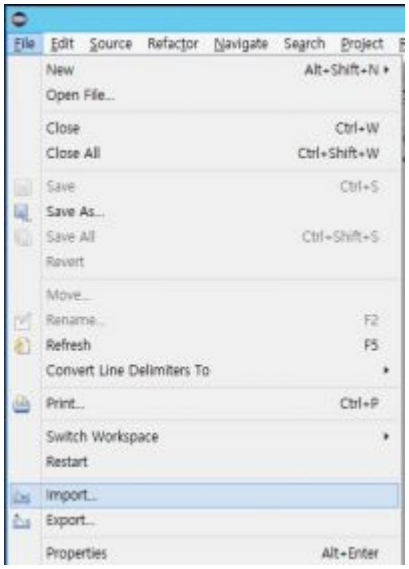
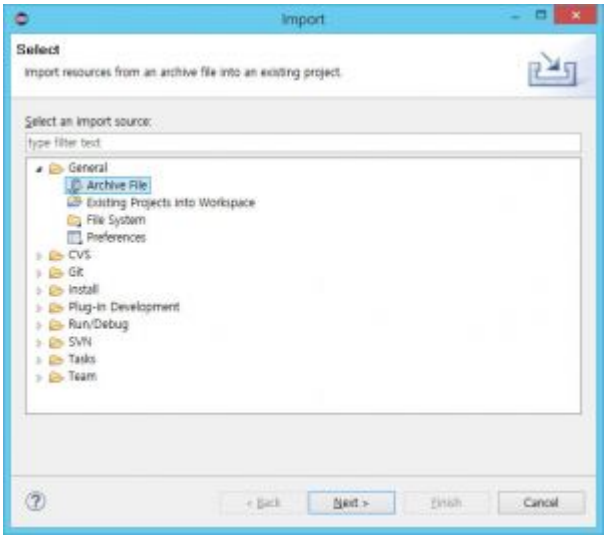
Java SWT(Standard Widget Toolkit) Library

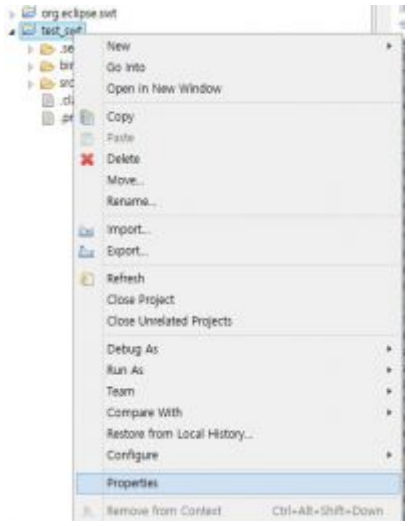
라이브러리 적용방법

- http://eclipse.org/swt 에서 다운로드
- 다운로드 링크
  - windows-32bit
  - windows-64bit
- Eclipse를 실행하여 File-Import 메뉴를 선택하여 Archive File 선택후 next

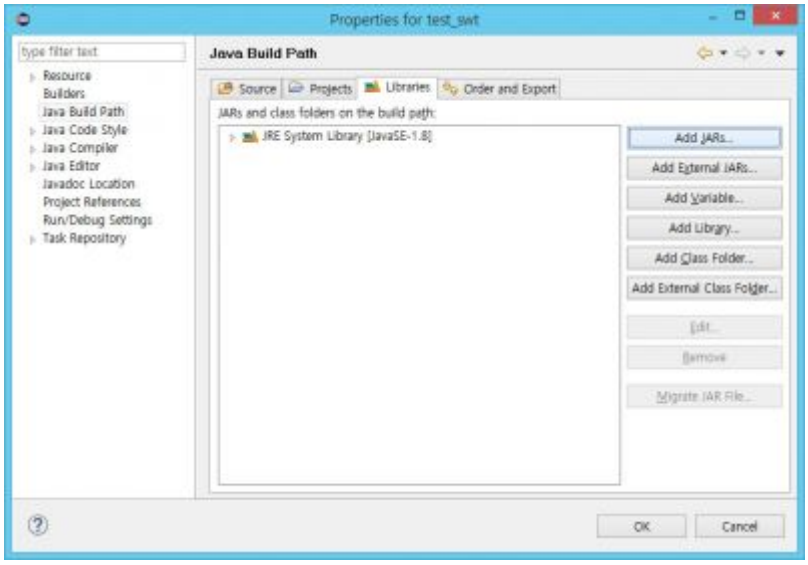




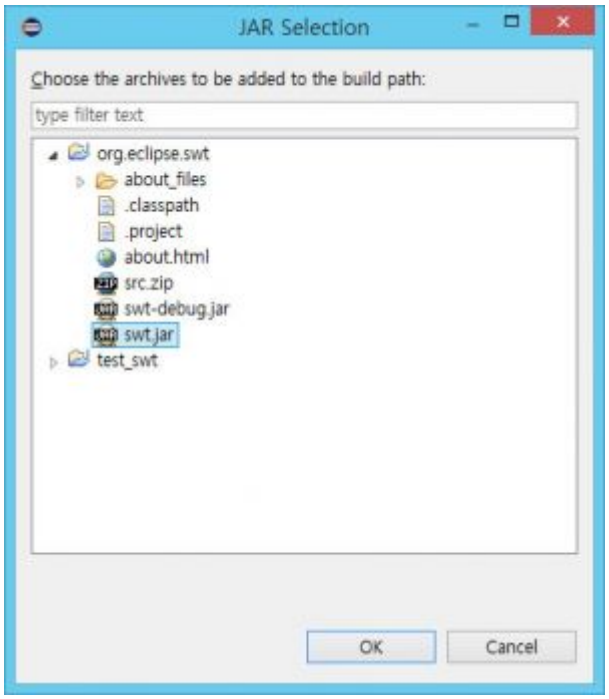
- 다운받은 압축파일을 선택하여 Finish
- 프로젝트 목록에 org.eclipse.swt가 추가된 것을 확인 후 새로운 자바 프로젝트 생성
- 생성한 프로젝트를 오른쪽 클릭하여 Properties 선택



- Java Build Path 항목의 Libraries 탭에서 Add JARs 를선택



• org.eclipse.swt 프로젝트 하위에 있는 swt.jar 파일을 선택, 확인



• 이후 프로젝트에서 라이브러리를 import하여 컨트롤 사용가능

구현방법

• swt에는 Display라는 객체가 하나 존재해야 하며, 운영체제의 메시지를 받아 하위 쉘들에게 전달하는 역할을 한다.
• 쉘이란, GUI창을 말하며, 쉘 하위에 여러 컨트롤을 담아 사용할 수 있다.
• 간단한 swt 예제 코드

```
public static void main(String[] args)
{
    Display display = new Display(); //Display 객체 생성
    Shell shell = new Shell(display); //Shell 객체 생성

    Text helloText = new Text(shell, SWT.CENTER); //가운데정렬 옵션을 가진 Text객체 생성, shell 하위에 존재
    helloText.setText("Hello SWT!");
    helloText.pack(); //컨트롤 크기에 맞게 크기 자동조정

    Button button = new Button(shell, SWT.PUSH); //누름 옵션을 가진 버튼 객체 생성
    button.setText("Button test");
    button.setLocation(new Point(0, 20)); //버튼의 위치 설정
    button.setSize(new Point(100, 50)); //pack 메소드로 크기조정을 하지 않을경우에는 수동조정이 필요

    Composite composite = new Composite(shell, SWT.BORDER);
    composite.setLocation(new Point(10, 80));
    FillLayout composite_layout = new FillLayout(SWT.VERTICAL);
    composite.setLayout(composite_layout);

    /*=============================================================*/
    /*   composite 하위에 있는 객체 생성 레이아웃이 존재하면 size나 location이 자동으로 결정    */
    Label label1 = new Label(composite, SWT.NONE); //특별한 옵션이 필요없는경우 스타일 파라미터값 : SWT.None
    label1.setText("label test");

    Button button1 = new Button(composite, SWT.PUSH);
    button1.setText("Button test");
    /*=============================================================*/

    composite.pack(); //composite 내부에 존재하는 컨트롤들을 모두 담을 수 있는 크기로 자동조정

    shell.pack();
    shell.open();
    /*=============================================================*/
    /*                  프로그램이 종료하는것을 기다리기 위한 블록. 반드시 필요                */
    while(!shell.isDisposed())
    {
        if(!display.readAndDispatch())
            display.sleep();
    }
    display.dispose();
    /*=============================================================*/
}
```

- 실행 결과

- **컨트롤**
  - Label
    - 가장 간단한 형태로, 텍스트를 화면에 표시하고자 하는 경우 사용
  - Text
    - 사용자가 입력할 수 있는 텍스트박스

- Button
  - 사용자가 클릭할 수있는 버튼
- List
  - 여러 항목을 리스트로 볼 수 있는 컨트롤
- Combo
  - 여러 항목을 콤보박스 형태로 볼 수 있는 컨트롤
- Canvas
  - 자유로운 도형을 그릴 수 있는 컨트롤

컴포지트 없이 컨트롤을 사용하는 경우는 default 크기가 0이기 때문에 setSize 메소드를 통해 크기 설정을 해줘야함

- **Composite**
  - 일정한 크기를 가지고 다른 컨트롤들을 포함하는 클래스
  - Shell 클래스 또한 Composite를 상속받은 클래스이다.
  - 일반적인 Composite 생성자
    - public Composite(Composite parent, int style)
    - 대부분의 컨트롤들은 생성자의 인자에서 부모 Composite를 요구함
    - Composite의 레이아웃을 설정하게 되면 하위 컨트롤들은 레이아웃에 따라 자동적으로 배치됨
    - 레이아웃 정의 메소드
      - setLayout(레이아웃 객체);

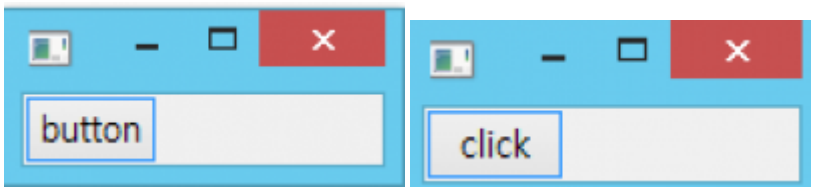
```
Composite test_composite = new Composite(shell, SWT.NONE);
FillLayout composite_layout = new FillLayout();
test_composite.setLayout(composite_layout);
```

- 레이아웃의 종류
  - FillLayout
    - 모든 하위 컨트롤들을 공백없이 Composite에 가득 차게 그리는 방식
    - 기본값은 가로정렬로 되어 있으며, 생성자 style 파라미터로 설정을 바꿀 수 있다.
  - RowLayout
    - 한줄로 쭉 늘여놓는 방식(컨트롤마다 크기가 다를 수 있다)
  - GridLayout
    - 일반적으로 생각할 수 있는 table 형태의 레이아웃
  - FormLayout
    - 상대적인 위치를 이용하여 그리는 방식

- **Event Handler**
  - 버튼 클릭 등의 이벤트 동작을 받기 위한 클래스
  - add+EventHandler 메소드를 통해 핸들러를 등록하여 해당 이벤트 동작과 연결
  - 하나의 컨트롤에 여러 개의 Event Handler 등록 가능
  - 예제 소스코드

```
private static Button button;
public static void main(String[] args)
{
    Display display = new Display(); //Display 객체 생성
    Shell shell = new Shell(display); //Shell 객체 생성

    button = new Button(shell, SWT.NONE);
    button.setText("button");
    button.pack();

    button.addMouseListener(new MouseListener()
    {
        public void mouseDoubleClick(MouseEvent e)
        {

        }

        public void mouseDown(MouseEvent e)
        {

        }

        public void mouseUp(MouseEvent e)
        {
            button.setText("click");
        }
    });
```

• 실행결과



• **Canvas**
  ◦ 자유도형을 그리기 위한 컨트롤
  ◦ PaintEventListener에 정의된 내용대로 그리게 된다.
  ◦ PaintEventListener에서 gc 객체를 사용하여 도형을 그린다.
  ◦ 예제 소스코드

```
Canvas canvas = new Canvas(shell, SWT.NONE);
canvas.setBackground(new Color(display, 255,255,255));
canvas.setSize(new Point(100,100));

canvas.addPaintListener(new PaintListener()
{
    public void paintControl(PaintEvent e)
    {
        e.gc.drawRectangle(10,10,30,30);
        e.gc.drawOval(10, 50, 40, 40);
        e.gc.drawLine(60, 10, 90, 20);
        int[] vector = {60, 40, 50, 60, 70, 90, 90, 80, 70, 60};
        e.gc.drawPolygon(vector);
    }
});
```

- 실행결과