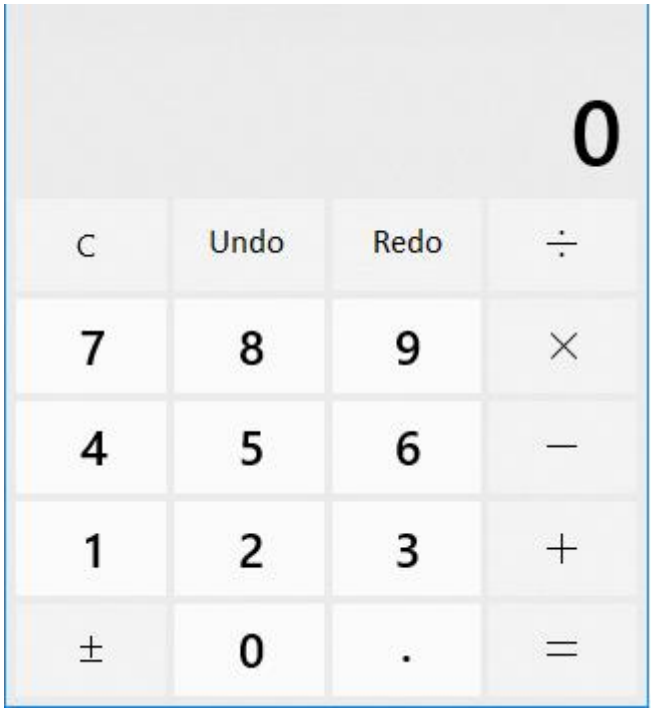
Bài kiểm tra đánh giá phân loại

Môn: Lập trình Android

Lớp: CTK40

Câu 1. Thiết kế giao diện theo hình sau:



Câu 2. Cho đoạn code sau trong ngôn ngữ lập trình C#. Hãy chuyển đổi sang ngôn ngữ lập trình Java và nhúng vào ứng dụng trên.

```
using System;
using System.Collections.Generic;

namespace DoFactory.GangOfFour.Command.RealWorld
{
  /// <summary>
  /// MainApp startup class for Real-World
  /// Command Design Pattern.
  /// </summary>
  class MainApp
  {
    /// <summary>
    /// Entry point into console application.
    /// </summary>
    static void Main()
    {
      // Create user and let her compute
      User user = new User();

```

```
      // User presses calculator buttons
      user.Compute('+', 100);
      user.Compute('-', 50);
      user.Compute('*', 10);
      user.Compute('/', 2);

      // Undo 4 commands
      user.Undo(4);

      // Redo 3 commands
      user.Redo(3);

      // Wait for user
      Console.ReadKey();
    }
  }

  /// <summary>
  /// The 'Command' abstract class
  /// </summary>
  abstract class Command
  {
    public abstract void Execute();
    public abstract void UnExecute();
  }

  /// <summary>
  /// The 'ConcreteCommand' class
  /// </summary>
  class CalculatorCommand : Command
  {
    private char _operator;
    private int _operand;
    private Calculator _calculator;

    // Constructor
    public CalculatorCommand(Calculator calculator,
      char @operator, int operand)
    {
      this._calculator = calculator;
      this._operator = @operator;
      this._operand = operand;
    }

    // Gets operator
    public char Operator
    {
      set { _operator = value; }
    }

    // Get operand
```

```
    public int Operand
    {
      set { _operand = value; }
    }

    // Execute new command
    public override void Execute()
    {
      _calculator.Operation(_operator, _operand);
    }

    // Unexecute last command
    public override void UnExecute()
    {
      _calculator.Operation(Undo(_operator), _operand);
    }

    // Returns opposite operator for given operator
    private char Undo(char @operator)
    {
      switch (@operator)
      {
        case '+': return '-';
        case '-': return '+';
        case '*': return '/';
        case '/': return '*';
        default: throw new
         ArgumentException("@operator");
      }
          }
        }

        /// <summary>
        /// The 'Receiver' class
        /// </summary>
        class Calculator
        {
          private int _curr = 0;

          public void Operation(char @operator, int operand)
          {
            switch (@operator)
            {
              case '+': _curr += operand; break;
              case '-': _curr -= operand; break;
              case '*': _curr *= operand; break;
              case '/': _curr /= operand; break;
            }
            Console.WriteLine(
              "Current value = {0,3} (following {1} {2})",
              _curr, @operator, operand);
```

```
    }
  }

  /// <summary>
  /// The 'Invoker' class
  /// </summary>
  class User
  {
    // Initializers
    private Calculator _calculator = new Calculator();
    private List<Command> _commands = new List<Command>();
    private int _current = 0;

    public void Redo(int levels)
    {
      Console.WriteLine("\n---- Redo {0} levels ", levels);
      // Perform redo operations
      for (int i = 0; i < levels; i++)
      {
        if (_current < _commands.Count - 1)
        {
          Command command = _commands[_current++];
          command.Execute();
        }
      }
    }

    public void Undo(int levels)
    {
      Console.WriteLine("\n---- Undo {0} levels ", levels);
      // Perform undo operations
      for (int i = 0; i < levels; i++)
      {
        if (_current > 0)
        {
          Command command = _commands[--_current] as Command;
          command.UnExecute();
        }
      }
    }

    public void Compute(char @operator, int operand)
    {
      // Create command operation and execute it
      Command command = new CalculatorCommand(
        _calculator, @operator, operand);
      command.Execute();

      // Add command to undo list
      _commands.Add(command);
      _current++;
```

```
173.             }
174.           }
175.         }
```