

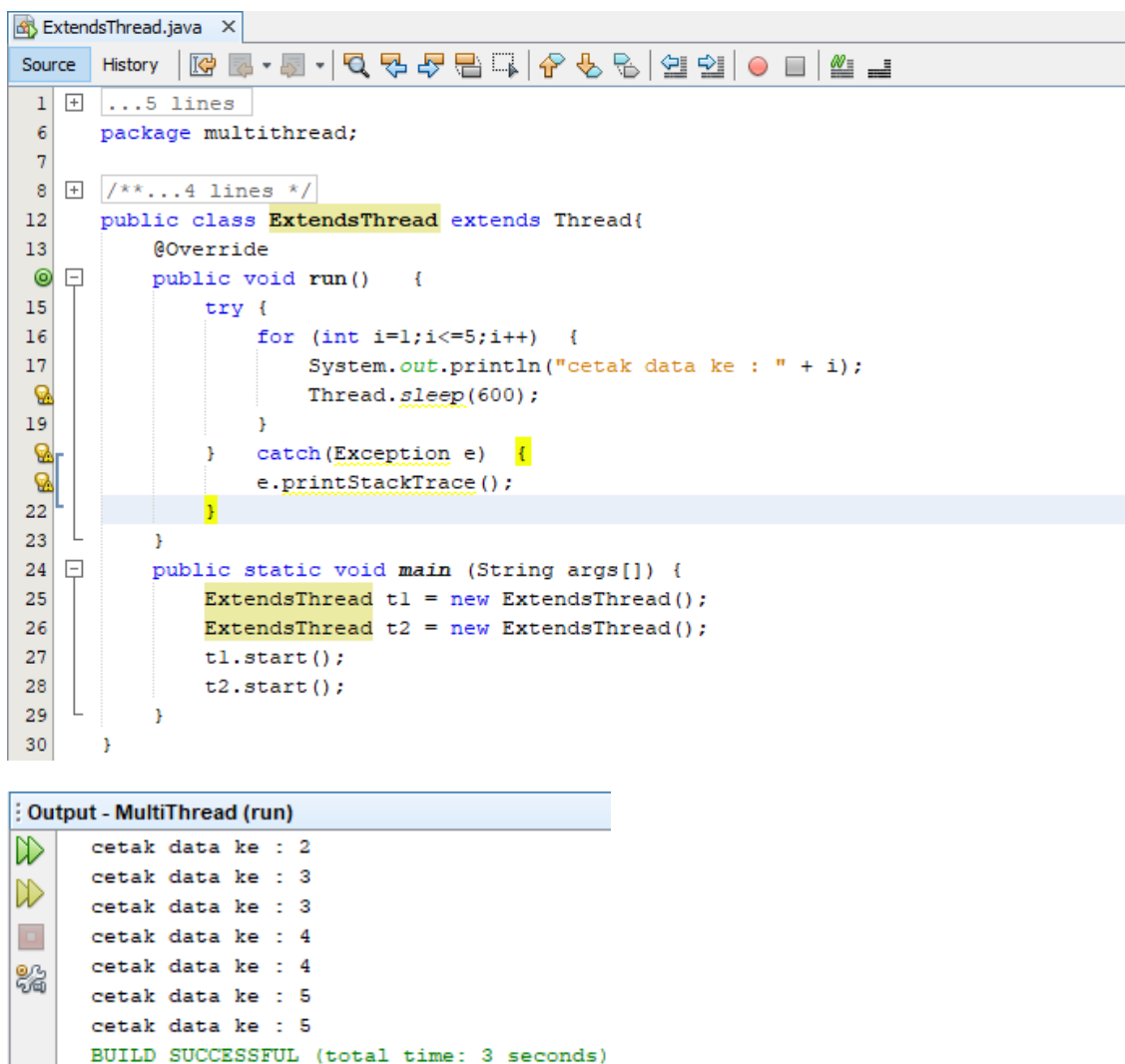
Nama : Alya Aiman Salsabila Arif

NPM : 1817101379

Kelas : III Rekayasa Perangkat Lunak Kripto

Tugas 10

1. Implementasi *Thread* dengan Kelas Thread



The screenshot displays an IDE window titled "ExtendsThread.java" with a source code editor and an output console. The code defines a class `ExtendsThread` that extends `Thread`. It overrides the `run()` method to print data from 1 to 5, with a 600ms delay between each print. A `main` method creates two instances of `ExtendsThread` and starts them. The output console shows the execution results, including the printed data and a successful build message.

```
1  ...5 lines
6  package multithread;
7
8  /**...4 lines */
12 public class ExtendsThread extends Thread{
13     @Override
14     public void run() {
15         try {
16             for (int i=1;i<=5;i++) {
17                 System.out.println("cetak data ke : " + i);
18                 Thread.sleep(600);
19             }
20         } catch (Exception e) {
21             e.printStackTrace();
22         }
23     }
24     public static void main (String args[]) {
25         ExtendsThread t1 = new ExtendsThread();
26         ExtendsThread t2 = new ExtendsThread();
27         t1.start();
28         t2.start();
29     }
30 }
```

Output - MultiThread (run)

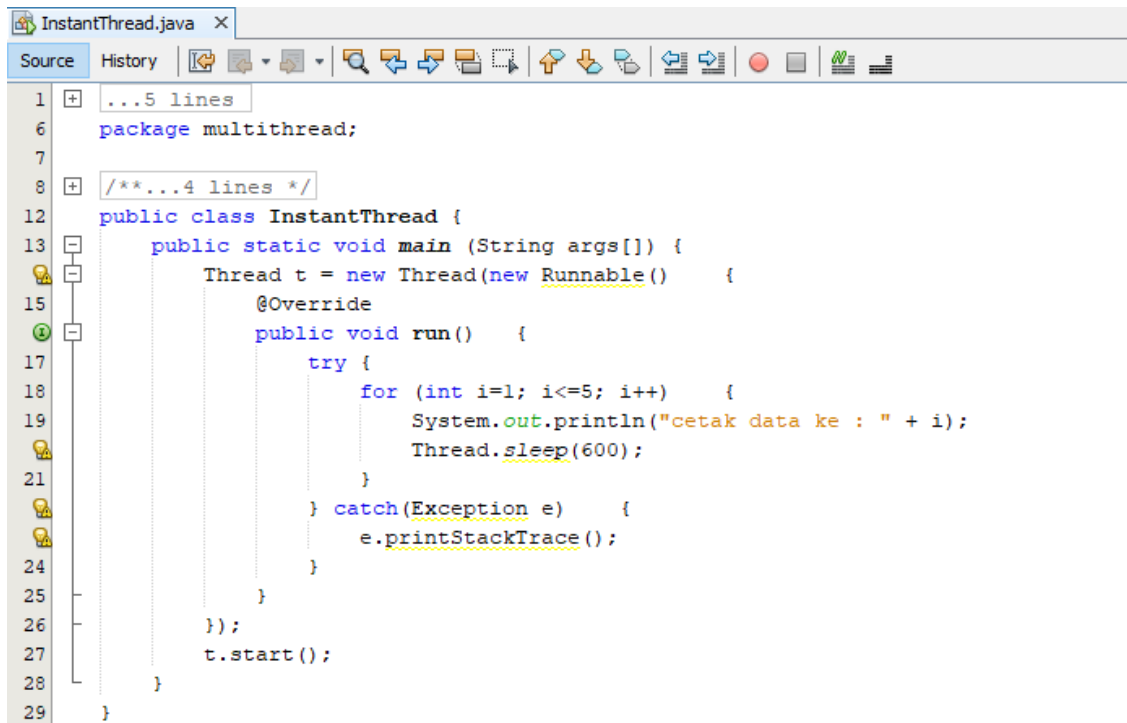
```
cetak data ke : 2
cetak data ke : 3
cetak data ke : 3
cetak data ke : 4
cetak data ke : 4
cetak data ke : 5
cetak data ke : 5
BUILD SUCCESSFUL (total time: 3 seconds)
```

2. Implementasi *Thread* dengan *Interface Runnable*

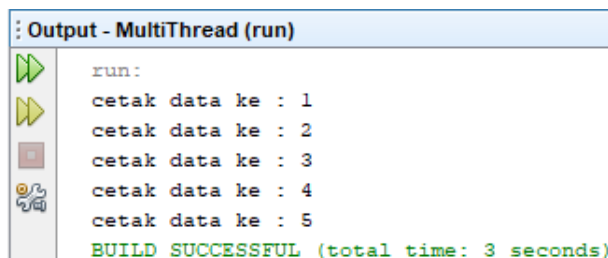
```
ImplementsRunnable.java x
Source History
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6   package multithread;
7
8   /**
9    *
10   * @author Alya Aiman Salsabila Arif
11   */
12   public class ImplementsRunnable implements Runnable {
13       @Override
14       public void run() {
15           try {
16               for (int i=1; i<=5; i++) {
17                   System.out.println("cetak data ke : " + i);
18                   Thread.sleep(600);
19               }
20           } catch (Exception e) {
21               e.printStackTrace();
22           }
23       }
24       public static void main (String args[]) {
25           ExtendsThread t1 = new ExtendsThread();
26           t1.start();
27       }
28   }
```

```
Output - MultiThread (run)
run:
cetak data ke : 1
cetak data ke : 2
cetak data ke : 3
cetak data ke : 4
cetak data ke : 5
BUILD SUCCESSFUL (total time: 3 seconds)
```

3. Implementasi *Thread* dengan Membuat *Instance* dari Kelas *Thread*



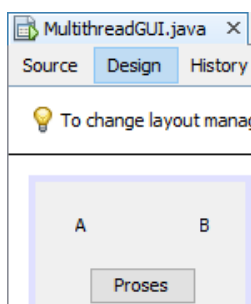
```
1  ...5 lines
6  package multithread;
7
8  /**...4 lines */
12 public class InstantThread {
13     public static void main (String args[]) {
14         Thread t = new Thread(new Runnable() {
15             @Override
16             public void run() {
17                 try {
18                     for (int i=1; i<=5; i++) {
19                         System.out.println("cetak data ke : " + i);
20                         Thread.sleep(600);
21                     }
22                 } catch (Exception e) {
23                     e.printStackTrace();
24                 }
25             }
26         });
27         t.start();
28     }
29 }
```



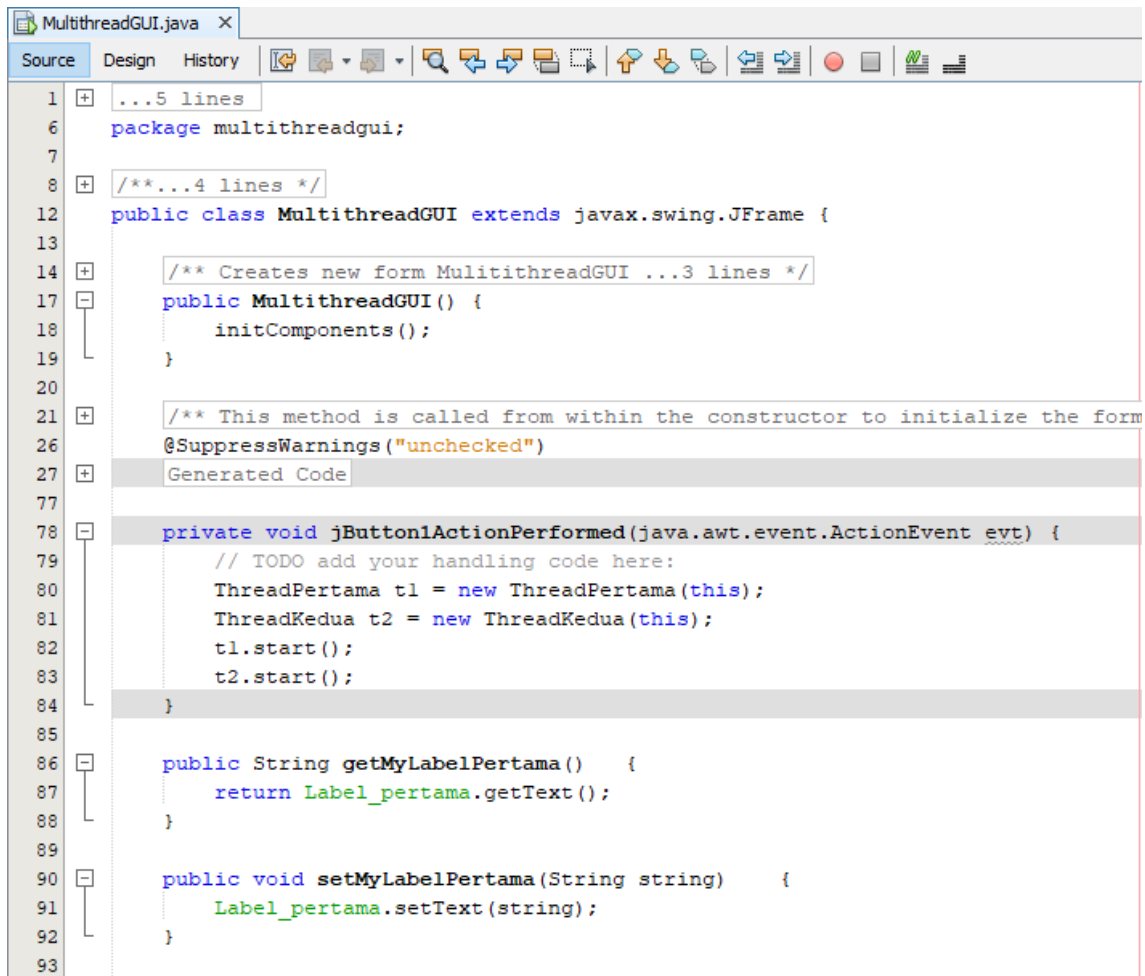
```
Output - MultiThread (run)
run:
cetak data ke : 1
cetak data ke : 2
cetak data ke : 3
cetak data ke : 4
cetak data ke : 5
BUILD SUCCESSFUL (total time: 3 seconds)
```

4. Praktikum

Buatlah projek baru dengan nama MultithreadGUI. Buatlah kelas MultithreadGUI dengan JFrame Form. Buatlah desain kelas MultithreadGUI



Masukkan *source code* berikut pada kelas MultithreadGUI



```
1  ...5 lines
6  package multithreadgui;
7
8  /**...4 lines */
12 public class MultithreadGUI extends javax.swing.JFrame {
13
14     /** Creates new form MultitithreadGUI ...3 lines */
17     public MultithreadGUI() {
18         initComponents();
19     }
20
21     /** This method is called from within the constructor to initialize the form
26     @SuppressWarnings("unchecked")
27     Generated Code
77
78     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
79         // TODO add your handling code here:
80         ThreadPertama t1 = new ThreadPertama(this);
81         ThreadKedua t2 = new ThreadKedua(this);
82         t1.start();
83         t2.start();
84     }
85
86     public String getMyLabelPertama() {
87         return Label_pertama.getText();
88     }
89
90     public void setMyLabelPertama(String string) {
91         Label_pertama.setText(string);
92     }
93
```

```

94     public String getMyLabelKedua() {
95         return Label_kedua.getText();
96     }
97
98     public void setMyLabelKedua(String string) {
99         Label_kedua.setText(string);
100     }
101
102     /**
103      * @param args the command line arguments
104      */
105     public static void main(String args[]) {
106         /* Set the Nimbus look and feel */
107         Look and feel setting code (optional)
108         //</editor-fold>
109
110         /* Create and display the form */
111         java.awt.EventQueue.invokeLater(new Runnable() {
112             public void run() {
113                 new MultithreadGUI().setVisible(true);
114             }
115         });
116     }
117
118     // Variables declaration - do not modify
119     private javax.swing.JLabel Label_kedua;
120     private javax.swing.JLabel Label_pertama;
121     private javax.swing.JButton jButton1;
122     // End of variables declaration
123 }

```

Buat kelas ThreadPertama dan masukkan *source code* sebagai berikut

```

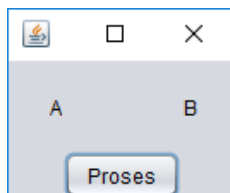
ThreadPertama.java x
Source History
1  ...5 lines
2  package multithreadgui;
3
4  import java.util.logging.Level;
5  import java.util.logging.Logger;
6
7  /**...4 lines */
8
9  public class ThreadPertama extends Thread{
10     private MultithreadGUI f;
11
12     public ThreadPertama(MultithreadGUI f) {
13         this.f = f;
14     }
15
16     @Override
17     public void run() {
18         //body of Thread
19         for (int i=0;i<20;i++) {
20             f.setMyLabelPertama(String.valueOf(i));
21             try {
22                 Thread.sleep(900);
23             } catch (InterruptedException ex) {
24                 Logger.getLogger(ThreadPertama.class.getName()).log(Level.SEVERE, null, ex);
25             }
26         }
27     }
28 }
29
30
31
32

```

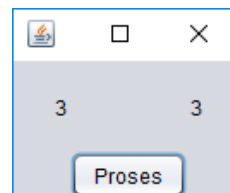
Buat kelas ThreadKedua dan masukkan *source code* sebagai berikut

```
ThreadKedua.java x
Source History
1  ...5 lines
6  package multithreadgui;
7
8  import java.util.logging.Level;
9  import java.util.logging.Logger;
10 /**...4 lines */
14 public class ThreadKedua extends Thread{
15     private MultithreadGUI f;
16
17     public ThreadKedua(MultithreadGUI f){
18         this.f = f;
19     }
20     @Override
21     public void run() {
22         //body of Thread
23         for (int i=0;i<20;i++) {
24             f.setMyLabelKedua(String.valueOf(i));
25             try {
26                 Thread.sleep(900);
27             } catch (InterruptedException ex) {
28                 Logger.getLogger(ThreadPertama.class.getName()).log(Level.SEVERE, null, ex);
29             }
30         }
31     }
32 }
```

Jalankan program. Program akan menampilkan sebagai berikut



yang dibuat pada kelas ThreadPertama dan kelas ThreadKedua



Tekan tombol *button* Proses. Program akan menampilkan angka perulangan secara bersamaan sesuai perulangan