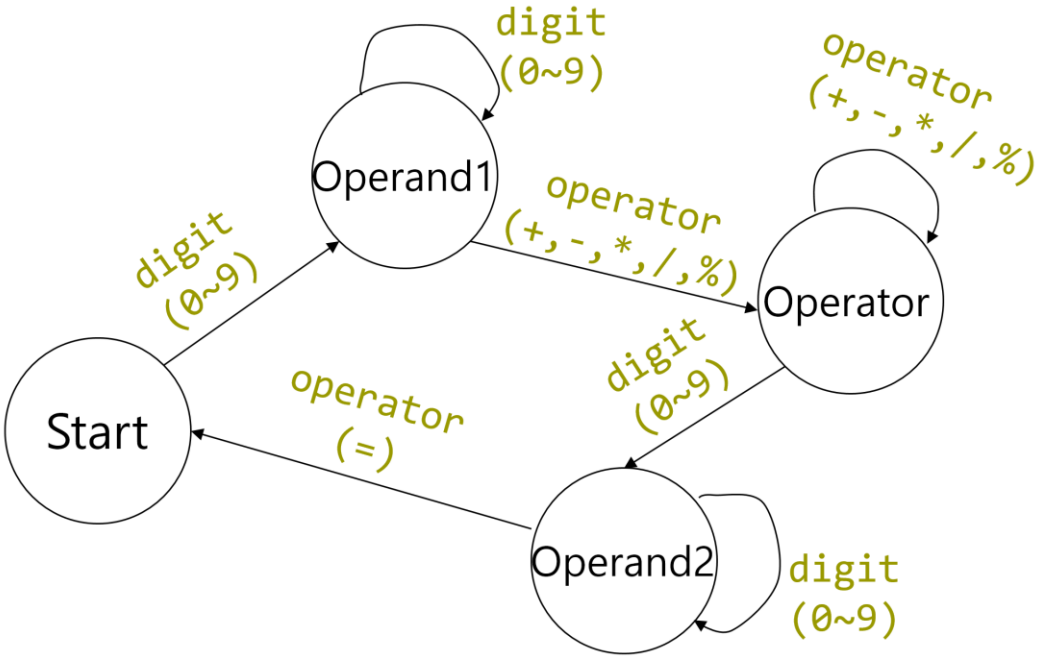
# 고급객체지향프로그래밍
# 실습과제 #09

---

조용주
ycho@smu.ac.kr

# 실습 과제 #09

- 간단한 산술 연산이 가능한 계산기 프로그램을 스테이트 패턴을 이용해서 구현
  - 정수 범위 내에 있는 숫자는 계산 가능
  - 사칙 연산과 나머지 연산(+, -, *, /, %)
- State Machine



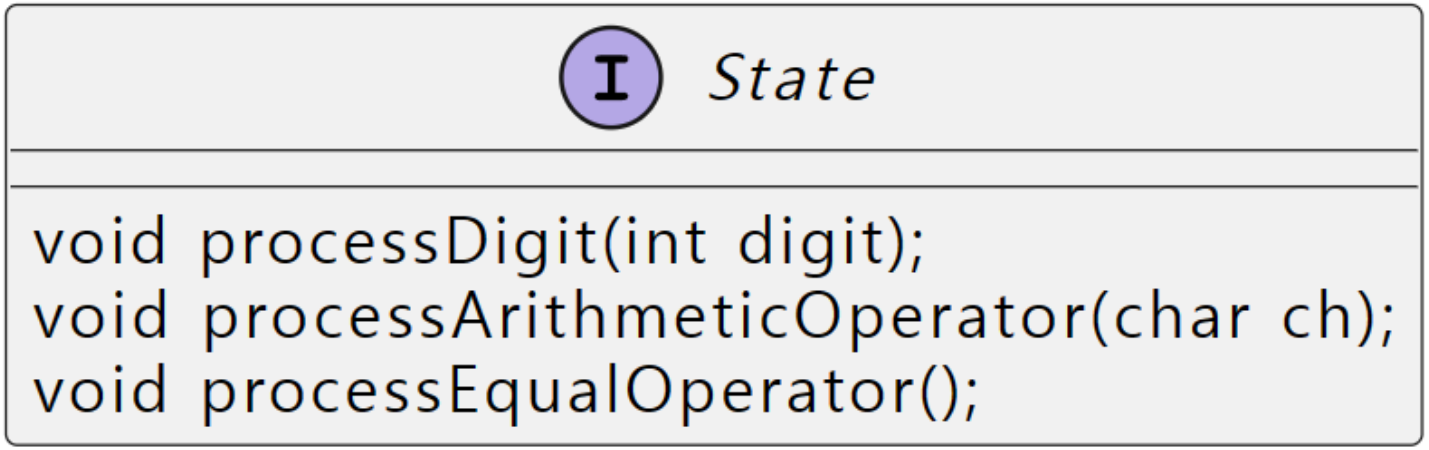
# 실습 과제 #09

- 상태에서 다른 상태로의 이동은 설명 내용에 해당되는 입력이 들어올 때 일어나며, 사용자는 숫자(들), 산술 연산자, 숫자(들), '=' 순서로 정확하게 입력한다고 가정할 것

| 상태 | 설명 |
|---|---|
| Start | 시작 상태, 피연산자 2개를 저장하는 변수는 0으로 초기화 |
| Operand1 | 첫 번째 피연산자 입력 상태 |
| Operator | 연산자 입력 상태. Operator 상태에서 다시 연산자가 입력되면 기존 연산자를 덮어 씀(overwrite) |
| Operand2 | 두 번째 피연산자 입력 상태<br>'='가 입력되면 결과값을 계산해서 저장한 후 Start 상태로 이동 |

# 실습 과제 #09

- State 인터페이스



# 실습 과제 #09

- 구현 클래스
  - Calculator
    - 계산기 기능 담당
    - State 생성 및 상태 저장
    - 피연산자와 연산자 저장
    - 산술 연산 기능(함수로 구현)
    - 산술 연산 결과를 저장한 후, main() 함수에서 결과를 요청하면 반환

# 실습 과제 #09

- Main 클래스
  - 주어진 코드를 완성
  - 사용자로부터 한 글자씩 입력받고 숫자, 산술 연산자, '=', 'q', 'Q'가 아닌 다른 글자가 입력되면 "invalid input"을 출력하고 계속 실행
  - 'q'나 'Q'를 입력하면 프로그램 중단
  - '='를 입력받으면 상태 기계를 이용해서 계산 후, 결과를 다음 형태로 출력한 후 계산기를 초기화(객체를 새로 생성하는 것이 아니라, Calculator 객체 내부의 피연산자나 결과값을 0으로 초기화하고 새로 계산할 수 있도록 함

```
숫자 연산자 숫자 = 결과값
```

# 실습 과제 #09

- 출력 예시

```
2
3
+
3
2
4
=
23 + 324 = 347
3
2
*
3
=
32 * 3 = 96
```

# 실습 과제 #08

- 주어진 파일 중에서 다음 파일들은 수정불가
  - Executable 클래스
  - Linker 인터페이스 (자세한 내용은 Linker.java의 주석 참조)
- 주어진 파일 중에서 다음 파일들은 수정해야 함
  - **Main 클래스**
  - **Builder 인터페이스** (자세한 내용은 Builder.java의 주석 참조)
- Main 클래스를 실행했을 때 화면에 출력되는 내용은 "실행 결과.txt"파일에 있음

# 실습 과제 #08 – 구현해야 하는 클래스

- **SourceCode 클래스**
  - C 코드 파일 이름(확장자 .c)을 멤버 변수로 포함
  - 파일 이름은 생성자의 인자로 전달됨
    - 예: new SourceCode("a.c")
  - 필요한 get/set 함수 추가할 것

- **ObjectCode 클래스**
  - Obj 파일 이름(확장자 .obj)을 멤버 변수로 포함
  - 파일 이름은 생성자의 인자로 전달됨
    - 예: new ObjectCode("a.obj")
  - 필요한 get/set 함수 추가할 것

# 실습 과제 #08 – 구현해야 하는 클래스

- **Preprocessor 인터페이스**
  - SourceCode를 입력으로 받고 새로운 SourceCode 객체를 반환하는 preprocess() 함수 포함
  - preprocess()에서 하는 일
    - "Preprocessing code: 파일_이름"을 화면에 출력
      - 파일_이름은 인자로 주어진 SourceCode 파일 이름
    - 주어진 SourceCode 파일 이름 앞에 "preprocessed_"를 붙여서 새로운 파일 이름을 가진 SourceCode 객체 생성
    - "Generating a new C code: 파일_이름"을 화면에 출력
      - 파일_이름은 새로 생성된 소스 코드의 파일 이름 (preprocessed_가 붙은 c 파일 이름)
    - 새로 생성된 SourceCode 객체 반환

# 실습 과제 #08 – 구현해야 하는 클래스

- **C언어 프리프로세서 클래스(이름은 본인이 정할 것)**
  - Preprocessor를 구현 (.c 확장자를 가진 파일 이름을 받아서 preprocessed_를 붙인 소스 코드를 생성)
    - 예
      - 인자로 전달된 SourceCode가 a.c라는 파일 이름을 가지고 있다면 preprocess() 함수가 호출되면 다음 내용이 화면에 출력되고 preprocessed_a.c 파일 이름을 가진 새로운 SourceCode 객체가 반환됨
      - 화면 출력 내용

```
Preprocessing code: a.c
Generating a new code: preprocessed_a.c
```

# 실습 과제 #08 – 구현해야 하는 클래스

- **Compiler 인터페이스**
  - SourceCode를 입력으로 받고 ObjectCode를 반환하는 compile() 함수 포함
  - Compile()에서 하는 일
    - "Compiling code: 파일_이름"을 화면에 출력(파일_이름은 SourceCode에 주어진 파일 이름)
    - SourceCode에 있는 파일 이름의 확장자를 원본에서(예: .c) 오브젝트 파일의 확장자(예: .obj)로 변환한 후에 ObjectCode 객체 생성
    - "Generating object code: 오브젝트_파일_이름"을 화면에 출력
    - ObjectCode 객체 반환

# 실습 과제 #08- 구현해야 하는 클래스

- **C 컴파일러 클래스(이름은 본인이 지정)**
  - Compiler 인터페이스 구현
    - 인자로 전달된 SourceCode가 a.c라는 파일 이름을 가지고 있다면 compile() 함수가 호출될 때 다음 내용이 화면에 출력됨
    - 화면 출력 내용

```
Compiling code: a.c
Generating object code: a.obj
```

- **C 링커 클래스(이름은 본인이 지정)**
  - Linker 인터페이스 구현(Linker.java의 주석 참조해서 구현)

# 실습 과제 #08 – 구현해야 하는 클래스

- **Builder 인터페이스**
  - Builder.java의 주석문을 참고해서 인터페이스에 들어가야 하는 세 개 함수의 헤더를 선언
- **IDE 클래스**
  - Builder 인터페이스를 구현
- **Main 클래스 일부**
  - main() 함수 시작 부분에 Preprocessor, Compiler, Linker를 구현한 클래스 객체들을 생성하는 코드 추가
  - main() 함수 마지막 부분에 IDE를 사용해서 두 개 exe 파일을 생성하고 실행하는 코드 추가