1. 클래스와 객체의 차이는 무엇인가?

```
Public class A{
    Void method1(){
        B b = new B();
        C c = new C();
        D d = new D();
    }
}
```

클래스

인스턴스

인스턴스

인스턴스

객체

```
Public Class B{ }

Public Class C{ }

Public Class D{ }
```

클래스

2. 객체와 인스턴스의 차이는 무엇인가?
3. Call by value 와 Call by reference 의 차이

```
Void method1(int param){
    param = 3;
}
Void method2(Hello param){
    param.field = 3;
}
Int number = 1;
Hello hello = new Hello();
hello.field = 1;
method1(number);
method2(hello);
```

System.out.println(number);

System.out.println(hello.field);

method1(hello.field);

System.out.println(hello.field);

4. Primitive type 과 Reference type 의 차이

Primitive type : 기본 타입 – int, short, long, char, … (리터럴)

Reference type : 배열, 객체

5. 오버로딩과 오버라이딩?

```
Class Parent{
  int a;
  void move(int c){
    this.a = c;
  }
}

Class Child extends Parent{
  int b;
  String k;
  @override
  void move(int c){
    this.b = c;
  }
  int move(String c){
```

```
        this.k = c;
      }
    }
```

6. 추상 클래스와 인터페이스의 차이

   * 추상 클래스

   - 한 개 이상의 추상메서드를 갖는 클래스 -> 일반 메소드 가능

    > 추상 메소드 : 내용이 없는 메소드

   - 상속받은 자식 클래스에서 내용을 오버라이드해서 사용.

   - new 키워드로 객체 생성 불가능

     public abstract class Test { }

     Test test = new Test(); -> x

     public class Hello extends Test{ }

     Test test = new Hello();

   * 인터페이스

    - 추상 메소드와 상수로만 이루어짐.

    - 다중 사용이 가능

7. StringBuilder 와 StringBuffer 의 차이

   - StringBuilder : 동기화 지원 X

   - StringBuffer : 동기화 지원 O

   예) 멀티스레드 환경에서 A 스레드와 B 스레드 모두 같은 StringBuffer 클래스 객체 sb 의 append() 메서드를 사용한다면

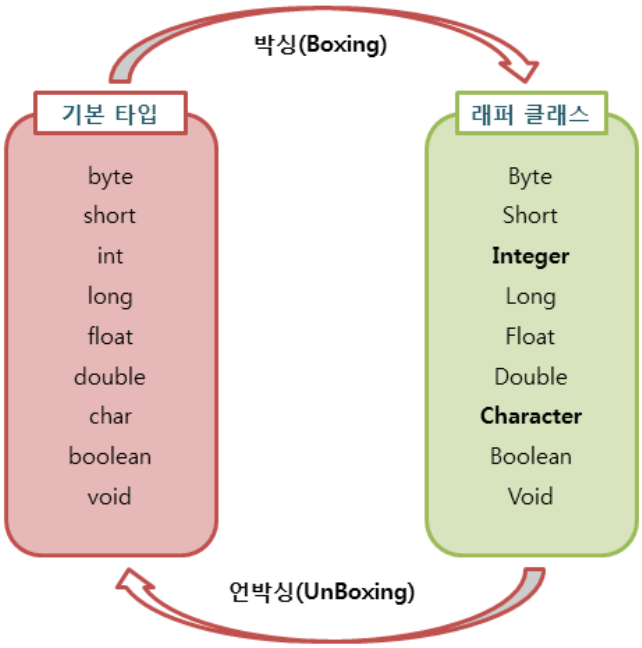
   > A 스레드 : sb 의 append() 동기화 블록에 접근 및 실행

> B 스레드 : A 스레드 sb 의 append() 동기화 블록에 들어가지 못하고 block 상태가 됨.

> A 스레드 : sb 의 append()동기화 블록에서 탈출

> B 스레드 : block 에서 running 상태가 되며 sb 의 append() 동기화 블록에 접근 및 실행.

8. 오토 박싱 & 오토 언박싱의 차이



객체 비교는 eqals

9. String a = “apple”, String b = new String(”apple”) 차이

String 을 리터럴 값으로 할당하는 경우엔 Heap 메모리 영역안의 특별한 메모리 공간인 String constant pool 에 저장된다.