



8_01_2022__java_UtilClass, Exception, Swing

Step12_UtilClass

~Step14_Swing

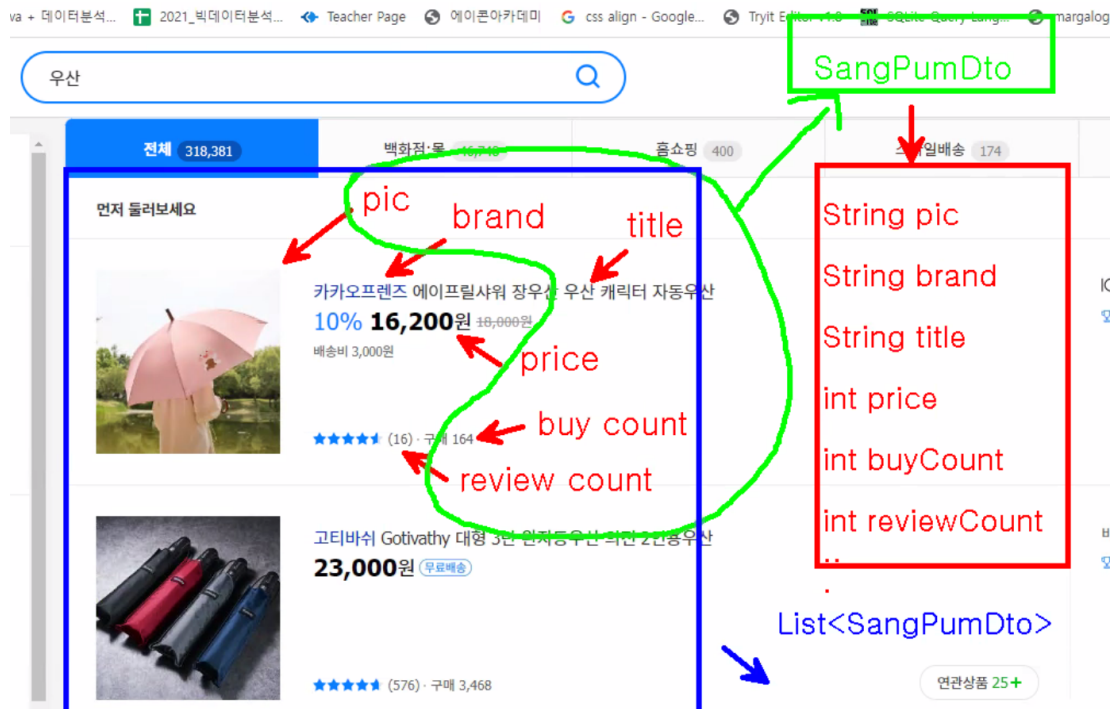
ArrayList

- 어떤 데이터를 순서대로 관리하고 싶을 때 사용하는 객체
- 주로 List 인터페이스 타입으로 받아서 사용한다.
- GenericClass를 결정 해야한다. <>

```
//숫자
List<Integer> list = new ArrayList<>();
//문자열
List<String> list = new ArrayList<>();
```

```
//여러가지 의 정보를 순서대로 관리하고 싶을때 사용
List<Member>
List<MemberDto>
List<Map>
```

- 구성
 - 회원 한명의 정보를 담을 DTO 클래스 설계 → MemberDto
 - 상품 한개의 정보를 담을 DTO 클래스 설계 → GoodsDto
 - 게시글 한개의 정보를 담을 DTO 클래스 설계 → BoardDto
 - 뉴스 한개의 정보를 담을 DTO 클래스 설계 → NewsDto



- .add , .get , .size 를 가장 많이 사용함
- 반복문 for을 사용, 참조해 정보를 불러옴
 - 확장 for문

```
for(int i=0; i<XXX.size(); i++){
String tmp=XXX.~~~}
==
for(String tmp:XXX){}
```

```
friends.add("주영이");
friends.add("덩어리");
```

```
//friends 의 size 만큼 반복문 돌면서
for(int i=0; i<friends.size(); i++) {
//i번째 방에 저장된 친구 이름 참조
String tmp=friends.get(i);
```

```
for(String tmp:friends){
```

- ArrayList는 기본 데이터 타입을 저장할 수 없다.
 - WrapperClass를 사용해야 함 (따로 new 하지 않아도 됨)

```
ArrayList<Integer> nums = new ArrayList<Integer>();
nums.add(10);
nums.add(20);
nums.add(30);
//확장 for 문을 이용해서 저장된 정수를 순서대로 콘솔창에 출력하기
for(Integer tmp : nums) {
```

```
System.out.println(tmp);  
}
```

- GenericClass를 타입으로 지정해줄 수 있다.

```
ArrayList<Car> cars = new ArrayList<Car>();
```

- Dto
 - Member
 - 접근 지정자 public
 - 아무데서나 사용 할 수 있음
 - MemberDto
 - 접근 지정자 private
 - setter, getter를 지정해 줘야 함

HashMap

- 순서대로 관리 안해도 됨
- 정보를 KeyValue로 관리 하고 싶을 때
- key generic, value generic 지정 해줘야함

```
HashMap<String, String> dic = new HashMap<>();
```

- 여러가지 타입을 섞어서 담고 싶으면 Object type을 지정해주면 됨

```
HashMap<String, Object> map = new HashMap<>();  
map.put("num", 1);  
map.put("name", "Dubu");  
map.put("addr", "Fairfax");
```

- 불러올 때도 Object 타입으로 리턴 되기 때문에 원래 타입으로 casting 해주어야 함

```
int num=(int)map.get("num");  
String name=(String)map.get("name");  
String addr=(String)map.get("addr");
```

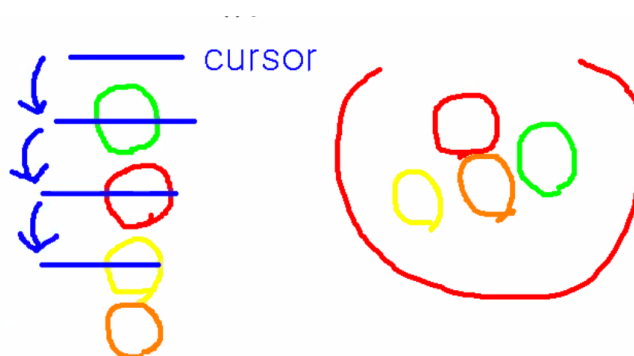
HashSet

- data를 묶음으로 관리하고 싶을 때 사용
- 순서, key값 없고 중복 불허용
- 외형상 무한루프

```
package test.main;

public class MainClass13 {
    public static void main(String[] args) {
        //외형상 무한 루프 이지만 특정 조건에서 탈출하는 반복문
        int count=0;
        //정확한 반복 횟수를 알 수 없을 때 사용할 수 있는 반복문
        while(true) {
            count++;
            System.out.println("반복횟수: "+count);
            if(count==1000) {
                break;//인접한 블록(반복문)탈출
            }
        }
    }
}
```

- Iterator(반복자)
 - 저장된 아이템을 일렬로 세운 것같은 모양을 만들



```
package test.main;

import java.util.HashSet;
```

```

import java.util.Iterator;
import java.util.Set;

public class MainClass14 {
    public static void main(String[] args) {
        Set<String> names = new HashSet<>();
        names.add("두부");
        names.add("유키");
        names.add("소주");
        names.add("재즈");
        names.add("달리");

        Iterator<String> it = names.iterator();
    }
}

```

합체예제

```

package test.main;

import java.util.HashSet;
import java.util.Iterator;
import java.util.Random;
import java.util.Set;

public class MainClass16 {
    public static void main(String[] args) {
        // 1. HashSet 객체를 생성해서 참조값을
        // 정수를 저장 할 수있는 Set 인터페이스 타입 변수에 담아보세요
        Set<Integer> numb = new HashSet<>();

        /*
         * 2. 반복문을 사용해 1~45 사이의 랜덤한 정수를 얻어내서 HashSet 객체에 담아보세요
         * 총 6개가 담길 때 까지 반복문이 사용되어야 합니다.
         * 주의) 우연히 이전에 나왔던 동일한 숫자가 나오면 HashSet은 하나만 저장하는걸 잊지 마세요
         */
        while (numb.size() < 6) {
            Random ran = new Random();
            int a = ran.nextInt(45) + 1;
            numb.add(a);
        }

        //선생님
        // 외형상 무한루프인 while문을 구성하고
        // while(true) {
        //     Random ran = new Random();
        //     int a = ran.nextInt(45) + 1;
        //     numb.add(a);
        // 만일 numb 의 사이즈가 6이 되면
        //     if(numb.size() == 6) {
        //         break; 반복문 탈출하기
        //     }
        // }
    }
}

```

```
// 3. HashSet 객체에 담긴 숫자를 콘솔창에 하나하나씩 모두 출력해 보세요.
Iterator<Integer> it = numb.iterator();
while(it.hasNext()) {
    int a = it.next();
    System.out.println(a);
}
}
```

Exception

```
package test.main;

import java.util.Scanner;

public class MainClass01 {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("숫자 입력 : ");
        //숫자 형식의 문자열을 입력 받는다 "10", "10.2" 등등
        String inputNum=scan.nextLine();

        //입력한 숫자를(문자열) 실제 숫자로 바꾼다.
        double num=Double.parseDouble(inputNum);
        //입력한 숫자에 100을 더한다.
        double result=num+100;
        System.out.println("입력한 숫자 + 100 : "+result);
        System.out.println("무언가 중요한 마무리 작업을 하고 main 메소드가 종료 됩니다.");
    }
}
```

위 클래스에서 문자열을 입력하면 exception 에러가 나온다

이런 경우에 다른 결과를 전달해 주고 싶으면 아래를 사용하면 된다.

```
try{
    에러나는 코드를 넣는다
}catch(에러내용 이름){
    에러대신 나올 메소드
}
```

```

package test.main;

import java.util.Scanner;

public class MainClass01 {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("숫자 입력 : ");
        //숫자 형식의 문자열을 입력 받는다 "10", "10.2" 등등
        String inputNum=scan.nextLine();

        try {
            //입력한 숫자를(문자열) 실제 숫자로 바꾼다.
            double num=Double.parseDouble(inputNum);
            //입력한 숫자에 100을 더한다.
            double result=num+100;
            System.out.println("입력한 숫자 + 100 : "+result);

        }catch(NumberFormatException nfe) {
            System.out.println("숫자로 입력하십시오!");
            System.out.println(nfe.getMessage());
            //콘솔에 자세한 경고 메시지 출력하기
            nfe.printStackTrace();
        }
        System.out.println("무언가 중요한 마무리 작업을 하고 main 메소드가 종료 됩니다.");
    }
}

```

결과

```

숫자 입력 :
구
숫자로 입력하십시오!
무언가 중요한 마무리 작업을 하고 main 메소드가 종료 됩니다.

```

(Exception e) ← 어떤 exception이든 처리할 수 있다.

- 프로그램의 흐름을 일정시간 잡아두는 방법

```

package test.main;

public class MainClass03 {
    public static void main(String[] args) {
        System.out.println("main 메소드가 시작 됩니다.");

        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

```

```

        System.out.println("main 메소드가 종료 됩니다.");
    }
}

```

- 반드시 try, catch로 묶어줘야함

• 파일 생성하기

```

package test.main;

import java.io.File;
import java.io.IOException;

public class MainClass04 {
    public static void main(String[] args) {
        File f=new File("C:\\Users\\HNJN-PC\\Desktop\\school\\java_work\\myFolder\\memo.txt");
        try {
            f.createNewFile();
        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.println("main 메소드가 종료 됩니다.");
    }
}

```

• Exception 을 만드는 방법

- 필요에 따라 예외 객체를 생성할 클래스를 직접 만들 수 있다.
- RuntimeException을 상속받으면 실행중에 발생하는 예외를 만들 수 있다.

```

package test.mypac;

public class SleepyException extends RuntimeException {
    //생성자
    public SleepyException(String msg) {
        //예외 메시지를 생성자의 인자로 전달 받아서 부모 생성자에 전달한다.
        super(msg);
        //전달된 예외 메시지는 나중에 .getMessage() 를 호출하면 리턴된다.
    }
}

```

```

package test.main;

```



```

import java.util.Random;

import test.mypac.SleepyException;

public class MainClass05 {
    public static void main(String[] args) {
        Random ran = new Random();

        for (int i = 0; i < 100; i++) {
            int ranNum = ran.nextInt(10);
            if (ranNum == 5) { // 우연히 랜덤한 정수가 5가 나오면 예외를 발생시킨다.
                // throw 예약어와 함께 예외 객체를 생성하면 예외가 발생한다.
                throw new SleepyException("너무 졸려~");
            }
            System.out.println(i + 1 + " 번째 작업중...");
        }
        System.out.println("main 메소드가 종료 됩니다.");
    }
}

```

- 호출하는 쪽에서 알아서 하게 만드는 방법
 - 메소드 안에서 발생하는 Exception을 던져버리는 경우
 - 메소드를 호출하는 쪽에서 해당 예외를 처리해야 한다.

```

public static void send() throws InterruptedException {
    System.out.println("5초동안 전송을 해요!");
    Thread.sleep(5000);
}

```

- 예제
 - myUtil 클래스

```

package test.mypac;

public class MyUtil {
    public static void draw() {
        System.out.println("5초동안 그림을 그려요!");

        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("그림 완성!");
    }
}

```

```

    }
    public static void send() throws InterruptedException {
        System.out.println("5초동안 전송을 해요!");
        Thread.sleep(5000);
    }
}

```

◦ 메인클래스

```

package test.main;

import test.mypac.MyUtil;

public class MainClass06 {
    public static void main(String[] args) {
        MyUtil.draw();

        try {
            MyUtil.send();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

```

import 받는 쪽에서 send메소드를 받아 try, catch 해주면 된다.

JAVA의 GUI

- 윈도우 창이 뜨는 프로그래밍을 하고 싶다

```

package frame01;

import javax.swing.JFrame;

public class MyFrame extends JFrame {

    public MyFrame(String title) {
        // 부모 생성자에 프레임의 제목 넘겨주기
        super(title);
        setBounds(100, 100, 500, 500);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        // 프레임의 레이아웃 매니저를 사용하지 않기 때문에 UI를 절대 좌표에 직접 배치해야 한다.
        setLayout(null);
        // 프레임이 화면상에 보이도록 한다.
        setVisible(true);
    }
}

```

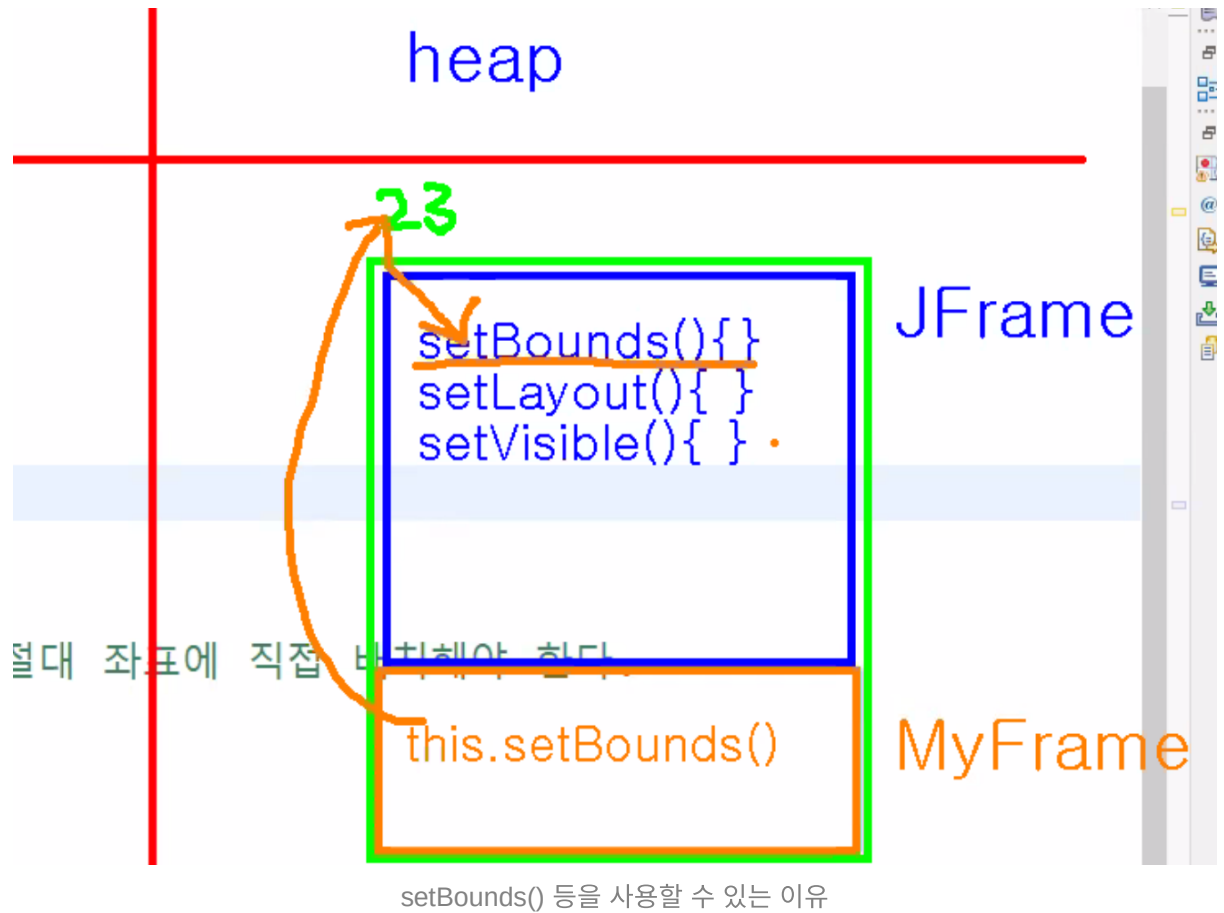
```

public static void main(String[] args) {

    new MyFrame("나의 프레임");

}
}

```



```

this.setBounds()
//this. 이 생략되어있음.

```

```

package frame01;

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

public class MyFrame extends JFrame {

    public MyFrame(String title) {
        // 부모 생성자에 프레임의 제목 넘겨주기
        super(title);
        // setBounds(x,y, width, height)
        setBounds(100, 100, 500, 500);
    }
}

```

```

// 창을 닫았을 때 프로세스도 같이 종료되도록 한다.
setDefaultCloseOperation(EXIT_ON_CLOSE);
// 프레임의 레이아웃 매니저를 사용하지 않기 때문에 UI를 절대 좌표에 직접 배치해야 한다.
setLayout(null);

JButton btn1 = new JButton("버튼1");
// 버튼의 위치
btn1.setLocation(10, 10);
// 버튼의 크기
btn1.setSize(100, 30);
add(btn1); // 프레임에 btn1 추가하기

JButton btn2=new JButton("버튼2");
//setLocation(), setSize() 메소드 대신에 한번에 설정할 수 있는 메소드
btn2.setBounds(120, 10, 100, 30);
add(btn2); //프레임에 btn2 추가하기

JButton btn3=new JButton("버튼3");
btn3.setBounds(230, 10, 100, 30);
add(btn3);

// 프레임이 화면상에 보이도록 한다.
setVisible(true);
}

public static void main(String[] args) {

    new MyFrame("나의 프레임");

}
}

```

- 레이아웃 매니저를 사용하는 경우

```

package frame02;

import java.awt.FlowLayout;

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

public class MyFrame extends JFrame {

    public MyFrame(String title) {
        // 부모 생성자에 프레임의 제목 넘겨주기
        super(title);
        // setBounds(x,y, width, height)
        setBounds(100, 100, 500, 500);
        // 창을 닫았을 때 프로세스도 같이 종료되도록 한다.
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        // 프레임의 레이아웃 매니저를 사용하지 않기 때문에 UI를 절대 좌표에 직접 배치해야 한다.

        FlowLayout layout = new FlowLayout(FlowLayout.CENTER);
        setLayout(layout);
    }
}

```

```

        JButton btn1 = new JButton("버튼1");
        add(btn1);
        JButton btn2=new JButton("버튼2");
        add(btn2);
        JButton btn3=new JButton("버튼3");
        add(btn3);

        setVisible(true);

    }

    public static void main(String[] args) {

        new MyFrame("나의 프레임");

    }
}

```

```

package frame03;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.LayoutManager;

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

public class MyFrame extends JFrame {

    public MyFrame(String title) {
        // 부모 생성자에 프레임의 제목 넘겨주기
        super(title);
        // setBounds(x,y, width, height)
        setBounds(100, 100, 500, 500);
        // 창을 닫았을 때 프로세스도 같이 종료되도록 한다.
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        // 프레임의 레이아웃 매니저를 사용하지 않기 때문에 UI를 절대 좌표에 직접 배치해야 한다.

        setLayout(new GridLayout(2, 2));

        JButton btn1 = new JButton("버튼1");
        add(btn1);
        JButton btn2=new JButton("버튼2");
        add(btn2);
        JButton btn3=new JButton("버튼3");
        add(btn3);

        setVisible(true);
    }
}

```

```

    }

    public static void main(String[] args) {

        new MyFrame("나의 프레임");

    }
}

```

```

package frame04;

import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.LayoutManager;

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

public class MyFrame extends JFrame {

    public MyFrame(String title) {
        // 부모 생성자에 프레임의 제목 넘겨주기
        super(title);
        // setBounds(x,y, width, height)
        setBounds(100, 100, 500, 500);
        // 창을 닫았을 때 프로세스도 같이 종료되도록 한다.
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        // 프레임의 레이아웃 매니저를 사용하지 않기 때문에 UI를 절대 좌표에 직접 배치해야 한다.

        //BorderLayout 객체 사용하기
        setLayout(new BorderLayout());

        JButton btn1 = new JButton("버튼1");
        add(btn1);
        JButton btn2=new JButton("버튼2");
        add(btn2);
        JButton btn3=new JButton("버튼3");
        add(btn3);

        setVisible(true);

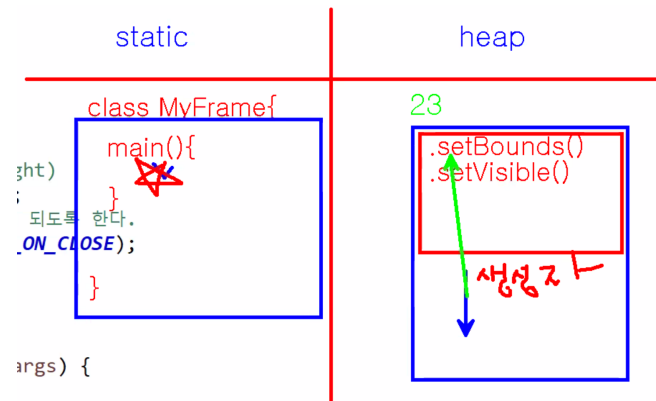
    }

    public static void main(String[] args) {

        new MyFrame("나의 프레임");

    }
}

```



```

1 package frame05;
2
3 import java.awt.BorderLayout;
4
5 public class MyFrame extends JFrame {
6
7     public MyFrame(String title) {
8         // 부모 생성자에 프레임의 제목 넘겨주기
9         super(title);
10        // setBounds(x,y, width, height)
11        setBounds(100, 100, 500, 500);
12        // 창을 닫았을 때 프로세스도 같이 종료되도록 한다.
13        setDefaultCloseOperation(EXIT_ON_CLOSE);
14        setVisible(true);
15        this.
16    }
17
18    public static void main(String[] args) {
19        JFrame f = new MyFrame("frame05");
20        f.setVisible(true);
21        f.setBounds(100, 100, 500, 500);
22        f.setDefaultCloseOperation(EXIT_ON_CLOSE);
23    }
24 }

```

A

```
package frame05;

import java.awt.BorderLayout;

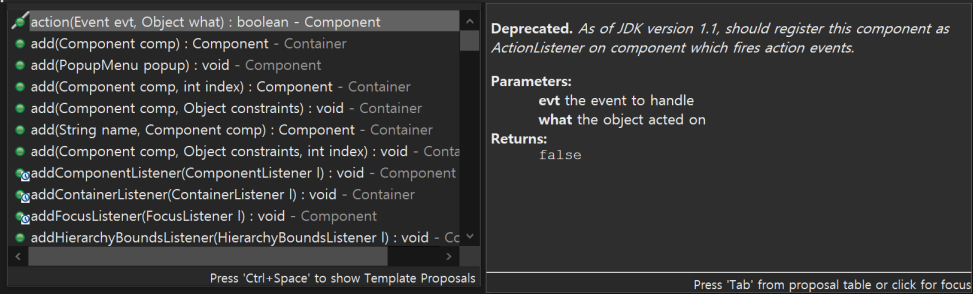
public class MyFrame extends JFrame {

    public MyFrame(String title) {
        // 부모 생성자에 프레임의 제목 넘겨주기
        super(title);
        // setBounds(x,y, width, height)
        setBounds(100, 100, 500, 500);
        // 창을 닫았을 때 프로세스도 같이 종료되도록 한다.
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setVisible(true);
    }

    public static void main(String[] args) {

        JFrame f = new MyFrame("나의 프레임");
        f.

    }
}
```



B

A와 B들이 같다



창에 여러 버튼 만들고 버튼마다 다른 메시지 출력되게 만들기
Frame05\\MyFrame.java