

**Nama: Rifki Abiyan**

**NIM: 2309106030**

**Kelas: A2 2023**

---

## **Pemrograman Berbasis Objek**

### **POSTTEST 5**

#### **WatchOperations.JAVA**

---

```
1 package com.precisionwatchcare;
2
3 public interface WatchOperations {
4     void displayWatchDetails();
5     void performMaintenance();
6 }
```

#### **App.JAVA**

---

```
App.java U x
src > com > precisionwatchcare > App.java > ...
9 public class App {
13     public static void main(String[] args) {
14         // admin acc = admin | admin123
16         while (true) {
17             System.out.println("\n=== Precision Watch Care ===");
18             System.out.println("1. Register");
19             System.out.println("2. Login");
20             System.out.println("3. Exit");
21             System.out.println("\n=====");
22             System.out.print("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("Exiting program...");
35                     return;
36                 default:
37                     System.out.println("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("\nEnter username: ");
45          String username = scanner.nextLine();
46          System.out.print("\nEnter password: ");
47          String password = scanner.nextLine();
48
49          for (UserAccount account : userAccounts) {
50              if (account.getUsername().equals(username)) {
51                  System.out.println("\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("\Registration successful! You can now login.");
58      }
59
60      // Method Login
61      private static void login() {
62          System.out.print("\nEnter username: ");
63          String username = scanner.nextLine();
64          System.out.print("\nEnter 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("admin")) {
72                      Admin.adminMenu();
73                  } else {
74                      User.userMenu();
75                  }
76                  return;
77              }
78          }
79
80          System.out.println("\Invalid username or password. Please try again.");
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("\n==== Admin Menu =====");
15             System.out.println("\t1. View All Services");
16             System.out.println("\t2. Estimate Service Cost");
17             System.out.println("\t3. Update Service Status");
18             System.out.println("\t4. Return Watch");
19             System.out.println("\t5. Logout");
20             System.out.println("\n=====");
21             System.out.print("\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();

```

```

36         case 4:
37             returnWatch();
38             break;
39         case 5:
40             System.out.println(x:"Logging out...");
41             return;
42         default:
43             System.out.println(x:"Invalid option. Please try again.");
44     }
45 }
46
47
48 // Method Service
49 private static void viewAllServices() {
50     ArrayList<Service> serviceList = User.getServiceList();
51     if (serviceList.isEmpty()) {
52         System.out.println(x:"No services available.");
53     } else {
54         for (Service service : serviceList) {
55             System.out.println(service);
56         }
57     }
58 }
59
60 // Method Service Cost
61 private static void estimateServiceCost() {
62     try {
63         viewAllServices();

```

```

64     if (!User.getServiceList().isEmpty()) {
65         System.out.print(s:"Enter the service ID to estimate cost: ");
66         String serviceId = scanner.nextLine();
67         if (serviceId.isEmpty()) {
68             throw new IllegalArgumentException(s:"Service ID cannot be empty");
69         }
70
71         System.out.print(s:"Enter estimated cost: Rp");
72         double cost = scanner.nextDouble();
73         scanner.nextLine();
74         if (cost <= 0) {
75             throw new IllegalArgumentException(s:"Cost must be positive");
76         }
77
78         boolean found = false;
79         for (Service service : User.getServiceList()) {
80             if (service.getServiceId().equals(serviceId)) {
81                 service.setCost(cost);
82                 System.out.println(x:"Cost estimated successfully!");
83                 found = true;
84                 break;
85             }
86         }
87         if (!found) {
88             System.out.println(x:"Service ID not found.");
89         }
90     }

```

```

91     } catch (InputMismatchException e) {
92         System.out.println(x:"Error: Invalid input for cost. Please enter a number.");
93         scanner.nextLine(); // Clear the invalid input
94     } catch (IllegalArgumentException e) {
95         System.out.println("Error: " + e.getMessage());
96     } catch (Exception e) {
97         System.out.println("An unexpected error occurred: " + e.getMessage());
98     }
99 }
100
101 // Method Status Update
102 private static void updateServiceStatus() {
103     viewAllServices();
104     if (!User.getServiceList().isEmpty()) {
105         System.out.print(s:"Enter the service ID to update status: ");
106         String serviceId = scanner.nextLine();
107         System.out.print(s:"Enter new status (Pending/In Progress/Completed): ");
108         String status = scanner.nextLine();
109
110         for (Service service : User.getServiceList()) {
111             if (service.getServiceId().equals(serviceId)) {
112                 service.setStatus(status);
113                 System.out.println(x:"Status updated successfully!");
114                 return;
115             }
116         }
117         System.out.println(x:"Service ID not found.");
118     }

```

```

119 }
120
121 // Method Return Watch
122 private static void returnWatch() {
123     viewAllServices();
124     if (!User.getServiceList().isEmpty()) {
125         System.out.print(s:"Enter the service ID to return watch: ");
126         String serviceId = scanner.nextLine();
127
128         for (Service service : User.getServiceList()) {
129             if (service.getServiceId().equals(serviceId)) {
130                 if (service.getStatus().equals(anObject:"Completed")) {
131                     System.out.println(x:"Watch returned successfully!");
132                     User.getServiceList().remove(service);
133                 } else {
134                     System.out.println(x:"Service is not yet completed.");
135                 }
136                 return;
137             }
138         }
139         System.out.println(x:"Service ID not found.");
140     }
141 }
142 }

```

# 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         try {
48             System.out.print(s:"Enter brand: ");
49             String brand = scanner.nextLine();
50             if (brand.isEmpty()) {
51                 throw new IllegalArgumentException(s:"Brand cannot be empty");
52             }
53
54             System.out.print(s:"Enter model: ");
55             String model = scanner.nextLine();
56             if (model.isEmpty()) {
57                 throw new IllegalArgumentException(s:"Model cannot be empty");
58             }
59
60             System.out.print(s:"Enter year: ");
```

```

61     int year = scanner.nextInt();
62     scanner.nextLine();
63     if (year < 1900 || year > java.time.Year.now().getValue()) {
64         throw new IllegalArgumentException(s:"Invalid year");
65     }
66
67     System.out.print(s:"Enter complication/features: ");
68     String complication = scanner.nextLine();
69     System.out.print(s:"Enter type (Analog/Digital): ");
70     String type = scanner.nextLine();
71
72     Watch watch;
73     if (type.equalsIgnoreCase(anotherString:"Analog")) {
74         watch = new AnalogWatch(brand, model, year, complication);
75     } else if (type.equalsIgnoreCase(anotherString:"Digital")) {
76         watch = new DigitalWatch(brand, model, year, complication);
77     } else {
78         System.out.println(x:"Invalid type. Defaulting to Analog.");
79         watch = new AnalogWatch(brand, model, year, complication);
80     }
81
82     watchList.add(watch);
83     watch.displaySpecialFeature();
84     watch.displayWatchDetails(); // Using interface method
85     watch.performMaintenance(); // Using interface method
86     System.out.println("Total watches in system: " + Watch.getTotalWatches()); // Using static method
87     System.out.println(x:"Watch added successfully!");
88 } catch (IllegalArgumentException e) {

```



```

89         System.out.println("Error: " + e.getMessage());
90     } catch (Exception e) {
91         System.out.println("An unexpected error occurred: " + e.getMessage());
92     }
93 }
94
95 // Method View Watch
96 private static void viewMyWatches() {
97     if (watchList.isEmpty()) {
98         System.out.println(x:"No watches available.");
99     } else {
100         for (Watch watch : watchList) {
101             System.out.println(watch);
102         }
103     }
104 }
105
106 private static void requestService() {
107     viewMyWatches();
108     if (!watchList.isEmpty()) {
109         System.out.print(s:"Enter the index of the watch to service: ");
110         int index = scanner.nextInt();
111         scanner.nextLine();
112
113         if (index >= 0 && index < watchList.size()) {
114             System.out.print(s:"Enter service type (Cleaning/Repair/Battery Change/Custom): ");
115             String serviceType = scanner.nextLine();
116

```

```

116
117         Service service;
118         System.out.print(s:"Use courier? (y/n): ");
119         String courierChoice = scanner.nextLine();
120
121         if (courierChoice.equalsIgnoreCase(anotherString:"y")) {
122             service = new Service("SERV" + (serviceList.size() + 1), serviceType, useCourier:true);
123         } else {
124             service = new Service("SERV" + (serviceList.size() + 1), serviceType);
125         }
126
127         serviceList.add(service);
128         System.out.println("Service requested successfully! Service ID: " + service.getServiceId());
129     } else {
130         System.out.println(x:"Invalid index.");
131     }
132 }
133
134
135 public static ArrayList<Service> getServiceList() {
136     return serviceList;
137 }
138 }

```

# UserAccount.java

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

# Watch.java

---

```
1  package com.precisionwatchcare;
2
3  public abstract class Watch implements WatchOperations {
4      private String brand;
5      protected String model;
6      final int year;
7      public String complication;
8      private String type;
9      private static int totalWatches = 0;
10
11
12      public Watch(String brand, String model, int year, String complication, String type) {
13          this.brand = brand;
14          this.model = model;
15          this.year = year;
16          this.complication = complication;
17          this.type = type;
18          totalWatches++;
19      }
20
21      // Static method to get total watches
22      public static int getTotalWatches() {
23          return totalWatches;
24      }
25
26      // Implement interface methods
27      @Override
28      public void displayWatchDetails() {
29          System.out.println("Brand: " + brand + ", Model: " + model + ", Year: " + year);
30      }
```

```
31      @Override
32      public void performMaintenance() {
33          System.out.println("Performing basic maintenance for " + brand + " " + model);
34      }
35
36      // Added abstract method
37      public abstract void displaySpecialFeature();
38
39      // Getter dan Setter
40      public final String getBrand() {
41          return brand;
42      }
43
44      public void setBrand(String brand) {
45          this.brand = brand;
46      }
47
48      public String getModel() {
49          return model;
50      }
51
52      public void setModel(String model) {
53          this.model = model;
54      }
55
56      public String getType() {
57          return type;
58      }
```

```

60     public void setType(String type) {
61         this.type = type;
62     }
63
64     public String getComplication() {
65         return complication;
66     }
67
68     @Override
69     public String toString() {
70         displaySpecialFeature();
71         if (type.equalsIgnoreCase(anotherString:"Analog")) {
72             return "Analog Watch [Brand: " + brand + ", Model: " + model +
73                 ", Year: " + year + ", Complication: " + complication + "];";
74         } else if (type.equalsIgnoreCase(anotherString:"Digital")) {
75             return "Digital Watch [Brand: " + brand + ", Model: " + model +
76                 ", Year: " + year + ", Features: " + complication + "];";
77         }
78         return "Watch [Brand: " + brand + ", Model: " + model +
79             ", Year: " + year + ", Complication: " + complication + ", Type: " + type + "];";
80     }
81 }

```

## AnalogWatch.java

```

src > com > precisionwatchcare > AnalogWatch.java > AnalogWatch
1  package com.precisionwatchcare;
2
3  public class AnalogWatch extends Watch {
4      public AnalogWatch(String brand, String model, int year, String complication) {
5          super(brand, model, year, complication, type:"Analog");
6      }
7
8      @Override
9      public void displaySpecialFeature() {
10         System.out.println("Special Feature: Mechanical movement with " + getComplication() + " complication");
11     }
12 }

```

## DigitalWatch.java

```

src > com > precisionwatchcare > DigitalWatch.java > DigitalWatch
1  package com.precisionwatchcare;
2
3  public class DigitalWatch extends Watch {
4      public DigitalWatch(String brand, String model, int year, String features) {
5          super(brand, model, year, features, type:"Digital");
6      }
7
8      @Override
9      public void displaySpecialFeature() {
10         System.out.println("Special Feature: Electronic display with " + getComplication() + " features");
11     }
12 }

```

# Cara menjalankan program

---

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
[icon] java + - [icon] [icon] ... ^ 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: 
```