8_03_2022__java_Exception, Swing, Thread

Step13_Exception.java

~Step15_Thread.java

스태틱 메소드는 클래스명에 점 찍어서 사용

전송, 삭제 버튼 둘 다 똑같이 출력된다.

```
package frame06;

import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;

public class MyFrame extends JFrame implements ActionListener{

   //생성자
   public MyFrame(String title) {
      //부모생성자에 프레임의 제목 넘겨주기
      super(title);

      setLayout(new FlowLayout());



      JButton sendBtn=new JButton("전송");
      //프레임에 버튼 추가하기 ( FlowLayout 의 영향을 받는다 )
      add(sendBtn);
      sendBtn.addActionListener(this);



      //삭제 버튼을 만들어서
      JButton deleteBtn=new JButton("삭제");
      //프레임에 추가하기
      add(deleteBtn);

      deleteBtn.addActionListener(this);
   }
```

```
   public static void main(String[] args) {

      JFrame f=new MyFrame("나의 프레임");
      f.setBounds(100, 100, 500, 500);
      f.setDefaultCloseOperation(EXIT_ON_CLOSE);
      f.setVisible(true);

   }

   @Override
   public void actionPerformed(ActionEvent e) {

      JOptionPane.showMessageDialog(this, "전송 합니다.");
   }
}
```

다르게 출력되게 만들기

- getSource사용

```
package frame06;

import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;

public class MyFrame extends JFrame implements ActionListener{
  JButton sendBtn;
  JButton deleteBtn;

   //생성자
   public MyFrame(String title) {
      //부모생성자에 프레임의 제목 넘겨주기
      super(title);
      setLayout(new FlowLayout());



      sendBtn=new JButton("전송");
      //프레임에 버튼 추가하기 ( FlowLayout 의 영향을 받는다 )
      add(sendBtn);
      sendBtn.addActionListener(this);



      //삭제 버튼을 만들어서
      deleteBtn=new JButton("삭제");
      //프레임에 추가하기
      add(deleteBtn);

      deleteBtn.addActionListener(this);
   }
```

```
   public static void main(String[] args) {

      JFrame f=new MyFrame("나의 프레임");
      f.setBounds(100, 100, 500, 500);
      f.setDefaultCloseOperation(EXIT_ON_CLOSE);
      f.setVisible(true);

   }

   @Override
   public void actionPerformed(ActionEvent e) {
    //이벤트가 발생한 객체(여기서는 JButton 객체)의 참조값을 리턴해준다.
    Object obj = e.getSource();
    if(obj == sendBtn) {
      JOptionPane.showMessageDialog(this, "전송 합니다.");
    }else if(obj == deleteBtn) {
      JOptionPane.showMessageDialog(this, "삭제 합니다.");
    }
   }
}
```

객체 안에서 공유할 값들은 필드 안으로 넣으면 된다.

```
  JButton sendBtn;
  JButton deleteBtn;


 sendBtn=new JButton("전송");
deleteBtn=new JButton("삭제");
```

필드 자체에서 new 할수도 있다.

```
JButton sendBtn=new JButton("전송");
JButton deleteBtn=new JButton("삭제");
```

- getActionCOmmand 사용
  - 필드 만들지 않고 하는 방법

```
package frame06;

import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
```

```
public class MyFrame2 extends JFrame implements ActionListener{
   //생성자
   public MyFrame2(String title) {
      //부모생성자에 프레임의 제목 넘겨주기
      super(title);
      setLayout(new FlowLayout());



      JButton sendBtn=new JButton("전송");
      //프레임에 버튼 추가하기 ( FlowLayout 의 영향을 받는다 )
      add(sendBtn);
      sendBtn.addActionListener(this);



      //삭제 버튼을 만들어서
      JButton deleteBtn=new JButton("삭제");
      //프레임에 추가하기
      add(deleteBtn);
      deleteBtn.addActionListener(this);



      //각각의 버튼에 ActionCommand 설정
      sendBtn.setActionCommand("send");
      deleteBtn.setActionCommand("delete");
   }



   public static void main(String[] args) {

      JFrame f=new MyFrame2("나의 프레임");
      f.setBounds(100, 100, 500, 500);
      f.setDefaultCloseOperation(EXIT_ON_CLOSE);
      f.setVisible(true);

   }

   @Override
   public void actionPerformed(ActionEvent e) {
      //이벤트가 일어난 객체에 설정된 ActionCommand 문자열 읽어오기
     String command=e.getActionCommand();
     if(command == "send") {
       JOptionPane.showMessageDialog(this, "전송 합니다.");
     }else if(command == "delete") {
       JOptionPane.showMessageDialog(this, "삭제 합니다.");
     }
   }
}
```

문자열을 set해서 get으로 받아와 비교해 출력한다.

```
sendBtn.setActionCommand("send");
deleteBtn.setActionCommand("delete");


String command=e.getActionCommand();
if(command == "send") {
 JOptionPane.showMessageDialog(this, "전송 합니다.");
}else if(command == "delete") {
```

```
 JOptionPane.showMessageDialog(this, "삭제 합니다.");
}
```

- 문자열 비교에 대해서
  - 문자열을 비교한다는 것은 참조값이 같은지 비교하는 걸까?
  - 문자열의 내용이 같은지 비교하는 걸까?
  - 그렇다면 ==연산자는 뭘 비교하는 연산자일까?
    - 참조값을 비교하는 연산자이다.
  - 그러면 문자열의 내용이 같으면 참조값이 같을까?
    - 같을때도 있고 아닐때도 있다.

> 💡 결론 - 문자열의 내용을 비교할 때 ==를 사용하면 안된다.

  - 비교하는 방법은 String 객체의 .equals() 메소드를 활용하면 된다.

```
@Override
   public void actionPerformed(ActionEvent e) {
      //이벤트가 일어난 객체에 설정된 ActionCommand 문자열 읽어오기
     String command=e.getActionCommand();
     if(command.equals("send")) {
       JOptionPane.showMessageDialog(this, "전송 합니다.");
     }else if(command.equals("delete")) {
       JOptionPane.showMessageDialog(this, "삭제 합니다.");
     }
```

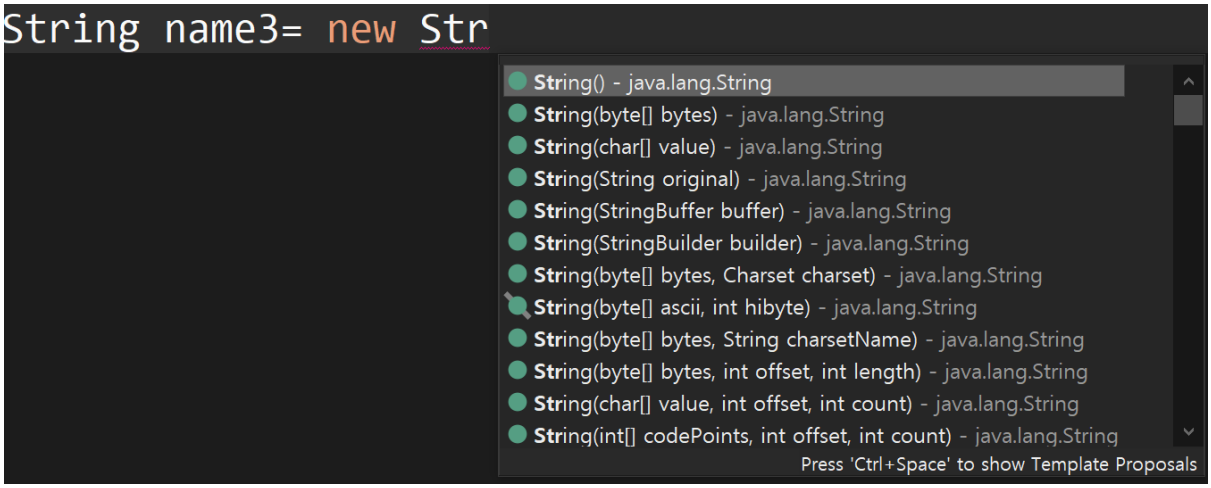
문자열의 내용이 같으면 참조값이 같을까?

- “”로 감싸서 만들면 같다

```
package frame06;

public class StringTestMain {
  public static void main(String[] args) {
    System.out.println("main 메소드가 시작 되었습니다.");
    String name1="두부";
    String name2="두부";
  }
}
```

- 하지만 더 자바스러운 방법인 new로 사용해서 만들면 엄청나게 많은 방법이 있다.

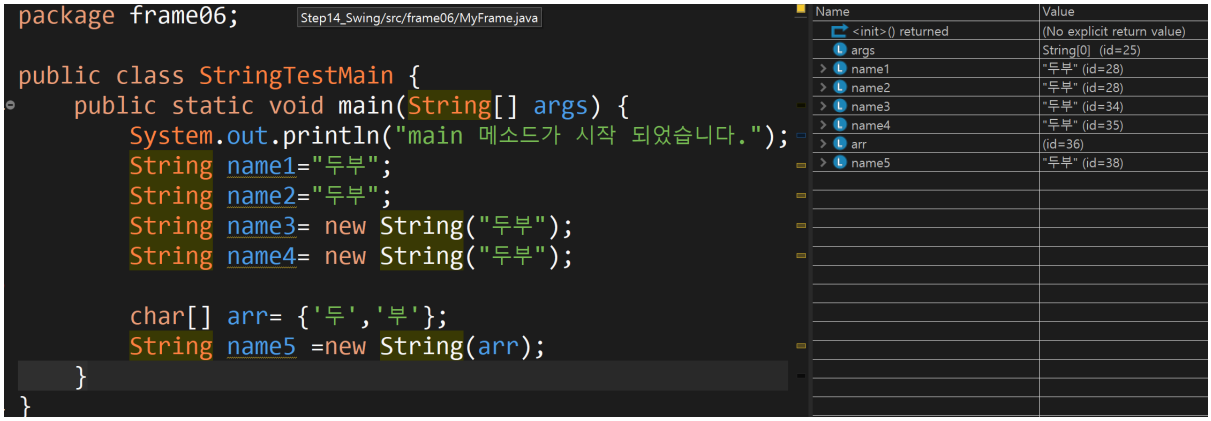
```
package frame06;

public class StringTestMain {
  public static void main(String[] args) {
    System.out.println("main 메소드가 시작 되었습니다.");
    String name1="두부";
    String name2="두부";
    String name3= new String("두부");
  }
}
```



- 같은 문자열이지만 참조값이 다르다

name1	"두부" (id=28)
name2	"두부" (id=28)
name3	"두부" (id=34)

- 여러 방법 비교



JTextField

- js 의 inputText
- 한쪽 텍스트 입력창에서 다른쪽 입력창으로 버튼을 누르면 옮겨가게 하는 방법

```
package frame07;

import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

public class MyFrame extends JFrame implements ActionListener{
  JTextField inputMsg1, inputMsg2;
  JButton btn;

  //생성자
  public MyFrame(String title) {
    super(title);
    //레이아웃
    setLayout(new FlowLayout());
    inputMsg1 = new JTextField(10);
    inputMsg2 = new JTextField(10);
    btn=new JButton("눌러보세요");

    add(inputMsg1);
    add(btn);
    add(inputMsg2);
    btn.addActionListener(this);
  }

  public static void main(String[] args) {
    JFrame f = new MyFrame("나의 프레임");
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    f.setBounds(100, 100, 500, 500);
    f.setVisible(true);
  }

  @Override
```

```
  public void actionPerformed(ActionEvent e) {
    String msg = inputMsg1.getText();
    inputMsg2.setText(msg);
  }
}
```

JPanel

- 배경으로 ui묶음
- 문자열 하나하나는 JLabel

계산기

```
package frame08;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class CalcFrame extends JFrame implements ActionListener {
  // 필드
  JTextField tf_num1, tf_num2;
  JLabel label_result;

  // default 생성자
  public CalcFrame() {
    // 프레임의 레이아웃 법칙 설정하기
    setLayout(new BorderLayout());

    // JPanel
    JPanel topPanel = new JPanel();
    topPanel.setBackground(Color.YELLOW);
    // Panel 을 북쪽에 배치하기

    add(topPanel, BorderLayout.NORTH);

    // JTextField 객체를 만들에서 JPanel 에 추가하기
    tf_num1 = new JTextField(10);
    topPanel.add(tf_num1);

    // 기능 버튼 객체를 만들어서 JPanel 에 추가하기
    JButton plusBtn = new JButton("+");
    JButton minusBtn = new JButton("-");
    JButton multiBtn = new JButton("*"); // actionCommand를 지정하지 않으면 버튼 텍스트가 actionCommand가 됨
    JButton divideBtn = new JButton("/");
    topPanel.add(plusBtn);
```

```
    topPanel.add(minusBtn);
    topPanel.add(multiBtn);
    topPanel.add(divideBtn);

    // 두번째 JTextField 만들어서 페널에 추가 하기
    tf_num2 = new JTextField(10);
    topPanel.add(tf_num2);

    // JLabel
    JLabel label1 = new JLabel("=");
    label_result = new JLabel("0");

    // 페널에 레이블 추가하기
    topPanel.add(label1);
    topPanel.add(label_result);

    // 버튼에 액션 리스너 등록
    plusBtn.addActionListener(this);
    minusBtn.addActionListener(this);
    multiBtn.addActionListener(this);
    divideBtn.addActionListener(this);

  }

  public static void main(String[] args) {

    CalcFrame frame = new CalcFrame();
    // 프레임의 제목 설정
    frame.setTitle("계산기");
    // 프레임을 닫으면 자동으로 프로세스가 종료 되도록 한다.
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setBounds(100, 100, 500, 500);
    frame.setVisible(true);
  }

  @Override
  public void actionPerformed(ActionEvent e) {
    String command = e.getActionCommand();
    // 입력한 문자열을 읽어낸다
    String strNum1 = tf_num1.getText();
    String strNum2 = tf_num2.getText();
    // 활용할 지역 변수를 미리 만들어 놓고
    double result = 0;
    try {
      // 문자열을 숫자로 바꿔서
      double num1 = Double.parseDouble(strNum1);
      double num2 = Double.parseDouble(strNum2);
      // 버튼에 액션 command를 지정하지 않으면 버튼 text가 액션 command가 된다.
      if (command.equals("+")) {
        // 미리 만들어 놓은 지역변수에 연산의 결과를 대입한다.
        result = num1 + num2;
      } else if (command.equals("-")) {
        result = num1 - num2;
      } else if (command.equals("*")) {
        result = num1 * num2;
      } else if (command.equals("/")) {
        if (num2 == 0) {
          JOptionPane.showMessageDialog(this, "0으로 나눌 수 없어요");
          return; // 메소드를 여기서 끝내기ㅣ
        }
        result = num1 / num2;
      }
      // 숫자를 문자열로 바꾸기
      String strResult = Double.toString(result);
      // 연산의 결과를 JLabel에 출력하기
```

```
      label_result.setText(strResult);
    } catch (NumberFormatException nfe) {
      JOptionPane.showMessageDialog(this, "숫자 형식으로 입력하세요!");
    }
  }
}
```

Thread (작업의 흐름)

- 작업단위가 하나만 사용중
- 필요에 따라 여러개 만들줄 알아야함
- JAVA에서 하나의 스레드만 사용하는 것은 카페에 갔는데 일하는 직원이 한명. 그 직원은 이렇게 일한다
  - 주문을 받는다
  - 해당메뉴를 준비한다
  - 메뉴가 완성된다
  - 다음 주문을 받는다
  - 그 메뉴가 다 만들어질 때 까지 다음 주문을 받지 않는다

```
package test.main;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

public class Frame02 extends JFrame implements ActionListener {

  // 생성자
  public Frame02() {
    // 레이아웃 설정
    setLayout(new BorderLayout());
    // 페널을 프레임의 상단에 배치
    JPanel panel = new JPanel();
    panel.setBackground(Color.YELLOW);
    add(panel, BorderLayout.NORTH);
    // 버튼을 페널에 추가 하고
    JButton countBtn = new JButton("1~10 까지 세기");
    panel.add(countBtn);
    // 버튼에 리스너 등록하기
    countBtn.addActionListener(this);
  }

  public static void main(String[] args) {
```

```
    // MyFrame 클래스를 이용해서 객체 생성하고 참조값을 지역변수 frame 에 담기
    Frame02 frame = new Frame02();
    // 프레임의 제목 설정
    frame.setTitle("Frame02");
    // 프레임을 닫으면 자동으로 프로세스가 종료 되도록 한다.
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setBounds(100, 100, 500, 500);
    frame.setVisible(true);
  }

  @Override
  public void actionPerformed(ActionEvent arg0) {
    int count = 0;
    while (true) {
      try {
        Thread.sleep(1000);
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
      count++;
      System.out.println("현재 카운트:" + count);
      if (count == 10) {
        break;
      }
    }

  }
}
```

- JAVA에서 여러개의 스레드를 사용하는 것은
  - 손님이 카페에 갔을 때 일하는 직원이 한명 생긴다.
    - 그 직원은 그 손님의 주문을 받고 준비한다.
  - 다른 손님이 오면 또 다른 직원이 생긴다
    - 그 직원은 그 손님의 주문을 받는다
  - 여러명이라면 계속…
- 새로운 스레드 만드는 방법
  1. Thread 클래스를 상속 받은 클래스를 정의 한다.
  2. run() 메소드를 오버라이드 한다.
  3. run() 메소드 안에서 새로운 스레드에서 해야 할 작업을 코딩한다.
  4. 만든 클래스로 객체를 생성하고 해당 객체의 start() 메소드를 호출하면 새로운 스레드가 시작된다.

```
package test.mypac;
public class CountThread extends Thread {

  @Override
  public void run() {
    //run 메소드 안쪽이 새로운 작업 단위가 된다.
    int count = 0;
```

```
    while (true) {
      try {
        Thread.sleep(1000);
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
      count++;
      System.out.println("현재 카운트:" + count);
      if (count == 10) {
        break;
      }
    }
  }
}
```

```
package test.main;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

import test.mypac.CountThread;

public class Frame02 extends JFrame implements ActionListener {

  // 생성자
  public Frame02() {
    // 레이아웃 설정
    setLayout(new BorderLayout());
    // 페널을 프레임의 상단에 배치
    JPanel panel = new JPanel();
    panel.setBackground(Color.YELLOW);
    add(panel, BorderLayout.NORTH);
    // 버튼을 페널에 추가 하고
    JButton countBtn = new JButton("1~10 까지 세기");
    panel.add(countBtn);
    // 버튼에 리스너 등록하기
    countBtn.addActionListener(this);
  }

  public static void main(String[] args) {
    // MyFrame 클래스를 이용해서 객체 생성하고 참조값을 지역변수 frame 에 담기
    Frame02 frame = new Frame02();
    // 프레임의 제목 설정
    frame.setTitle("Frame02");
    // 프레임을 닫으면 자동으로 프로세스가 종료 되도록 한다.
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setBounds(100, 100, 500, 500);
    frame.setVisible(true);
  }

  @Override
  public void actionPerformed(ActionEvent arg0) {
    new CountThread().start();
  }
}
```

- thred 객체는 1회용이다.
- 한번 run()메소드가 리턴하면 그 객체는 다시 사용할 수 없다.
- 새로운 스레드를 만드는 방법2
  1. Rannable 인터페이스를 구현한 클래스를 정의한다.
  2. run() 메소드를 강제 오버라이드 한다.
  3. Thread 클래스로 객체를 생성하면서 해당 클래스로 만든 객체를 생성자의 인자로 전달한다.
  4. Thread 클래스로 만든 객체의 start() 메소드를 호출해서 스레드를 시작 시킨다.

익명클래스로 바꾸면 새 클래스를 안만들어도 된다.

```
@Override
  public void actionPerformed(ActionEvent arg0) {
    Thread t = new Thread(new Runnable() {
      @Override
      public void run() {

      }
    });
    //아래의 내용이 즉시 콘솔창에 출력된다.
    System.out.println("새로운 스레드가 시작되었습니다.");
  }

}
```

어짜피 일회용인 스레드

```
@Override
public void actionPerformed(ActionEvent arg0) {
  new Thread(()->{
  }).start();
```

```
@Override
  public void actionPerformed(ActionEvent arg0) {
    new Thread(() -> {
      int count = 0;
      while (true) {
        try {
          Thread.sleep(1000);
        } catch (InterruptedException e) {
          e.printStackTrace();
        }
        count++;
        System.out.println("현재 카운트:" + count);
        if (count == 10) {
          break;
        }
```

```
      }
    }).start();
```

시간이 오래걸리거나 불확실한 작업은 스레드를 사용해야 한다!