Java Advanced

교육 일정

1일	2일	3일	4일
객체지향 프로그래밍 개요 Inner class	Thread Wrapper class	JDBC Programming	객체지향 모델링 객체지향 설계원칙
Exception Handling Java I/O	Collection API Generic Programming	JDBC Programming	UML과 모델링 디자인 패턴

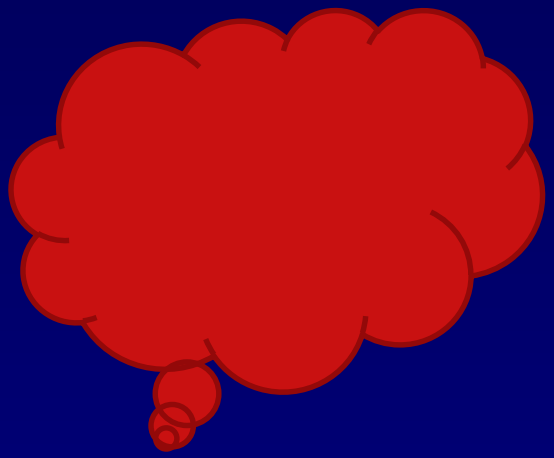
객체지향 원리

- 1. 추상화(Abstraction)
- 2. 캡슐화(Encapsulation)
- 3. 다형성(Polymorphism)

1-1.추상화(Abstraction)

- 사물의 공통된(추상적) 특징을 파악해 인식의 대상으로 삼는 행위
- 개체들이 소유한 공통의 보편적 특성의 이름으로 하나의 집합을 이룸
- 집합을 구성하는 개체들을 “일반화 " 하는 것

1-2.추상화의 예



1-3.추상화의 필요성

- 각 개체의 구체적인 개념에 의존하지 말고 추상적 개념에 의존해야 유연한 설계가 가능

2-1.캡슐화(Encapsulation)

- 객체의 내부 기능 구현을 외부에 감추는 것
- 캡슐화를 통해 낮은 결합도와 높은 응집도 구현
- 요구사항 변경에 대한 유연한(flexible) 대처 가능

2-2.캡슐화를 위한 규칙

1. Tell, don’t ask!

2. Law of Demeter
   - 메소드에서 생성한 객체의 메소드만 호출
   - 파라메터로 받은 객체의 메소드만 호출
   - 필드로 참조하는 객체의 메소드만 호출

3-1.다형성(polymorphism)

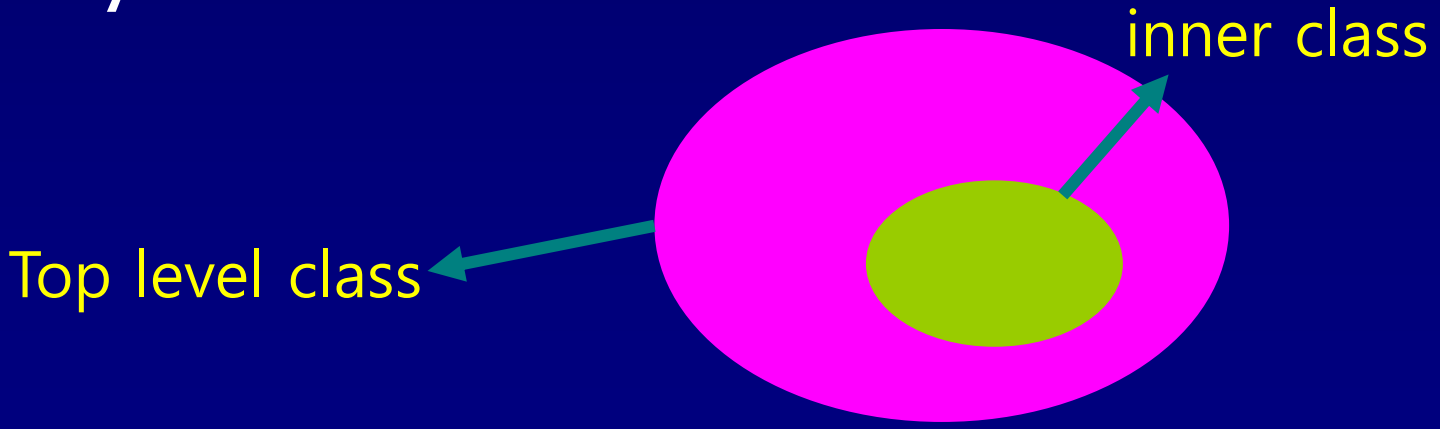
- 한 객체가 여러개(poly)의 모습(morph)을 가짐
- 한 객체가 여러 타입을 가질 수 있음
- 상속을 통해 가능

Inner class

Inner class

Inner Class의 종류

1. static class
2. member class
3. local class 제일 많이 사용 안됨
4. anonymous class 무명클래스. 자주 사용됨



Exception Handling

Exception

- 프로그램 실행 중 발생하는 문제 상황
- Ex) 데이터베이스 연결 시
  파일 입/출력시
  배열의 잘못된 인덱스 접근

Exception / Error

- Error
  - 복구 불가능한 문제 상황
  - StackOverFlowError, NoSuchMethodError
- Exception
  - 복구 가능한 문제 상황
  - NullPointerException, IOException
  - RuntimeException/Non-RuntimeException

Exception Handling

- 1. 예외가 발생한 곳에서 직접 처리

  try~catch~finally

- 2. 호출자로 보내서 예외처리 위임

  throws

Exception Handling -1

```
try{
        예외 발생 코드
}
```

```
catch(예외클래스명  변수명){
        예외 처리 코드
}catch(예외클래스명  변수명){
        예외 처리 코드
}
```

```
finally{
        예외 발생 여부와 관계없이 항상 실행될 코드
}
```

Exception Handling -2

- 메소드 선언부에 throws 구문을 이용하여 호출자측에 발생 가능한 예외를 알림

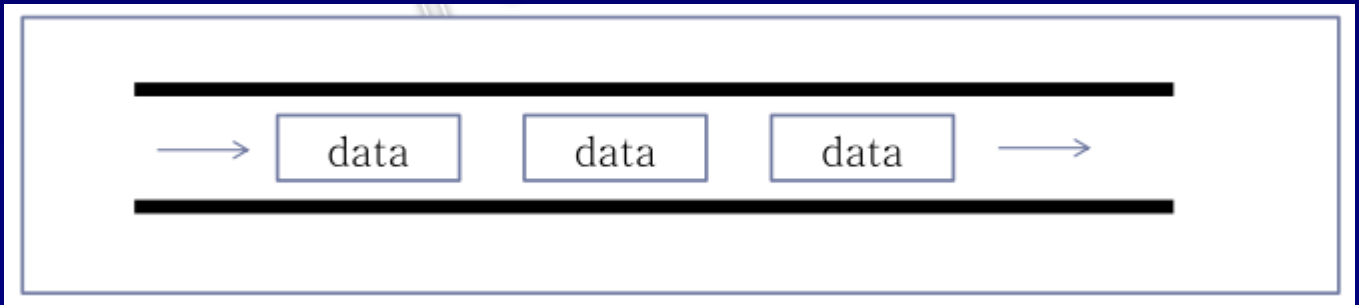
```
public void method() throws 예외클래스명, 예외클래스명
{


}
```

I/O Stream

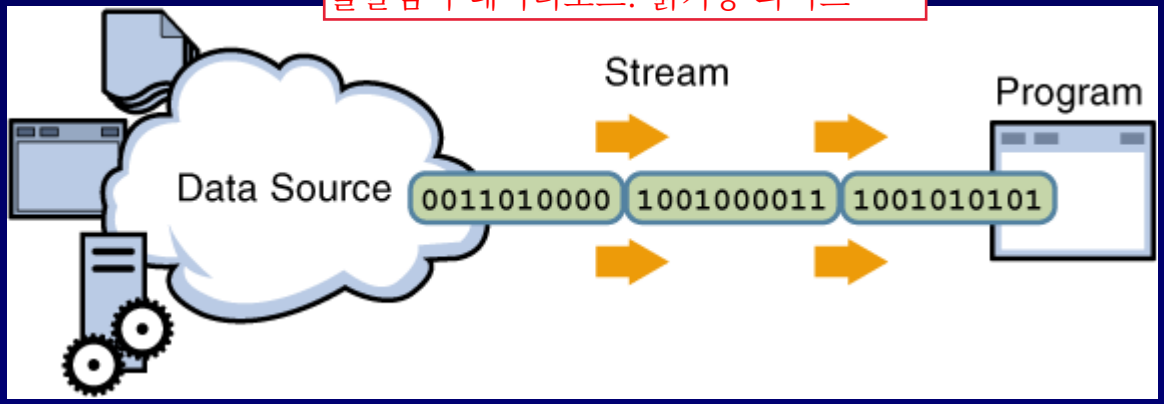
Stream

: 일련의 데이터를 이동시키는 입출력 파이프 혹은 그 데이터

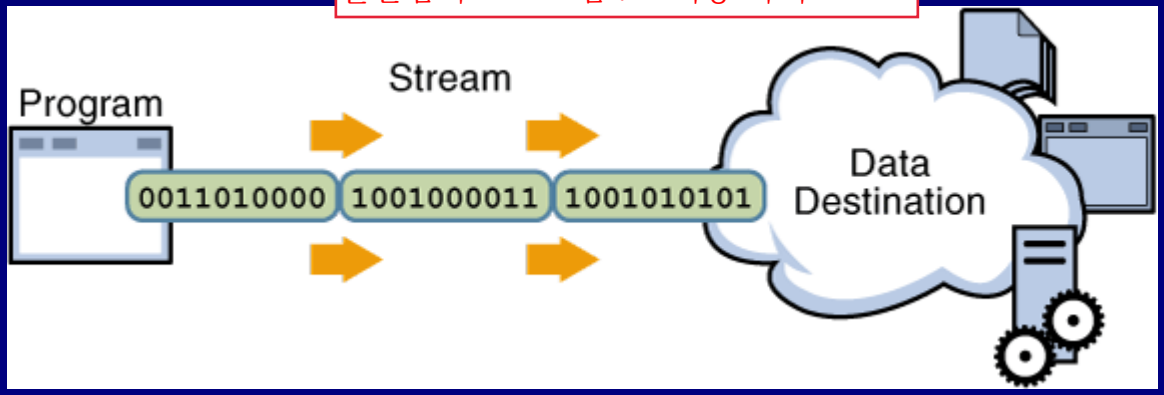


입출력 stream

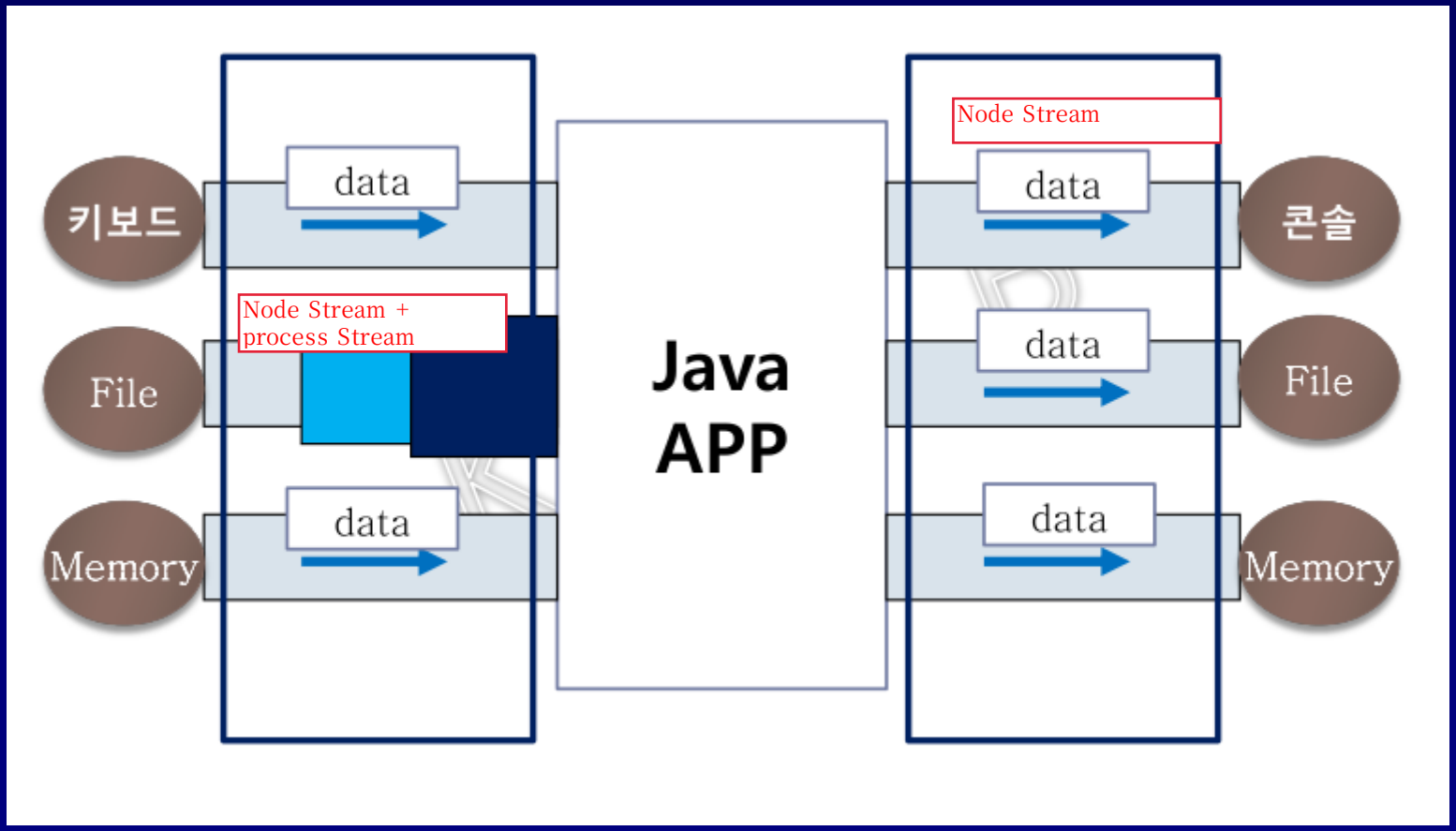
출발점이 데이터소스: 읽기용 파이프



출발점이 프로그램 : 쓰기용 파이프



입출력 stream


키보드
File
Memory
data
Node Stream +
process Stream
data
Java
APP
Node Stream
data
data
data
콘솔
File
Memory

Stream 종류

구분	Stream 종류
입력용/출력용	InputStream
	OutputStream
입출력 단위 (char / byte)	Character Stream
	Byte Stream
데이터입출력방법 (직/간접)	Node(Sink) Stream
	Process Stream

~reader 로 끝남

~stream 로 끝남

~직접 읽고 쓰고 쏨

~직접 읽고 쓰고 하지 않고 가공을 해줌

Node(Sink) Stream

근원지/ 목적지	Byte stream	Character stream
memory	ByteArrayInputStream ByteArrayOutputStream	CharArrayReader CharArrayWriter
file	FileInputStram FileOutputStream	FileReader FileWriter

Process Stream

여러가지 원하는 작업이 있을 때, 변환한다거나, 속도를 빠르게 한다거나, 객체를 주고받는다거나, node 스트림이 갖고 있지 않는 메서드를 사용한다거나

특별한 목적이 있을 때 필터로서 사용

처리	Byte Stream	Character Stream
버퍼링 속도가 느릴때, 버퍼를 하나 붙여서 버퍼 사이즈만큼 팍팍 읽어올 수 있도록 하는 프로세스(필터) 스트림	BufferedInputStream BufferedOutputStream	BufferedReader BufferedWriter
자료변환 **(기본자료형인식)**	DataInputStream DataOutputStream	
객체직렬화 **(객체입출력)**	ObjectInputStream ObjectOutputStream	

GUI (Graphic User Interface)

GUI

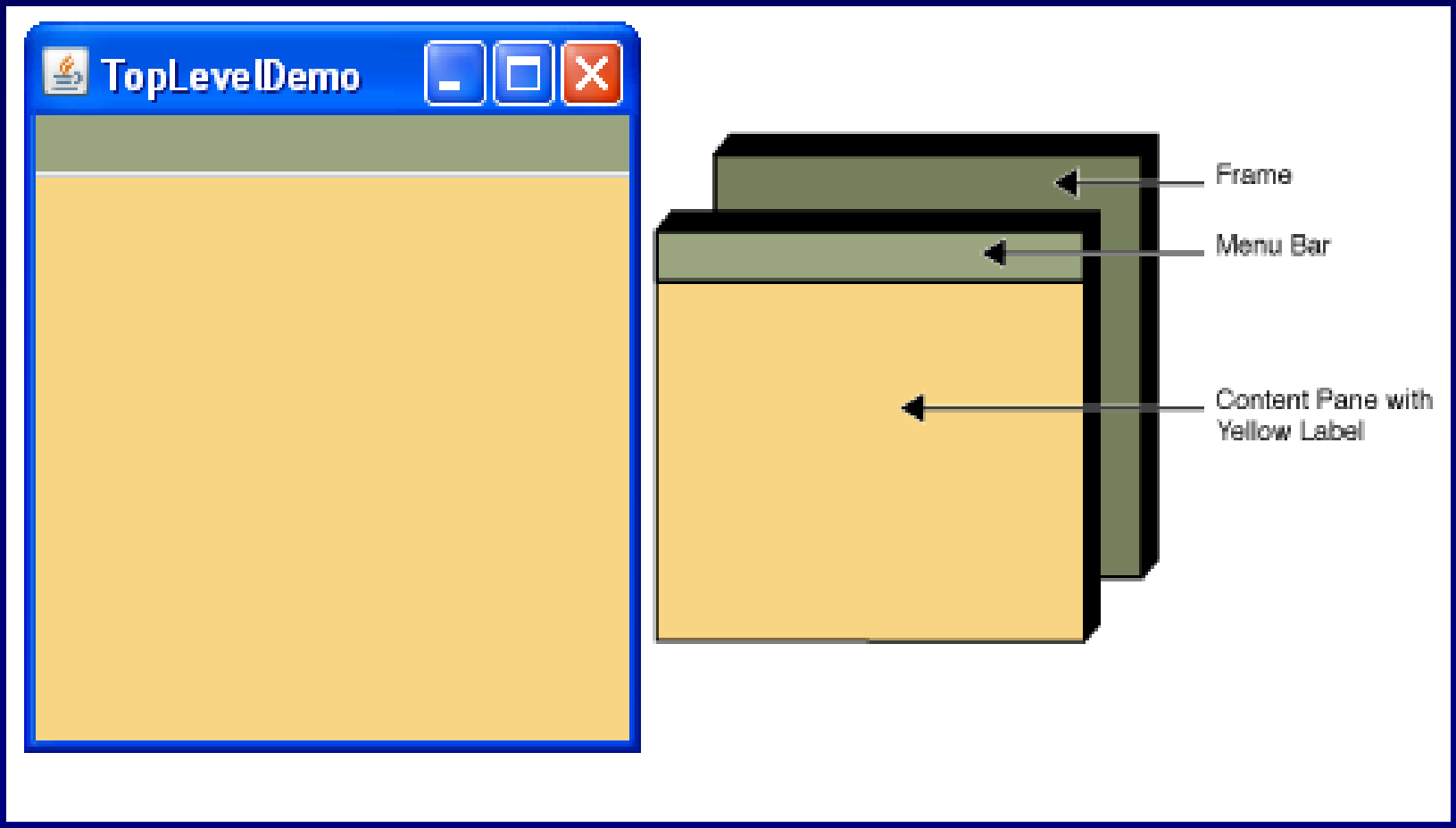
AWT

- JDK 초기 UI API
- 컴포넌트가 C와 같은 NATIVE CODE를 포함하고 있음

SWING

- 다양한 종류의 컴포넌트
- 컴포넌트가 100% JAVA로 작성되어 짐
- 어떤 실행 환경에서든 동일한 UI

JFrame의 구성


TopLevelDemo
Frame
Menu Bar
Content Pane with
Yellow Label

Layout Managers

- FlowLayout
- BorderLayout
- GridLayout
- GridBagLayout
- CardLayout
- BoxLayout

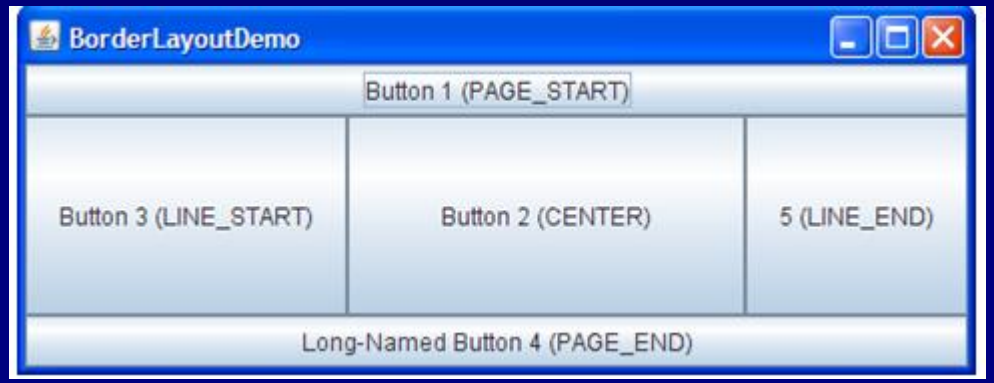
more

FlowLayout

- 화면의 상단에서부터 중앙 정렬되어 오른쪽 방향으로 배치됨
- 화면 크기가 변하면 배치가 달라짐

BorderLayout

- 화면이 동,서,남,북,중앙으로 나누어져 있음
- 화면 크기가 변해도 배치가 달라지지 않음

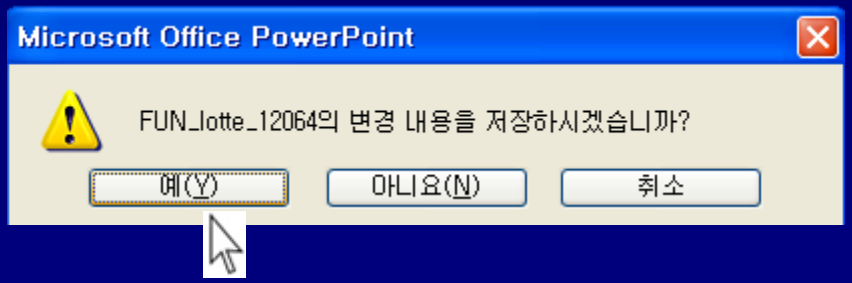


GridLayout

- 화면이 행렬 방식의 격자로 나누어져 있음
- 화면 크기가 변해도 배치가 달라지지 않음

Event Handling

- 화면을 구성하고 있는 컴포넌트들에 대해 발생하는 사건(이벤트)에 대한 처리작업



Event Handling

- Event Handler
  → Event 감시/처리자
  - Interface 타입으로 존재 → 추상메소드
  - Handler 이름 → ~Listener
  - java.awt.event / javax.swing.event

Event Handling

Component	Event	Listener
JButton, JTextField	ActionEvent	ActionListener
JList	ListSelectionEvent	ListSelectionListener
Window	WindowEvent	WindowListener

Event Handling 처리순서

1. 이벤트 처리 클래스 정의
   → Listener(감시자) implements 하기
2. Event발생 컴포넌트에 Listener 등록
3. Listener의 추상 메소드 구현
   → Event 발생시 원하는 처리작업 기술

JMenuBar, JMenu, JMenuItem