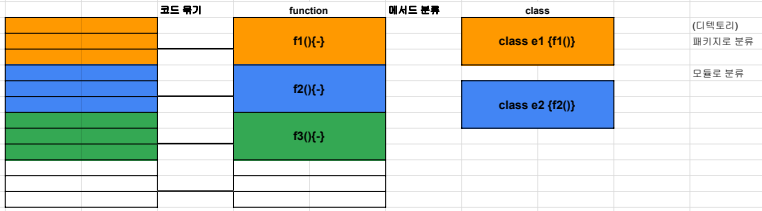
*코드를 유지보수하기 쉽게 만드는 방법



(디렉토리)
패키지로 분류

모듈로 분류

*메서드 : 특정 작업을 수행하는 코드를 묶어 놓은 것

```
void processBoardList() { ----}
```

listBoard	메서드명 : 메서드가 수행하는 작업을 잘 드러내는 이름으로 만든다. // 작업은 행위이기 때문에 [동사구] 형태로 만든다. // (가끔 명사구나 전치사구 형태일 경우도 있다.)
()	파라미터 목록 : 작업 수행에 필요한 데이터를 받을 변수 (parameter)
{---}	메서드 몸체(body) : 작업을 수행하는 코드
void	리턴 타입 : 작업 완료 후 결과값의 리턴 여부 // void는 리턴 안함. // 그밖에 리턴하는 값의 타입

*메서드에 묶인 코드 실행하기 : 메서드 호출(call)

변수 =	메서드명(값, 값, ----) ;	
↑	↑ ↑	메서드 파라미터에 전달할 값 : 아규먼트 (argument)
메서드가 값을	실행할 메서드	
리턴할 경우		
그 값을 변수		
(값은 받지 않아도 된다.)		

*파라미터와 아규먼트

int plus(int a, int b)		a, b = parameter
return a+b;	-> 메서드 정리 (define)	아규먼트 값을 받는 변수
}		100,200 = argument
int result = plus (100, 200) ;	-> 메서드 호출(call)	파라미터에 넘겨주는 값.

*메서드 호출

다중 쓰레드 : 멀티 쓰레드 = 병행쓰레드

실(Thread)	"main Thread"		
	main() {	m1() {	x() {
	-	-	-
	-	-	-
	-	m2() {	}
	m1() ;	_	y() {
JVM 종료	-	x();	-
	-	-	-
	m2() ;	-	2();
	-	}	-
	-		-
	m3() ;	m3() {	}
	-	-	
	-	-	
	}	y() ;	

*메서드와 변수의 종류

```
main() {
int i;      i= 로컬변수
m1();
m2();
}

int i;
static 변수
로컬변수로 밖으로 빼냄
```

```
m1(0 {
i = 100;
}
m2() {
i += 200;
}
```

*로컬변수(Local variables)
>메서드 블록 안에 선언된 변수
-메서드 실행할 때 생성
-메서드 완료할때 삭제
-메서드 안에서만 사용 가능

*메서드 분류 = 클래스 (class)

클래스명 : 명사 / 명사구 / 형용사

class App
main() {-}
welcome() {-}
displaymenu() {-}
promptmenu() {-}
displayline() {-}
prossesboardlist() {-}
prossesboarddetail() {-}
prossesboardinput() {-}
displayblankline() {-}

-->

class App
main() {-}
welcome() {-}
promptMenu() {-}

class board
processList(){-}
processDetail(){-}
processInput(){-}

class prompt
inputInt() {-}
inputString() {-}

*클래스 관계도

App → 사용 **BoardHadnler**
client **dependency**
↓ 사용 **의존객체**
Prompt ←
dependency

쌍방 참조가 일어나면 안됨
-유지 보수에 용이함
class -> Responsibility = Role
-각 클래스별로 책임이 있음.

***메서드 분류 = 클래스 (class)**

```
javac -d bin/main -encoding UTF-8 -sourcepath src/main/java src\main\java\com\bitcamp\board\App.java
```

***컴파일 옵션 : -sourcepath 소스폴더**

```
javac -encoding UTF-8   -d bin/main  -sourcepath src/main/java  src/main/java/bitcamp/board/App.java
```

javac	컴파일	
-encoding UTF-8		소스파일의 인코딩 문자집합 : 컴파일러에게 소스파일이 어떤 규칙에 따라 저장되었는지 알려주는 옵션
-d bin/main		컴파일 결과로 생성되는 .class 파일을 둘 폴더
-sourcepath src/main/java		소스파일에서 참조하는 클래스가 있는 파일의 경로를 알려준다.
src/main/java/bitcamp/board/App.java		소스파일

이클립스
이클립스 IDE 설정
eomcs-doc/devetol/개발도구준비.md.
naver d2

Eclipse IDE와 프로젝트

board-app /		
app/ <- 프로젝트폴더 <-(import) Eclipse IDE		
	eclipse 설정파일이 프로젝트 폴더에 있어야 한다.	
	그래야 이클립스가 프로젝트 폴더로 인식할 수 있다.	
	.project - 프로젝트 정보를 담은 파일	
Eclipse	.classpath - 소스 디렉토리 경로와 의존 라이브러리 정보를 담은 파일	
설정파일	.settings/* <- Eclipse IDE에서 자바관련 도구의 설정 정보를 담은 폴더	
	$gradle eclipse	< 설정 파일 생성
	$gradle cleanEclipse	