```
class A {
int a;          non-static 필드(변수) = 인스턴스 ==> Heap 영역에 만들어 진다.     Heap => gabage collector가 관리하는 영역
static int b;   static 필드(변수) = 클래스 ==> method Area
}
```

*인스턴스(non-static) 필드

```
                class A{                    로컬변수    인스턴스 생성 명령
변수를 만들라는 명령어   int v1;              A obj1 =    new A();
                boolean v2;                 A obj2 =    new A();
                }                           A obj3 =    new A();

                                            obj = local 변수
                                            메서드 호출이 끝나면 사라짐

                void m1() {                 new A(); = 인스턴스
                }
                {------------}
                obj1.m1() ;     call 했을때
                obj1.v1(100);
```

JVM stack

main()	
obj1	200
obj2	300
obj3	400

레퍼런스(인스턴스 주소를 담는 것 = 레퍼런스)

Heap

200	
v1	v2

A의 인스턴스

*클래스(static) 필드(변수)

```
class A {                   main{
static int v1;      →       A obj;
static boolean v2;          A.v1 = 100;
                            A.v2 = tree;


                            }
```

method arad	
A class	v1
	v2
JVM stack	Heap

1. Method Arad에 클래스가 로딩될때 스태틱 변수가 생성.

*클래스 로딩과 실행

```
$java Hello
1. Hello.class 를 찾는다.
2. Bytecode 검증
3. Method Area에 로딩한다.      →       .class      →
4. 스태틱 필드 생성                                 →
5. 스태틱 블록 실행                                 →
6. main() 호출      JVM으로 직접 실행할 경우만 실행  →
```

method arad	
메서드코드	관리하기 쉽게
변수코드	정리해서
상속코드	메모리에
- - -	배치

*클래스 로딩, 스태틱 필드 및 인스턴스 필드 생성

```
class Exam0140

class A {                   →       Exam0140.class
static int v1 ;                     ~~A.class~~
int v2 ;
}                           →       xam$0140.Aclass
main() { ~ }
    5번 A.v1 = 100;
7번     A p = new A() ;
10번    p.v2 = 200;
12번    A p2 = new A ();
15번    p2.v2 = 300;
```

method arad	
Exam0140.class >1/2번	
A class > 4번	v1 = 100 >>5/6번
	v2
JVM stack	Heap
main()	V2 = 200> 12번
p = 111 > 9번	V2 = 300 > 13번
p2 = 222 > 14번	

1. Exam0140.class
2. main() 호출
3.main()의 로컬변수 생성
4. A.class 를 메모리에 로딩
5. v1 스태틱 변수 생성
6. A.v1 = 100
7. new A();
8.인스턴스 생성
9.P = 200

Method Area	JVM stack	Heap
1.Exam0140.class 실행		
2. main() 메서드를 호출		
	3. main()의 로컬 변수 생성	
4.A.class		
5.v1 스태틱 변수 생성		
6. A.v1 = 100; 실행		
7. A p = new A();		8. v2 인스턴스 생성
	9. p > 주소	
10. p.v2 = 200;		11. v2 = 200;
12. A p2 = new A();		
		13 . v2 인스턴스 생성
	14. p > 주소	
15. p2.v2 = 300;		16. v2 = 300;

1	Exam0140.class 실행	3	main() 로컬 변수 생성	8	v2 인스턴스 생성

2	main() 메서드를 호출	9	p	11	v2 = 200 ;
4	A class	14	p2	13	v2 인스턴스 생성
5	v1 스태틱 변수 생성			16	v2 = 300 ;
6	A.v1 = 100; 실행				
7	A p = new A();				
10	p.v2 = 200				
12	A p2 = new A();				
15	p2.v2 = 300;				

***인스턴스 변수와 인스턴스 메서드가 놓이는 위치**

Method Area		JVM stack			Heap		
Exam0230.class		main()			new Calculater()		
main() { ~ }		args	c1	c2			
					result	~~plus() {~}~~	~~minus(){~}~~
Calculator.class							
result							
plus() { ~ }							
minus() { ~ }							

인스턴스를 생성할 때
메서드가 생성되는 것이 아니다.
메서드는 Method Area에 있다.

static변수는 이름으로 호출 가능.
static void m1()
{system.out.println("m1()")}

instence 변수는 주소가 있어야 호출할 수 있음.

void m2()

A obj1 = new A();
obj1.m2();

***인스턴스와 생성자**

2. 인스턴스 생성 후 그 주소를 가지고 기본 생성자를 호출한다.

Score	s	new Score	( ) ;

3.번지수가 들어감. **1. Score 설계도에 따라 인스턴스 변수를 Heap에 저장**

인스턴스 변수 생성과 생성자 호출이 끝나면, 인스턴스 주소를 리턴한다.

Score s1 = new Score

s1	200

200	name	kor	eng	math	sum	aver

Score s1 = new Score("홍길동", 100, 100, - - -) ;

s1	200

200	name	kor	eng	math	sum	aver
	홍길동	100	100	90	290	96. - - -

Score s2 = new Score("유관순", 100, 100, - - -) ;

s1	300

300	name	kor	eng	math	sum	aver
	유관순	100	100	90	290	96. - - -

인스턴스를 사용하기 전에 유효한 값으로 초기화시켜주는 것.

1	Exam0691.java			
2	main() 호출		사용하는 시점에 클래스 로딩.	
3	A.class			
4	A.class	a		
5	A.class	a = 7		
6	B.class			
7	B.class	b		
8	B.class	b = 22		
8	B.static { - } 출력			
9	B.static { - }	b = 22 + 7 = 29	B.class 로딩 끝	
10	A.class	a = 7 + b = 29 = 36		

```
class A {                   {
A() {  [복사]               인스턴스 초기화 블록 => 컴파일러가 모든 생성자의 앞 부분에 복사한다.
}                           }                    >> 여러 생성자에 공통으로 들어갈 코드가 있다면 인스턴스 초기화 블록에 담아두라.
A(m+a) { [복사]
}
A(String s) { [복사]
}
```

스태틱 초기화 블럭은 합쳐지고
인스턴스 초기화 블럭은 순서가 생성자 앞으로 이동한다.

변수 초기화 문장 = 스태틱 블럭이 합쳐지고, 인스턴스 블럭은 생성자와 합쳐진다.

*기능을 확장하는 방법 : 1. 기존클래스를 변경

Car
model
maker
capacity
car()
car(~생성자)
run()
sunroof()
auto()
Car(선루프 생성자 추가)
>기존 코드에 새 기능을 위한 코드 추가<

>기존 소스를 변경하면 기존의 Car를 사용해서 만든 프로그램에도 영향
>심각한 오류가 발생할 수 있다.
>쓰지 않는 코드가 누적되는 문제가 생긴다.