9/15 금요일 토론

정훈
2장 - stack memory, heap memory에 저장되는 reference primitive type, String pool
3장 - header block, body block, Method signature 정의
Object, instance class valuable과 instance valuable
polymorphism
inheritance composition

철환
java 변수 scope, lifetime
메소드를 실행했을때 메모리에서 일어나는 변화
Pass by Value

아영
Object, Instance, String pool, Abstraction, this(...) 에서 recursive invocation error 뜨는 이유

윤호
String pool, Polymorphism, Abstraction

Object vs Instance
비유하자면
class = 풀빵틀
object = 풀빵(개념)
instance = 풀빵틀에서 찍혀나온 각각의 풀빵 (객체가 소프트웨어에 실체화돼서 메모리에 올라온것)

activation Record
heap에 new로 객체를 생성했을 때 Heap 메모리에 쌓이는 구조가 Stack과 같이 쌓이는지

Method signature
메서드 명 파라미터 타입,개수

```
Public static resultType methodName(formalParameters). <— header
{.                                  <—— body
…..
}

methodName(formalParameters) ←- Method Signature
```

String Constatnt pool , String의 불변성의 중요성

String은 literal 방식과 new String(); 방식으로 생성이 가능한데

literal 방식으로 생성하면 intern() 메소드가 실행되고 String Constant Pool에 있으면 주소 값을, 없으면 저장하고 그 주소 값을 리턴한다.
new 연산자로 생성하면 Heap 영역에 생성된다.

불변인 이유 = 보안, 캐싱, 빠른 재사용, 동기화(Thread-safe)

Abstraction
내부 구현을 숨기고 간추리는것.

recursive constructor invocation error -> method signature가 같은 constructor를 함수 안에서 생성하면서 생기는 오류

변수	선언위치	생성시기(메모리 할당)	lifetime
클래스 변수(static)	클래스 영역	클래스가 메모리에 올라갈 때	프로그램 종료 까지
인스턴스 변수	클래스 영역	인스턴스가 생성될 때	클래스의 객체가 메모리에 남아있을때 까지
지역 변수	클래스 이외의 영역(메서드, 생성자, 초기화 블럭)	변수 선언문이 수행되었을 때	선언된 블록 내부를 프로그램이 실행되는 동안

inheritance vs composition

Super class의 field 값, 구현이 노출되는 상속이라면 캡슐화를 깨뜨릴 수 있다. 결합도가 높아져 의존하게 되고 변화에 유연하게 대처하기 힘들어짐.

1. 확장을 고려하고 설계한 확실한 is - a 관계일 때
2. API에 아무런 결함이 없는 경우, 결함이 있다면 하위 클래스까지 전파돼도 괜찮은 경우
3. 불필요하게 super class의 내부 구현을 노출하지 않는지

확인하고 이런 경우에는 상속을 적절하게 사용하고 composition도 적절히 사용하면 좋은 코드를 만들 수 있다.

Inheritance Is-a realationship
Composition has-a relationship

Polymorphism
동적바인딩, Liskov 치환 원칙

Pass by Value
자바는 모든 변수를 Pass by Value 함. Pass by Reference와 헷갈리지 않게 주의