Overview

This C++ program provides a set of interactive tasks that the user can choose to perform. Each task represents a separate functionality and is structured as a class that inherits from a base abstract class `LabTask`. The user selects a task from a menu, and the selected task is executed. The available tasks include a Guessing Game, a Calculator, an Even/Odd checker, and a Change Calculator.

Key Classes and Components

LabTask

This is an abstract base class that defines a pure virtual function `execute()`. All task classes inherit from this base class and implement their own version of `execute()`.

GuessingGame

This class implements a number-guessing game where the user attempts to guess a randomly generated number between 1 and 100. Feedback is provided to guide the user until they guess correctly.

Calculator

A templated class that performs basic arithmetic operations (addition, subtraction, multiplication, and division). It also records the history of operations and displays them at the end of the session.

EvenOddChecker

This class checks if a given number is even or odd and provides the result.

ChangeCalculator

This class calculates the change to be returned after a transaction and breaks it down into various denominations such as 100s, 50s, 20s, etc.

Class Descriptions

LabTask Class

```cpp
class LabTask {
public:
    virtual void execute() = 0;
};
```

This class defines the structure for the various tasks in the program. Each derived class must implement the `execute()` function, which performs the task-specific logic.

GuessingGame Class

```cpp
class GuessingGame : public LabTask {
    int targetNumber;
public:
    GuessingGame();
    void execute() override;
};
```

Functionality:

- The class generates a random target number between 1 and 100.
- The user is prompted to guess the number, receiving feedback ("too small" or "too big") until the correct number is guessed.

Calculator Class

```
template <typename T>
class Calculator : public LabTask {
    std::vector<std::string> history;
public:
    T add(T num1, T num2);
    T subtract(T num1, T num2);
    T multiply(T num1, T num2);
    T divide(T num1, T num2);
    void execute() override;
};
```

Functionality:

- This templated class performs basic arithmetic operations (addition, subtraction, multiplication, and division).
- Division handles errors (e.g., division by zero).
- All operations are logged in a history that is displayed when the session ends.

EvenOddChecker Class

```
class EvenOddChecker : public LabTask {
public:
    void check(int number);
    void execute() override;
};
```

Functionality:

- This class checks if the input number is even or odd and prints the result.

ChangeCalculator Class

```
class ChangeCalculator : public LabTask {
public:
    void calculateChange(int payable, int given);
    void execute() override;
};
```

Functionality:

- This class calculates the change that needs to be returned after a transaction.

- The change is broken down into standard denominations (100, 50, 20, 10, 5, and 1).

Execution Flow

Main Menu:

The `main()` function presents a menu to the user to select one of the tasks. The user can choose between:

1. Guessing Game
2. Calculator
3. Even or Odd Checker
4. Change Calculator

Task Execution:

Based on the user's selection, the corresponding task's `execute()` method is called. The user interacts with the program according to the task's logic, and each task handles its own input/output.

Program Repetition:

After completing a task, the user is asked if they want to select another task or exit the program. If the user chooses 'y', the menu is displayed again for a new task selection. Otherwise, the program terminates.

Example User Interaction

Guessing Game

```
Select a task:
1. Guessing Game
2. Calculator
3. Even or Odd Checker
4. Change Calculator
>> 1
Guess a number between 1 and 100: 50
Value is too small!
Guess a number between 1 and 100: 75
Value is too big!
Guess a number between 1 and 100: 62
Correct! You guessed the number.
Do you want to choose a different Program? (y/n): y
```

Calculator

```
Select a task:
1. Guessing Game
2. Calculator
3. Even or Odd Checker
4. Change Calculator
>> 2
```

```
Enter first number: 15
Enter second number: 5
Choose an operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
>> 4
Result: 3
Do you want to perform another operation? (y/n): y
Enter first number: 10
Enter second number: 2
Choose an operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
>> 1
Result: 12
Do you want to perform another operation? (y/n): n

Operation History:
Divided: 15 / 5 = 3
Added: 10 + 2 = 12

Do you want to choose a different Program? (y/n): n
```

Even or Odd Checker

```
Select a task:
1. Guessing Game
2. Calculator
3. Even or Odd Checker
4. Change Calculator
>> 3
Enter a number to check if it's even or odd: 7
7 is odd.
Do you want to choose a different Program? (y/n): n
```

Change Calculator

```
Select a task:
1. Guessing Game
2. Calculator
3. Even or Odd Checker
4. Change Calculator
>> 4
Enter the amount to pay: 85
Enter the amount given: 100
Total change: 15
10: 1
```

```
5: 1
1: 0

Do you want to choose a different Program? (y/n): n
```

Enhancements & Improvements

Error Handling:

- Handle non-numeric or invalid input for menu selection, number input, and operations.

Random Number Persistence:

- For the Guessing Game, the randomly generated number could be saved for future guesses to allow resuming a session.

Calculator Features:

- More advanced operations (e.g., square root, power function) could be added to the calculator for additional functionality.

Input Validation:

- Add input validation for edge cases like negative values in the Change Calculator.

Conclusion

This program effectively demonstrates the use of object-oriented programming principles, particularly class inheritance and polymorphism. Each task is encapsulated within its respective class and operates independently, making the code modular and extensible. With a few enhancements, the program can become even more robust and user-friendly.