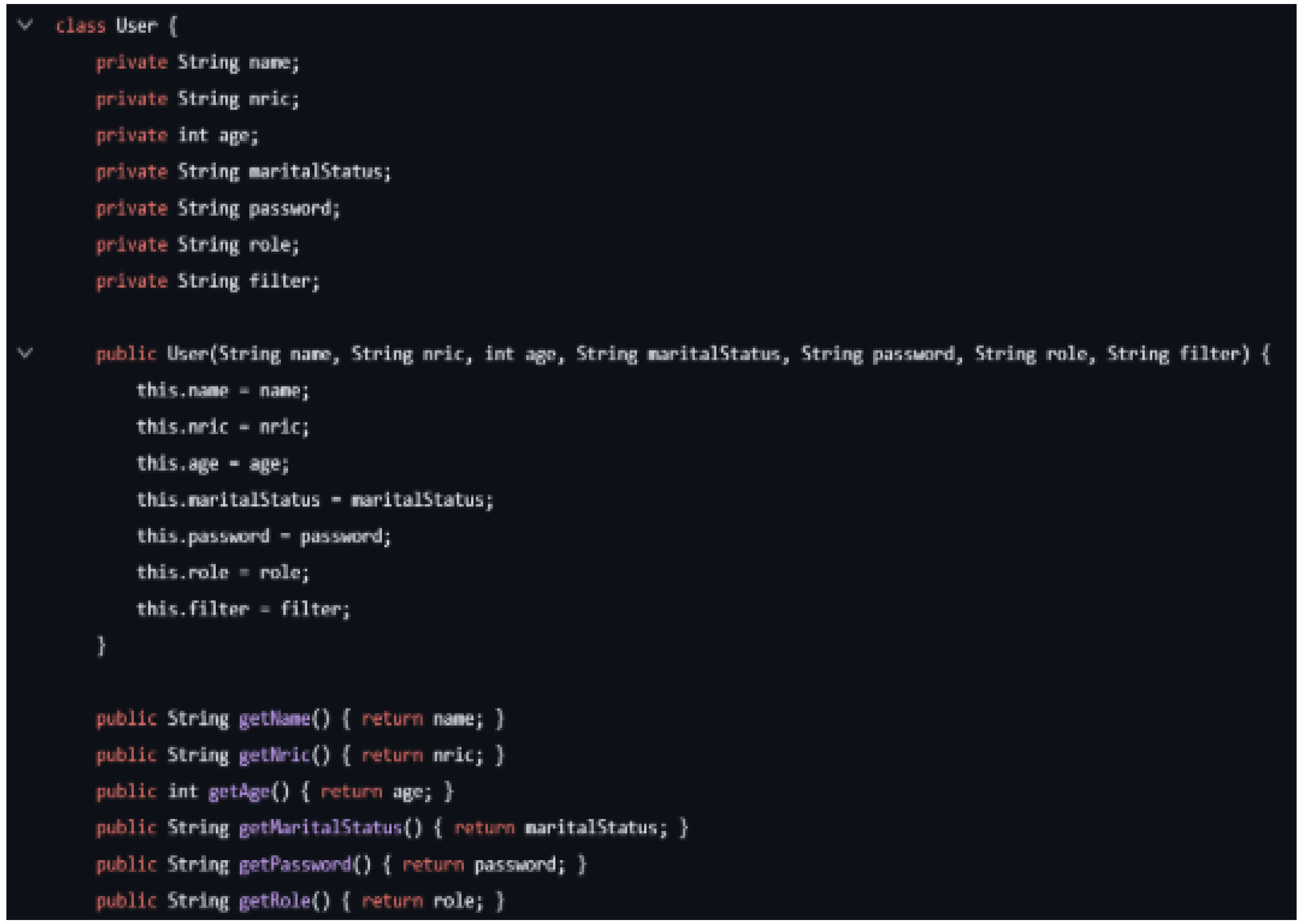
Assumptions made

- Users initialized from .csv files with default passwords.
- Users understand role-specific features (minimal UI guidance).
- Officers/Managers can only register if not already assigned.
- Sequential processing assumed (no concurrency).
- Visibility toggling affects listings, not access to prior applications.

DESIGN CONSIDERATIONS

Object-oriented programming principles

Abstraction :

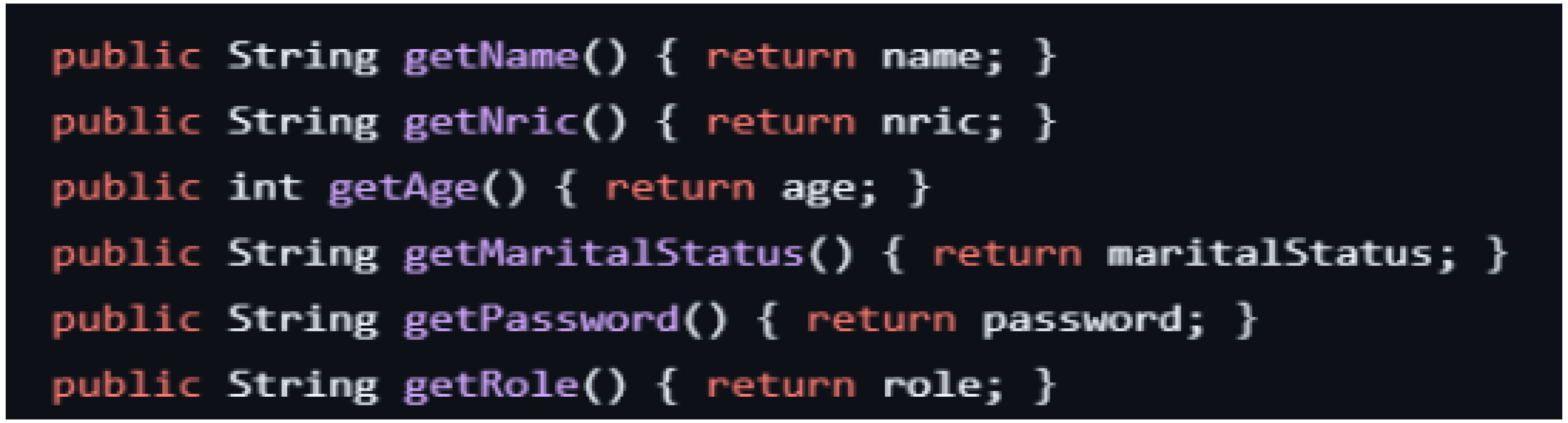


```
class User {
    private String name;
    private String nric;
    private int age;
    private String maritalStatus;
    private String password;
    private String role;
    private String filter;

    public User(String name, String nric, int age, String maritalStatus, String password, String role, String filter) {
        this.name = name;
        this.nric = nric;
        this.age = age;
        this.maritalStatus = maritalStatus;
        this.password = password;
        this.role = role;
        this.filter = filter;
    }

    public String getName() { return name; }
    public String getNric() { return nric; }
    public int getAge() { return age; }
    public String getMaritalStatus() { return maritalStatus; }
    public String getPassword() { return password; }
    public String getRole() { return role; }
```

DESIGN CONSIDERATIONS

Object-oriented programming principles

Encapsulation :

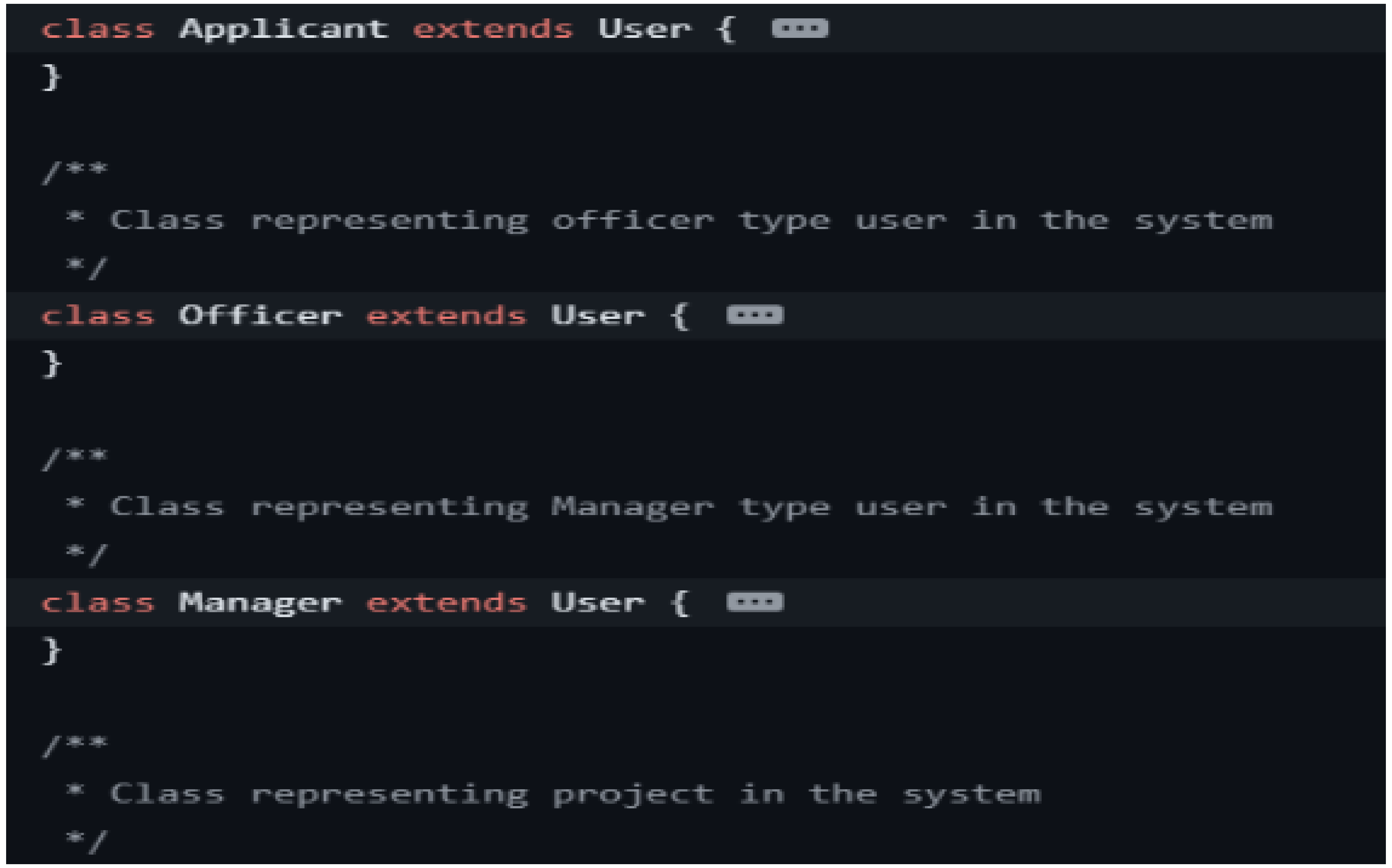


```java
public String getName() { return name; }
public String getNric() { return nric; }
public int getAge() { return age; }
public String getMaritalStatus() { return maritalStatus; }
public String getPassword() { return password; }
public String getRole() { return role; }
```

DESIGN CONSIDERATIONS

Object-oriented programming principles

Inheritance :

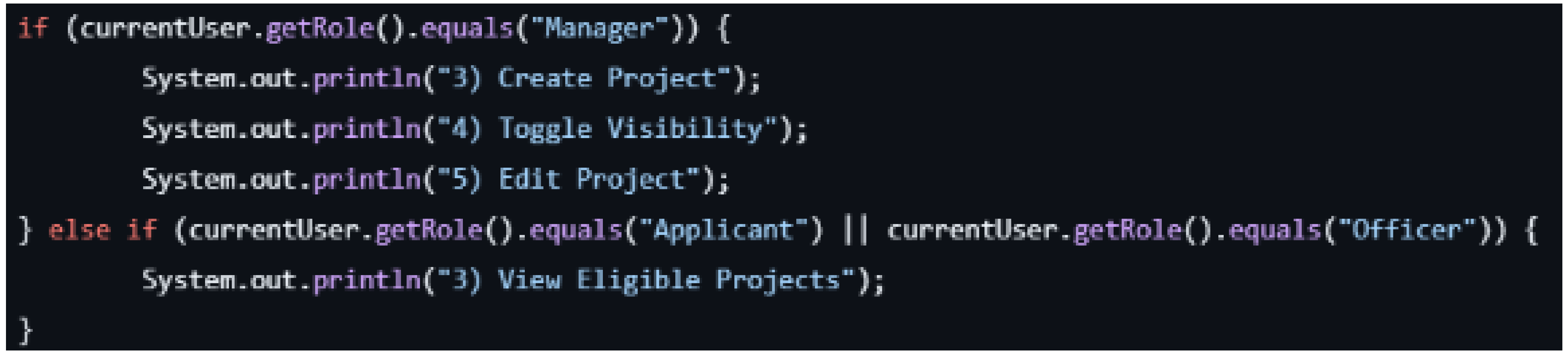


```
class Applicant extends User { ...
}

/**
 * Class representing officer type user in the system
 */
class Officer extends User { ...
}

/**
 * Class representing Manager type user in the system
 */
class Manager extends User { ...
}

/**
 * Class representing project in the system
 */
```

DESIGN CONSIDERATIONS

Object-Oriented Programming principles

Polymorphism :



```
if (currentUser.getRole().equals("Manager")) {
        System.out.println("3) Create Project");
        System.out.println("4) Toggle Visibility");
        System.out.println("5) Edit Project");
} else if (currentUser.getRole().equals("Applicant") || currentUser.getRole().equals("Officer")) {
        System.out.println("3) View Eligible Projects");
}
```

DESIGN CONSIDERATIONS

Application of SOLID principles

- Single Responsibility Principle (SRP)
- Open-Closed Principle (OCP)
- Liskov Substitution Principle (LSP)