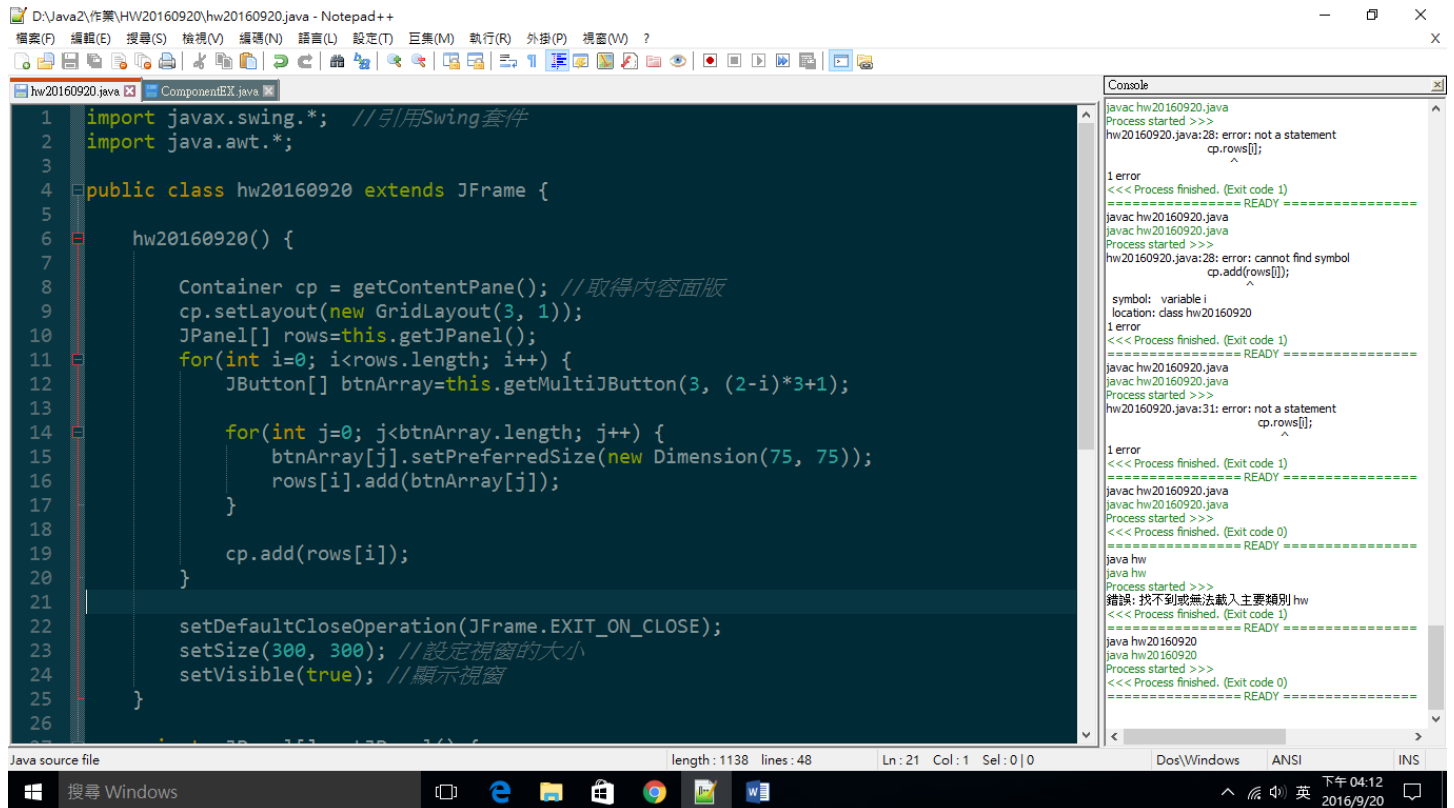


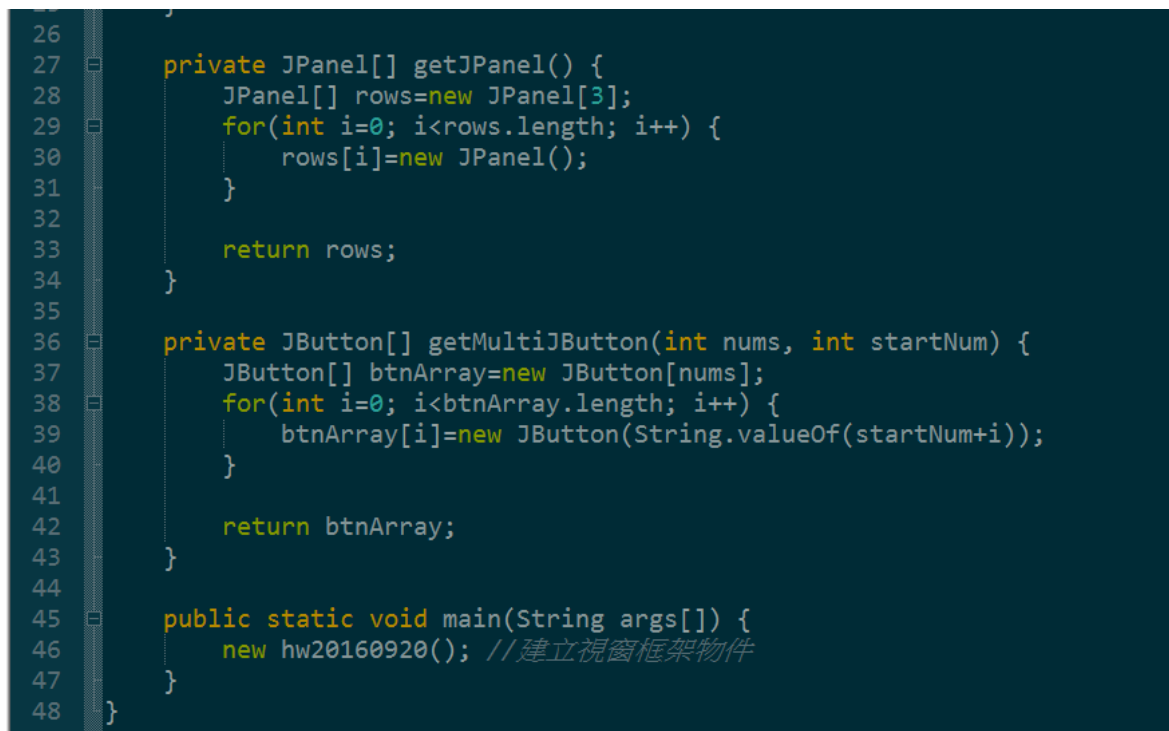
Hw20160920



The screenshot shows a Notepad++ window with a Java source file named `hw20160920.java`. The code defines a class `hw20160920` that extends `JFrame`. It includes imports for `javax.swing.*` and `java.awt.*`. The `hw20160920()` constructor initializes a `Container` `cp` with a `GridLayout(3, 1)`, creates a `JPanel` array `rows`, and populates it with `JButton` objects. The `main` method creates an instance of `hw20160920` and displays it.

```
1 import javax.swing.*; //引用Swing套件
2 import java.awt.*;
3
4 public class hw20160920 extends JFrame {
5
6     hw20160920() {
7
8         Container cp = getContentPane(); //取得內容面板
9         cp.setLayout(new GridLayout(3, 1));
10        JPanel[] rows=this.getJPanel();
11        for(int i=0; i<rows.length; i++) {
12            JButton[] btnArray=this.getMultiJButton(3, (2-i)*3+1);
13
14            for(int j=0; j<btnArray.length; j++) {
15                btnArray[j].setPreferredSize(new Dimension(75, 75));
16                rows[i].add(btnArray[j]);
17            }
18
19            cp.add(rows[i]);
20        }
21
22        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
23        setSize(300, 300); //設定視窗的大小
24        setVisible(true); //顯示視窗
25    }
26 }
```

The console window on the right shows the compilation and execution process. It includes messages like "Process started >>>", "Process finished. (Exit code 1)", and "error: not a statement" for the `cp.add(rows[i]);` line. The final output shows the application running successfully.



This screenshot shows the implementation of the helper methods for the `hw20160920` class. It includes the `getJPanel()` method, which creates a `JPanel` array and returns it, and the `getMultiJButton()` method, which creates an array of `JButton` objects and returns it. The `main` method is also shown, which creates an instance of the class and displays it.

```
26
27 private JPanel[] getJPanel() {
28     JPanel[] rows=new JPanel[3];
29     for(int i=0; i<rows.length; i++) {
30         rows[i]=new JPanel();
31     }
32
33     return rows;
34 }
35
36 private JButton[] getMultiJButton(int nums, int startNum) {
37     JButton[] btnArray=new JButton[nums];
38     for(int i=0; i<btnArray.length; i++) {
39         btnArray[i]=new JButton(String.valueOf(startNum+i));
40     }
41
42     return btnArray;
43 }
44
45 public static void main(String args[]) {
46     new hw20160920(); //建立視窗框架物件
47 }
48 }
```

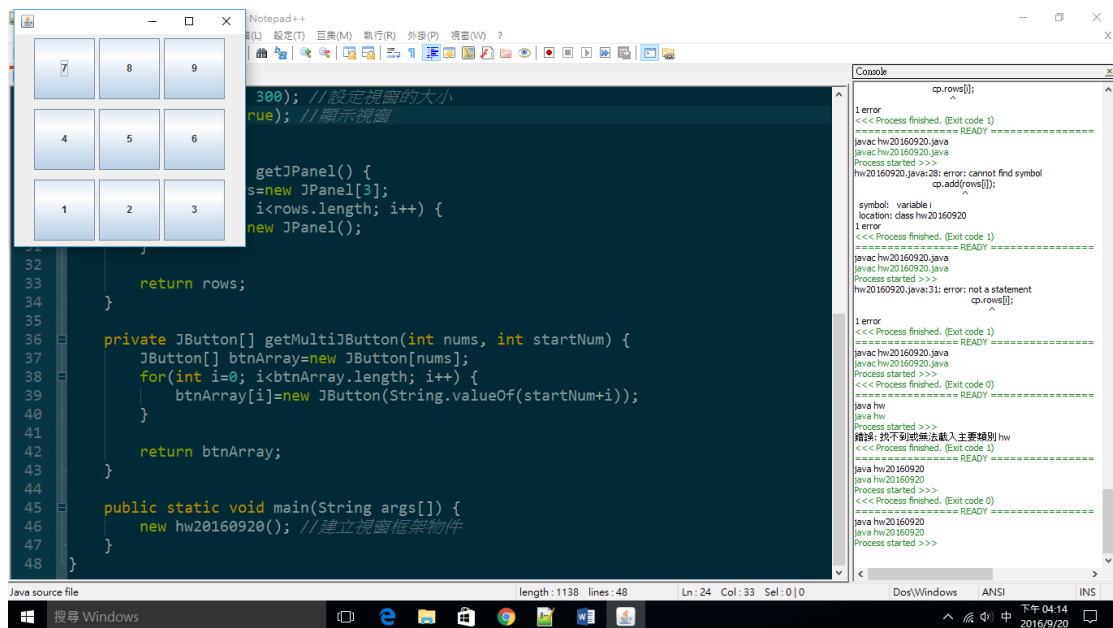
封裝函式：

1. JPanel[] getPanel(int nums)：產生幾個 JPanel
2. JButton[] getMultiJButton(int nums, int startNum)：產生幾個連續數值的 JButton

完整程式碼：

<https://goo.gl/ZP43TN>

執行結果



The screenshot displays a Java IDE with a 3x3 grid of buttons in the top-left window. The buttons are labeled with numbers 1 through 9. The main editor window shows the Java code for this application. The code includes two methods: `getPanel()` which creates a 3x3 grid of `JPanel` objects, and `getMultiJButton()` which creates an array of `JButton` objects with sequential labels. The `main` method initializes the application and displays the grid.

```
300); //設定視窗的大小
true); //顯示視窗

getJPanel() {
    s=new JPanel[3];
    i<rows.length; i++) {
        new JPanel();
    }
    return rows;
}

private JButton[] getMultiJButton(int nums, int startNum) {
    JButton[] btnArray=new JButton[nums];
    for(int i=0; i<btnArray.length; i++) {
        btnArray[i]=new JButton(String.valueOf(startNum+i));
    }
    return btnArray;
}

public static void main(String args[]) {
    new hw20160920(); //建立視窗框架物件
}
```

The console window on the right shows the output of the program, including the execution of the `main` method and the creation of the `JPanel` and `JButton` arrays.