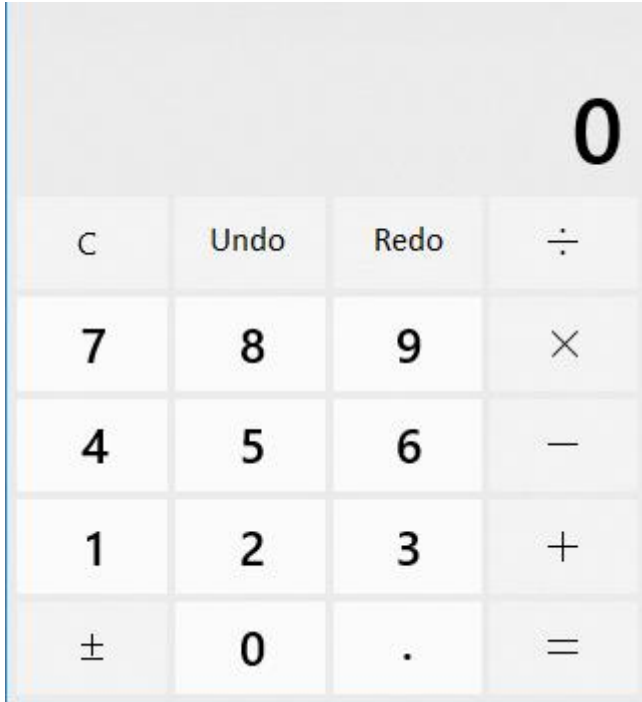


## Bài kiểm tra đánh giá phân loại

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.

```
1. using System;
2. using System.Collections.Generic;
3.
4. namespace DoFactory.GangOfFour.Command.RealWorld
5. {
6.     /// <summary>
7.     /// MainApp startup class for Real-World
8.     /// Command Design Pattern.
9.     /// </summary>
10.    class MainApp
11.    {
12.        /// <summary>
13.        /// Entry point into console application.
14.        /// </summary>
15.        static void Main()
16.        {
17.            // Create user and let her compute
18.            User user = new User();
19.
```

```

20.     // User presses calculator buttons
21.     user.Compute('+', 100);
22.     user.Compute('-', 50);
23.     user.Compute('*', 10);
24.     user.Compute('/', 2);
25.
26.     // Undo 4 commands
27.     user.Undo(4);
28.
29.     // Redo 3 commands
30.     user.Redo(3);
31.
32.     // Wait for user
33.     Console.ReadKey();
34. }
35. }
36.
37. /// <summary>
38. /// The 'Command' abstract class
39. /// </summary>
40. abstract class Command
41. {
42.     public abstract void Execute();
43.     public abstract void UnExecute();
44. }
45.
46. /// <summary>
47. /// The 'ConcreteCommand' class
48. /// </summary>
49. class CalculatorCommand : Command
50. {
51.     private char _operator;
52.     private int _operand;
53.     private Calculator _calculator;
54.
55.     // Constructor
56.     public CalculatorCommand(Calculator calculator,
57.         char @operator, int operand)
58.     {
59.         this._calculator = calculator;
60.         this._operator = @operator;
61.         this._operand = operand;
62.     }
63.
64.     // Gets operator
65.     public char Operator
66.     {
67.         set { _operator = value; }
68.     }
69.
70.     // Get operand

```

```

71.     public int Operand
72.     {
73.         set { _operand = value; }
74.     }
75.
76.     // Execute new command
77.     public override void Execute()
78.     {
79.         _calculator.Operation(_operator, _operand);
80.     }
81.
82.     // Unexecute last command
83.     public override void UnExecute()
84.     {
85.         _calculator.Operation(Undo(_operator), _operand);
86.     }
87.
88.     // Returns opposite operator for given operator
89.     private char Undo(char @operator)
90.     {
91.         switch (@operator)
92.         {
93.             case '+': return '-';
94.             case '-': return '+';
95.             case '*': return '/';
96.             case '/': return '*';
97.             default: throw new
98.                 ArgumentException("@operator");
99.         }
100.    }
101. }
102.
103.     /// <summary>
104.     /// The 'Receiver' class
105.     /// </summary>
106.     class Calculator
107.     {
108.         private int _curr = 0;
109.
110.         public void Operation(char @operator, int operand)
111.         {
112.             switch (@operator)
113.             {
114.                 case '+': _curr += operand; break;
115.                 case '-': _curr -= operand; break;
116.                 case '*': _curr *= operand; break;
117.                 case '/': _curr /= operand; break;
118.             }
119.             Console.WriteLine(
120.                 "Current value = {0,3} (following {1} {2})",
121.                 _curr, @operator, operand);

```

```

122.     }
123. }
124.
125. /// <summary>
126. /// The 'Invoker' class
127. /// </summary>
128. class User
129. {
130.     // Initializers
131.     private Calculator _calculator = new Calculator();
132.     private List<Command> _commands = new List<Command>();
133.     private int _current = 0;
134.
135.     public void Redo(int levels)
136.     {
137.         Console.WriteLine("\n---- Redo {0} levels ", levels);
138.         // Perform redo operations
139.         for (int i = 0; i < levels; i++)
140.         {
141.             if (_current < _commands.Count - 1)
142.             {
143.                 Command command = _commands[_current++];
144.                 command.Execute();
145.             }
146.         }
147.     }
148.
149.     public void Undo(int levels)
150.     {
151.         Console.WriteLine("\n---- Undo {0} levels ", levels);
152.         // Perform undo operations
153.         for (int i = 0; i < levels; i++)
154.         {
155.             if (_current > 0)
156.             {
157.                 Command command = _commands[--_current] as Command;
158.                 command.UnExecute();
159.             }
160.         }
161.     }
162.
163.     public void Compute(char @operator, int operand)
164.     {
165.         // Create command operation and execute it
166.         Command command = new CalculatorCommand(
167.             _calculator, @operator, operand);
168.         command.Execute();
169.
170.         // Add command to undo list
171.         _commands.Add(command);
172.         _current++;

```

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