

Assignment 1

Smart Home Automation Problem

Scenario:

You are a software developer working on a smart home automation system. The system needs to control multiple devices in the house, including lights, a thermostat, and a security system.

Your task is

to create a command-line interface (CLI) application using Dart to manage the smart home's functionality.

The system will handle user input to toggle devices, adjust temperature, and monitor the system's status.

Requirements:

1. Main Method:

The program should start with a `main()` function that initializes the smart home system and provides the user with options to control devices.

2. Device Status:

○ The system should include three main devices:

- Lights (can be on or off)
- Thermostat (adjustable between 15°C and 30°C)
- Security System (can be armed or disarmed)

○ Initialize each device with default values (lightsOff, 20°C for thermostat, disarmed for the security system).

3. User Interaction:

Use a loop to allow the user to continuously interact with the system until they choose to exit. The user can:

- Turn the lights on/off
- Set the thermostat temperature between 15°C and 30°C
- Arm/disarm the security system
- View system status
- Exit the program

4. Input Validation:

Ensure valid input by rejecting values outside the temperature range and prompting the user for correct values when necessary.

5. Control Flow:

- Use if/else statements to handle user choices.
- Use a switch statement to handle device commands (lights, thermostat, security system).
- Implement a FizzBuzz-like exercise: if the temperature is a multiple of 3, print "Cool Day" along with the temperature when displaying status; if it's a multiple of 5, print "Warm Day"; if it's a multiple of both, print "Perfect Day."

6. String Interpolation and Operations:

- When displaying device status, use string interpolation to include the values of the lights, thermostat, and security system in user-friendly messages.
- Implement string operations to convert device status (e.g., "armed" to "ARMED") when showing system status.

7. Final/Const Variables:

- Use const for any constant values, such as maximum and minimum temperatures.

8. Logical & Relational Operators:

- Use these to validate user input when adjusting the thermostat and when turning the security system on/off.

Solution: Smart Home Automation (Dart)

```
// Smart Home Automation - Solution (Dart)
// This program implements the CLI described in the assignment.
// Comments explain each block and variable.

import 'dart:io';

void main() {
  // Constants for min and max thermostat temps
  const int MIN_TEMP = 15;
  const int MAX_TEMP = 30;

  // Initial device states
  bool lightsOn = false;    // lights are off by default
  int thermostat = 20;      // default temperature
  bool securityArmed = false; // security system is disarmed by default

  while (true) {
    // Menu shown to the user
    print("\n===== Smart Home Menu =====");
    print('1. Toggle Lights');
    print('2. Set Thermostat');
    print('3. Arm/Disarm Security System');
    print('4. View System Status');
    print('5. Exit');
    stdout.write('Enter your choice: ');
    String? choice = stdin.readLineSync();

    if (choice == null) continue;

    // Use if/else for main control flow
    if (choice == '1') {
      // Toggle lights
      lightsOn = !lightsOn;
      print('Lights are ${lightsOn ? "ON" : "OFF"}');
    } else if (choice == '2') {
      // Set thermostat with input validation
      stdout.write('Enter thermostat temperature ($MIN_TEMP-$MAX_TEMP): ');
      String? tempStr = stdin.readLineSync();
      int? temp = int.tryParse(tempStr ?? '');
      if (temp == null) {
        print('Invalid input. Please enter a number.');
```

```

// FizzBuzz-like logic for thermostat
bool mult3 = (thermostat % 3 == 0);
bool mult5 = (thermostat % 5 == 0);
String extra = "";
if (mult3 && mult5) {
    extra = 'Perfect Day';
} else if (mult3) {
    extra = 'Cool Day';
} else if (mult5) {
    extra = 'Warm Day';
}

print("\n--- System Status ---");
print('Lights: $lightsStatus');
// If extra is not empty, include it with the temperature
print('Thermostat: $thermostat°C ${extra.isNotEmpty ? "- $extra" : ""}');
// demonstrate string operation to uppercase the security status
print('Security: ${securityStatus.toUpperCase()}');
} else if (choice == '5') {
    print('Exiting... Goodbye!');
    break;
} else {
    print('Invalid choice. Please select a valid option.');
```

Budget Management Application

Scenario:

You have been hired by a small business to create a simple budget management application using Dart. The application will help the business manage its monthly income and expenses. It will allow the user to input various income sources (like sales or investments) and expenses (like rent, salaries, and utilities), then calculate and display the net profit or loss.

Requirements:

1. Main Method:

The application should begin by welcoming the user and displaying a menu of options they can perform on the budget (e.g., add income, add expense, view summary, or exit).

2. Income and Expenses Management:

○ The user can:

■ Add income: The program should prompt the user to enter the income description and amount.

■ Add expense: The program should prompt the user to enter the expense description and amount.

■ View net profit or loss: The program should calculate the total income, total expenses, and display the net profit (income - expenses) or net loss (if expenses exceed income).

■ Exit: The user should be able to exit the program.

3. Data Storage:

○ Use lists to store multiple income and expense entries. Each entry should be stored as a String (description) and a double (amount).

○ Example: `List<Map<String, double>> incomeList = []` and `List<Map<String, double>> expenseList = []`.

4. Input Validation:

○ Ensure that the user enters valid numeric values for income and expense

amounts (i.e., no negative amounts).

○ Use logical operators to validate inputs.

5. Control Flow & Loops:

○ Use if/else statements to handle menu options.

○ Use a loop to keep asking the user for actions until they decide to exit the program.

6. String Interpolation & Operations:

○ When displaying the summary of income, expenses, and net profit or loss, use string interpolation to clearly show amounts and messages like "Your net profit is \$5000" or "Your net loss is \$200".

7. Arithmetic Operations:

○ Perform calculations for total income, total expenses, and net profit or loss.

8. Exercise: Tax Calculation (Optional):

- Introduce a tax system where, at the end of the month, the business needs to pay 15% of its net profit as tax.
- Use the ternary operator to ensure the tax is only calculated if the business is profitable. If there's a loss, no tax is calculated.
- After displaying net profit or loss, if the business made a profit, display the tax amount to be paid.

Solution: Budget Management Application (Dart)

// Budget Management Application - Solution (Dart)
 // This program stores income and expenses and shows a summary including optional tax.

```
import 'dart:io';

void main() {
  // Lists to store entries. Each map uses description => amount.
  List<Map<String, double>> incomeList = [];
  List<Map<String, double>> expenseList = [];

  while (true) {
    print('\n===== Budget Management Menu =====');
    print('1. Add Income');
    print('2. Add Expense');
    print('3. View Summary');
    print('4. Exit');
    stdout.write('Enter your choice: ');
    String? choice = stdin.readLineSync();

    if (choice == null) continue;

    if (choice == '1') {
      // Add income
      stdout.write('Enter income description: ');
      String? desc = stdin.readLineSync();
      stdout.write('Enter income amount: ');
      String? amtStr = stdin.readLineSync();
      double? amt = double.tryParse(amtStr ?? '');
      // Validate input: non-empty description and non-negative amount
      if (desc == null || desc.trim().isEmpty || amt == null || amt < 0) {
        print('Invalid input. Description required and amount must be non-negative.');
```

```

}
for (var item in expenseList) {
    totalExpense += item.values.first;
}
double net = totalIncome - totalExpense;

print('\n--- Summary ---');
print('Total Income: \${totalIncome.toStringAsFixed(2)}');
print('Total Expenses: \${totalExpense.toStringAsFixed(2)}');

if (net > 0) {
    print('Net Profit: \${net.toStringAsFixed(2)}');
    // Optional tax calculation: 15% of profit. Use ternary to compute only if profit.
    double tax = net > 0 ? net * 0.15 : 0.0;
    if (tax > 0) {
        print('Tax to pay (15%): \${tax.toStringAsFixed(2)}');
    }
} else if (net == 0) {
    print('Break-even: No profit or loss.');
```

```

} else {
    print('Net Loss: \${(-net).toStringAsFixed(2)}');
}
} else if (choice == '4') {
    print('Exiting... Goodbye!');
    break;
} else {
    print('Invalid choice. Please choose a valid option.');
```

```

}
}
}

```