

**Nama: Rifki Abiyan**

**NIM: 2309106030**

**Kelas: A2 2023**

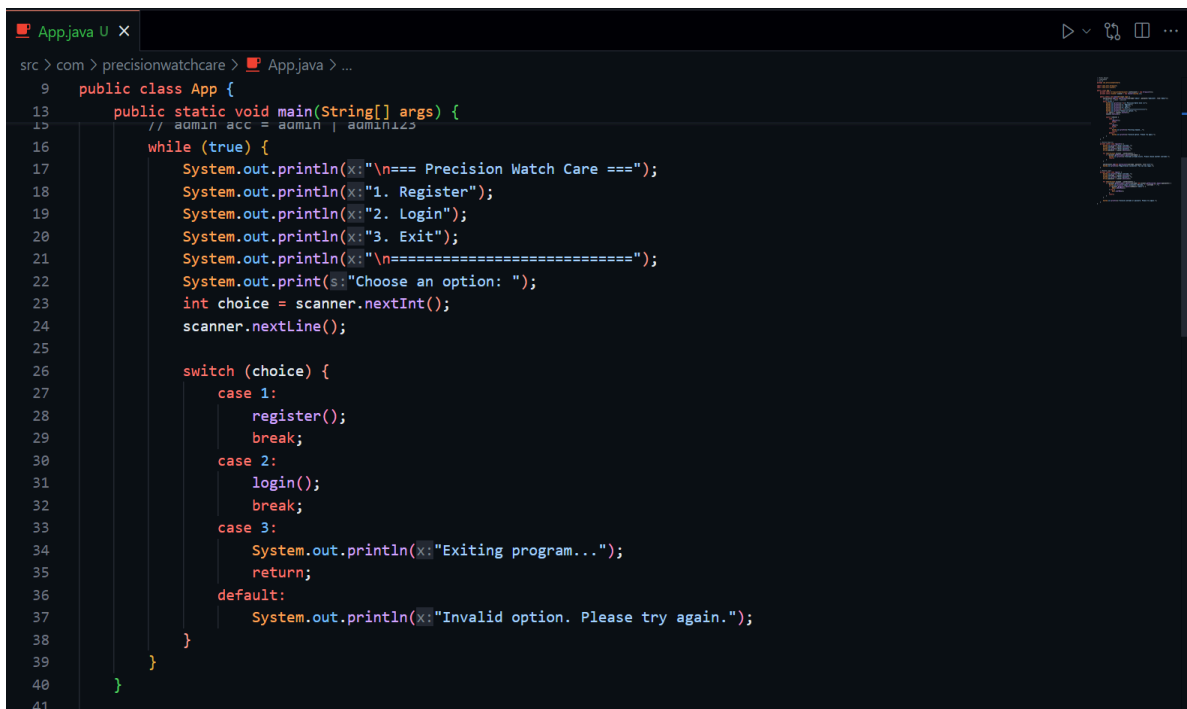
---

## **Pemrograman Berbasis Objek**

### **POSTTEST 4**

#### **APP.JAVA**

---



```
src > com > precisionwatchcare > App.java > ...
9  public class App {
13     public static void main(String[] args) {
14         // admin acc = admin | admin123
15
16         while (true) {
17             System.out.println(x: "\n=== Precision Watch Care ===");
18             System.out.println(x: "1. Register");
19             System.out.println(x: "2. Login");
20             System.out.println(x: "3. Exit");
21             System.out.println(x: "\n=====");
22             System.out.print(s: "Choose an option: ");
23             int choice = scanner.nextInt();
24             scanner.nextLine();
25
26             switch (choice) {
27                 case 1:
28                     register();
29                     break;
30                 case 2:
31                     login();
32                     break;
33                 case 3:
34                     System.out.println(x: "Exiting program...");
35                     return;
36                 default:
37                     System.out.println(x: "Invalid option. Please try again.");
38             }
39         }
40     }
41 }
```

```

9 public class App {
41
42     // Method Register
43     private static void register() {
44         System.out.print(s:"Enter username: ");
45         String username = scanner.nextLine();
46         System.out.print(s:"Enter password: ");
47         String password = scanner.nextLine();
48
49         for (UserAccount account : userAccounts) {
50             if (account.getUsername().equals(username)) {
51                 System.out.println(x:"Username already exists. Please choose another username.");
52                 return;
53             }
54         }
55
56         userAccounts.add(new UserAccount(username, password, role:"user"));
57         System.out.println(x:"Registration successful! You can now login.");
58     }
59
60     // Method Login
61     private static void login() {
62         System.out.print(s:"Enter username: ");
63         String username = scanner.nextLine();
64         System.out.print(s:"Enter password: ");
65         String password = scanner.nextLine();
66
67         for (UserAccount account : userAccounts) {
68             if (account.getUsername().equals(username) && account.getPassword().equals(password)) {
69                 System.out.println("Login successful! Welcome. " + username + ".");
70
71                 if (account.getRole().equals(anObject:"admin")) {
72                     Admin.adminMenu();
73                 } else {
74                     User.userMenu();
75                 }
76                 return;
77             }
78
79             System.out.println(x:"Invalid username or password. Please try again.");
80         }
81     }

```

## Admin.java

```

4 package com.precisionwatchcare;
5
6 import java.util.ArrayList;
7 import java.util.Scanner;
8 // admin acc = admin | admin123
9 public class Admin {
10     private static Scanner scanner = new Scanner(System.in);
11
12     public static void adminMenu() {
13         while (true) {
14             System.out.println(x:"\n==== Admin Menu =====");
15             System.out.println(x:"1. View All Services");
16             System.out.println(x:"2. Estimate Service Cost");
17             System.out.println(x:"3. Update Service Status");
18             System.out.println(x:"4. Return Watch");
19             System.out.println(x:"5. Logout");
20             System.out.println(x:"\n=====");
21             System.out.print(s:"Choose an option: ");
22             int choice = scanner.nextInt();
23             scanner.nextLine();
24
25             switch (choice) {
26                 case 1:
27                     viewAllServices();
28                     break;
29                 case 2:
30                     estimateServiceCost();
31                     break;
32                 case 3:
33                     updateServiceStatus();

```

```

35         case 4:
36             returnWatch();
37             break;
38         case 5:
39             System.out.println(x:"Logging out...");
40             return;
41     default:
42         System.out.println(x:"Invalid option. Please try again.");
43     }
44 }
45 }
46
47 // Method Service
48 private static void viewAllServices() {
49     ArrayList<Service> serviceList = User.getServiceList();
50     if (serviceList.isEmpty()) {
51         System.out.println(x:"No services available.");
52     } else {
53         for (Service service : serviceList) {
54             System.out.println(service);
55         }
56     }
57 }

```

```

60 private static void estimateServiceCost() {
61     viewAllServices();
62     if (!User.getServiceList().isEmpty()) {
63         System.out.print(s:"Enter the service ID to estimate cost: ");
64         String serviceId = scanner.nextLine();
65         System.out.print(s:"Enter estimated cost: Rp");
66         double cost = scanner.nextDouble();
67         scanner.nextLine();
68
69         for (Service service : User.getServiceList()) {
70             if (service.getServiceId().equals(serviceId)) {
71                 service.setCost(cost);
72                 System.out.println(x:"Cost estimated successfully!");
73                 return;
74             }
75         }
76         System.out.println(x:"Service ID not found.");
77     }
78 }

```

```

81     private static void updateServiceStatus() {
82         viewAllServices();
83         if (!User.getServiceList().isEmpty()) {
84             System.out.print(s: "Enter the service ID to update status: ");
85             String serviceId = scanner.nextLine();
86             System.out.print(s: "Enter new status (Pending/In Progress/Completed): ");
87             String status = scanner.nextLine();
88
89             for (Service service : User.getServiceList()) {
90                 if (service.getServiceId().equals(serviceId)) {
91                     service.setStatus(status);
92                     System.out.println(x: "Status updated successfully!");
93                     return;
94                 }
95             }
96             System.out.println(x: "Service ID not found.");
97         }
98     }

```

```

101     private static void returnWatch() {
102         viewAllServices();
103         if (!User.getServiceList().isEmpty()) {
104             System.out.print(s: "Enter the service ID to return watch: ");
105             String serviceId = scanner.nextLine();
106
107             for (Service service : User.getServiceList()) {
108                 if (service.getServiceId().equals(serviceId)) {
109                     if (service.getStatus().equals(anObject: "Completed")) {
110                         System.out.println(x: "Watch returned successfully!");
111                         User.getServiceList().remove(service);
112                     } else {
113                         System.out.println(x: "Service is not yet completed.");
114                     }
115                     return;
116                 }
117             }
118             System.out.println(x: "Service ID not found.");
119         }
120     }
121 }

```

## Service.java

```
4 package com.precisionwatchcare;
5
6 public class Service {
7     private String serviceId;
8     private String serviceType;
9     private double cost;
10    private String status;
11    private boolean useCourier;
12
13    public Service(String serviceId, String serviceType, double cost, boolean useCourier) {
14        this.serviceId = serviceId;
15        this.serviceType = serviceType;
16        this.cost = cost;
17        this.status = "Pending"; // Default
18        this.useCourier = useCourier;
19    }
20
21    // Overloaded constructor - without cost parameter (default to 0)
22    public Service(String serviceId, String serviceType, boolean useCourier) {
23        this(serviceId, serviceType, cost:0, useCourier);
24    }
25
26    // Overloaded constructor - without courier parameter (default to false)
27    public Service(String serviceId, String serviceType) {
28        this(serviceId, serviceType, cost:0, useCourier:false);
29    }
```

```
30
31    public String getServiceId() {
32        return serviceId;
33    }
34
35    public void setServiceId(String serviceId) {
36        this.serviceId = serviceId;
37    }
38
39    public String getServiceType() {
40        return serviceType;
41    }
42
43    public void setServiceType(String serviceType) {
44        this.serviceType = serviceType;
45    }
46
47    public double getCost() {
48        return cost;
49    }
50
51    public void setCost(double cost) {
52        this.cost = cost;
53    }
54
55    public String getStatus() {
56        return status;
57    }
```

```

38
39     public String getServiceType() {
40         return serviceType;
41     }
42
43     public void setServiceType(String serviceType) {
44         this.serviceType = serviceType;
45     }
46
47     public double getCost() {
48         return cost;
49     }
50
51     public void setCost(double cost) {
52         this.cost = cost;
53     }
54
55     public String getStatus() {
56         return status;
57     }
58
59     public void setStatus(String status) {
60         this.status = status;
61     }

```

```

63     public boolean isUseCourier() {
64         return useCourier;
65     }
66
67     public void setUseCourier(boolean useCourier) {
68         this.useCourier = useCourier;
69     }
70
71     @Override
72     public String toString() {
73         return "Service [ID: " + serviceId + ", Type: " + serviceType +
74             ", Cost: Rp" + cost + ", Status: " + status +
75             ", Use Courier: " + useCourier + "]\n";
76     }
77 }

```

## User.java

```

4  package com.precisionwatchcare;
5
6  import java.util.ArrayList;
7  import java.util.Scanner;
8
9  public class User {
10     private static ArrayList<Watch> watchList = new ArrayList<>();
11     private static ArrayList<Service> serviceList = new ArrayList<>();
12     private static Scanner scanner = new Scanner(System.in);
13
14     public static void userMenu() {
15         while (true) {
16             System.out.println(X:"\n=== User Menu ===");
17             System.out.println(X:"1. Add Watch");
18             System.out.println(X:"2. View My Watches");
19             System.out.println(X:"3. Request Service");
20             System.out.println(X:"4. Logout");
21             System.out.println(X:"\n=====");
22             System.out.print(S:"Choose an option: ");
23             int choice = scanner.nextInt();
24             scanner.nextLine();
25
26             switch (choice) {
27                 case 1:
28                     addWatch();
29                     break;
30                 case 2:
31                     viewMyWatches();
32                     break;

```

```

33                 case 3:
34                     requestService();
35                     break;
36                 case 4:
37                     System.out.println(X:"Logging out...");
38                     return;
39                 default:
40                     System.out.println(X:"Invalid option. Please try again.");
41             }
42         }
43     }
44
45     // Method Add Watch
46     private static void addWatch() {
47         System.out.print(S:"Enter brand: ");
48         String brand = scanner.nextLine();
49         System.out.print(S:"Enter model: ");
50         String model = scanner.nextLine();
51         System.out.print(S:"Enter year: ");
52         int year = scanner.nextInt();
53         scanner.nextLine();
54         System.out.print(S:"Enter complication: ");
55         String complication = scanner.nextLine();
56         System.out.print(S:"Enter type (Analog/Digital): ");
57         String type = scanner.nextLine();

```

```

58
59         Watch watch = new Watch(brand, model, year, complication, type);
60         watchList.add(watch);
61         System.out.println(X:"Watch added successfully!");
62     }
63
64     // Method View Watch
65     private static void viewMyWatches() {
66         if (watchList.isEmpty()) {
67             System.out.println(X:"No watches available.");
68         } else {
69             for (Watch watch : watchList) {
70                 System.out.println(watch);
71             }
72         }
73     }
74

```

```

9   public class User {
76      private static void requestService() {
78          if (!watchList.isEmpty()) {
79              System.out.print(s:"Enter the index of the watch to service: ");
80              int index = scanner.nextInt();
81              scanner.nextLine();
82
83              if (index >= 0 && index < watchList.size()) {
84                  System.out.print(s:"Enter service type (Cleaning/Repair/Battery Change/Custom: ");
85                  String serviceType = scanner.nextLine();
86
87                  // Using different constructors based on user input
88                  Service service;
89                  System.out.print(s:"Use courier? (y/n): ");
90                  String courierChoice = scanner.nextLine();
91
92                  if (courierChoice.equalsIgnoreCase(anotherString:"y")) {
93                      service = new Service("SERV" + (serviceList.size() + 1), serviceType, useCourier:true);
94                  } else {
95                      service = new Service("SERV" + (serviceList.size() + 1), serviceType);
96                  }
97
98                  serviceList.add(service);
99                  System.out.println("Service requested successfully! Service ID: " + service.getServiceId());
100             } else {
101                 System.out.println(x:"Invalid index.");
102             }
103         }
104     }
106     public static ArrayList<Service> getServiceList() {
107         return serviceList;
108     }
109 }

```

## UserAccount.java

```

6   public class UserAccount {
7       private String username;
8       private String password;
9       private String role;
10
11       public UserAccount(String username, String password, String role) {
12           this.username = username;
13           this.password = password;
14           this.role = role;
15       }
16
17       // Getter dan Setter
18       public String getUsername() {
19           return username;
20       }
21
22       public String getPassword() {
23           return password;
24       }
25
26       public String getRole() {
27           return role;
28       }
29
30       @Override
31       public String toString() {
32           return "UserAccount [Username: " + username + ", Role: " + role + "];"
33       }

```



# Watch.java

```
src / com / precisionwatchcare > Watch.java > Watch
1  package com.precisionwatchcare;
2
3  public class Watch {
4      private String brand; // private
5      protected String model; // protected
6      int year; // default
7      public String complication; // public
8      private String type; // private
9
10     public Watch(String brand, String model, int year, String complication, String type) {
11         this.brand = brand;
12         this.model = model;
13         this.year = year;
14         this.complication = complication;
15         this.type = type;
16     }
17
18     // Getter dan Setter untuk private dan protected
19     public String getBrand() {
20         return brand;
21     }
22
23     public void setBrand(String brand) {
24         this.brand = brand;
25     }
26
27     public String getModel() {
28         return model;
29     }
30
31     public void setModel(String model) {
32         this.model = model;
33     }
34
35     public String getType() {
36         return type;
37     }
38
39     public void setType(String type) {
40         this.type = type;
41     }
42
43     @Override
44     public String toString() {
45         if (type.equalsIgnoreCase(anotherString:"Analog")) {
46             return "Analog Watch [Brand: " + brand + ", Model: " + model +
47                 ", Year: " + year + ", Complication: " + complication + "];"
48         } else if (type.equalsIgnoreCase(anotherString:"Digital")) {
49             return "Digital Watch [Brand: " + brand + ", Model: " + model +
50                 ", Year: " + year + ", Features: " + complication + "];"
51         }
52         return "Watch [Brand: " + brand + ", Model: " + model +
53             ", Year: " + year + ", Complication: " + complication + ", Type: " + type + "];"
54     }
55 }
```

## Cara menjalankan program

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GIT LENS java + - [ ] [ ] ... ^ x

PS C:\Main Storage\Documents\vscode programs\JAVA\posttest4_PBO> javac -d bin src/com/precisionwatchcare/*.java
❖ PS C:\Main Storage\Documents\vscode programs\JAVA\posttest4_PBO> java -cp bin com.precisionwatchcare.App

=== Precision Watch Care ===
1. Register
2. Login
3. Exit

=====
Choose an option: 2
Enter username: 
```