**프로그램의 단위**

1. **모듈(class~ package)**
2. **라이브러리(package~)**
   1. **끌어와서 사용할 수 있는**
3. **API Application Programming interface**
   1. **라이브러리가 엄청 많이 모이면**
4. **프레임워크**
   1. **라이브러리나 API로 구성되어 있는데 프레임워크의 규칙에 맞춰서 개발을 진행해야함**

**자바의 데이터 타입**

1. **자료형**
   1. **class type**
      1. **선언** 
         1. **class 의 전역 변수를 정의 =>필드**
      2. **생성**
         1. **new 연산자로 생성자를 호출**
         2. **인스턴스 객체(자료형 데이터)가 반환**
         3. **인스턴스 : 동적메모리의 GC가 사용하지 않을 때 삭제해서**
      3. **초기화**
         1. **생성자를 매개변수를 정의 this로 필드의 값을 대입**
         2. **this == 객체 자신**
   2. **배열**
      1. **생성**
         1. **타입 [] 변수명=new 타입 [길이];**
         2. **타입 [] 변수명={data,data ....};**
      2. **구성**
         1. **길이와 인덱스(순서)**
      3. **특징**
         1. **자료의 참조를 직접하는 것이 아니라 순서대로 있는 자료들을 찾아서 반환하는 형식**
         2. **자료를 순서대로 배치하기 때문에 메모리를 효율적으로 사용**
         3. **바로 참조하는 것이 아니라 찾아서 반환하는 것이기 때문에 작을때는 비효율 적이지만 자료가 많아질 수로 메모리가 효율적이다.**
2. **기본형**
   1. **원시형 (이유)**
      1. **메모리에 수를 저장 (컴퓨터 입장에서 수는 원시 데이터)**
      2. **메모리를 직접 참조 (1개의 데이터만 있다.)**
      3. **그 프로그래밍 언어에서 원시형으로 사용하는 타입 => 기본형**
   2. **정수형**
      1. **byte**
         1. **1byte 정수 데이터**
         2. **-128~127**
         3. **java.lang.Byte 랩퍼클래스**
      2. **short**
         1. **2byte**
         2. **-2^15~2^15-1**
         3. **java.lang.Short 랩퍼클래스**
      3. **int**
         1. **4byte**
         2. **-2^31~2^31-1**
         3. **java.lang.Integer 랩퍼클래스**
         4. **자바에서 정수를 사용하면 무조건 int 가 된다.**
      4. **long**
         1. **8byte**
         2. **-2^63~2^63-1**
         3. **java.lang.Long 랩퍼클래스**
         4. **자바에서 정수를 사용하면 무조건 int 기 int 보다 큰 정수인 long 을 생성하려면 l,L을 정수 끝에 사용한다.**
   3. **실수형**
      1. **메모리 영역**
         1. **부호부**
            1. **수의 +-**
         2. **지수부**
            1. **10^n 수를 부동소수점으로 변경해서 그 수가 10의 몇 제곱인지 표현하는 부분**
         3. **가수부**
            1. **부동소수점으로 변경했을때 수들을 저장하는 부분 (만약 표현하기에 더 큰 수가 오면 나머지 모두 0으로 만든다.)**
      2. **특징**
         1. **지수부 덕에 아주아주 크거나 작은 수를 표현**
         2. **실수 연산이 부정확할 수 있다.** 
            1. **1.2-1.1=0.9999999998**
         3. **가수부가 작아서 정확도가 떨어진다.**
      3. **float**
         1. **4byte**
         2. **1bit 부호부**
         3. **8bit 지수부**
         4. **23bit 가수부(int 보다 작다)**
         5. **자바에서 실수를 작성하면 무조건 double 이기 때문에 float로 선언된 변수가 실수를 바로 참조할 수 없다. 때문에 캐스팅 형변환을 하거나 f,F를 작성할 수 있다.**
      4. **double**
         1. **8byte**
         2. **1bit 부호부**
         3. **11bit 지수부**
         4. **52bit 가수부**
         5. **자바에서 실수를 작성하면 double 형이 된다.**

* 1. **논리형**
     1. **boolean**
     2. **논리연산을 했을 때 결과**
     3. **1bit 데이터지만 크기는 1byte**
     4. **1 true : 참**
     5. **0 false : 거짓**
     6. **==,!=,>=,<=,>,<,!**
  2. **문자형**
     1. **char**
     2. **유니코드 문자표의 번호**
     3. **유니코드 문자표가 유동 크기 때문에 1~4byte 까지의 문자 크기를 갖는다.**
     4. **'a' or 97, 'A' or 65**
     5. **oxAC00 (유니코드표를 16 참조)**
     6. **'u\0042' (유니코드표를 참조)**
     7. **0123456789 (10진수)**
     8. **0123456789ABCDEF (16진수)**

1. **리터럴 표기법**
   1. **값이 보이는 대로 작성된 것** 
      1. **14 -> 수 (리터럴)**
      2. **new Student() -> 알수없음**
   2. **기본형 무조건 리터럴 표기 (규칙)**
   3. **자료형인데 리터럴하게 표기하는 것**
      1. **String s="안녕하세요";**
         1. **char [] arr= new char[5];**
         2. **arr[0]='안';**
         3. **arr[1]='녕';**
         4. **arr[2]='하';**
         5. **arr[3]='세';**
         6. **arr[4]='요';**
         7. **=>java 가 문자열을 물리적으로 저장하는 실제 방법**
      2. **char [] arr={'안','녕','하','세','요'};**