객체 지향 프로그래밍

- 1. 클래스와 객체

이클립스 단축키

- **프로젝트 및 클래스 생성**
  - Ctrl + n + java project : 새 프로젝트 생성
  - Ctrl + n + java class : 새 클래스 생성
- **프로그램 실행**
  - Ctrl + f11 : 프로그램 실행
- **코드 자동 정리**
  - Ctrl + Shift + O : 자동으로 import 하기, import 코드 정리
  - Ctrl + Shift + F : 코드 자동 정리; 코드 내용을 문법 템플릿에 맞게 포맷팅(들여쓰기) 해준다.
- **자동 완성**
  - Ctrl + Space : 자동 완성 기능
  - ma+Ctrl+Space+Enter: public static void main(String[] args) {}
  - sysout + Ctrl + Space : System.out.println();
- **코드 이동 및 수정**
  - Ctrl + D : 한줄 삭제
  - Ctrl + L : 특정 소스라인으로 이동
  - Alt + 방향키(위, 아래) : 특정 라인 한 줄 이동
  - Ctrl + Alt + 방향키(위, 아래) : 특정 라인 코드 복사 (위 누르면 위로 복사 아래 누르면 아래로 복사)
- **주석 달기**
  - Ctrl + / : 한줄 주석 처리 또는 제거
  - Ctrl + Shift + / : 블록 주석(/* */)
  - Ctrl + Shift + ₩ : 블록 주석 제거
- **검색**
  - Ctrl + H : 프로젝트 전체에서 특정 문구(텍스트) 포함 File 검색

실 습

강제 타입 변환

- **자동 타입 변환이 안 되는 경우 : 큰 타입이 작은타입으로 변환할 때**

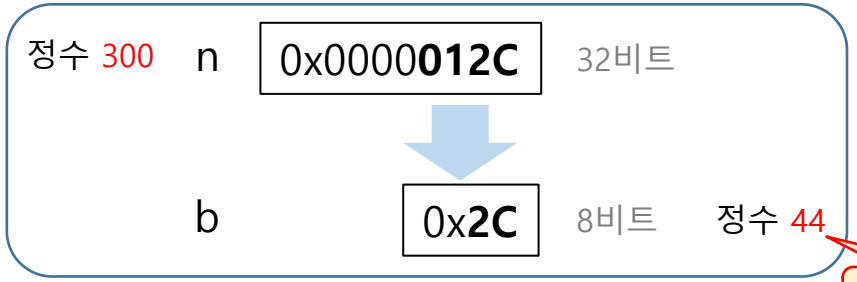
오류

```
int n = 300;
byte b = n; // 컴파일 오류. int 타입이 byte로 자동 변환 안 됨
```

강제 타입 변환하려면, byte b = (**byte**)n; 로 수정

- **강제 타입 변환**
  - 개발자가 필요하여 강제로 타입 변환을 지시
    - () 안에 변환할 타입 지정
  - 강제 변환은 값 손실 우려

byte b = (**byte**)**n**; 에 따른 손실



44 = 300%256

```
double d = 1.9;
int n = (int)d;  // n = 1
```

강제 타입 변환으로
소숫점 이하 0.9 손실

타입 변환

자동 타입 변환과 강제 타입 변환의 이해를 위한 예제이다.
다음 소스의 실행 결과는 무엇인가?

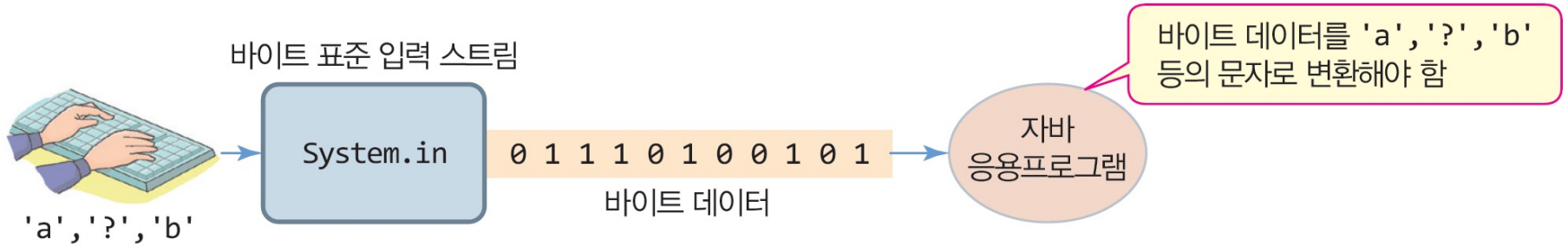
```
public class TypeConversion {
   public static void main(String[] args) {
      byte b = 127;
      int i = 100;

      System.out.println(b+i);
      System.out.println(10/4);
      System.out.println(10.0/4);
      System.out.println((char)0x12340041);
      System.out.println((byte)(b+i));
      System.out.println((int)2.9 + 1.8);
      System.out.println((int)(2.9 + 1.8));
      System.out.println((int)2.9 + (int)1.8);
   }
}
```

강제 타입 변환 결과 0x41이 되며, 문자 A의 코드임

```
227
2
2.5
A
-29
3.8
4
3
```

자바에서 키 입력

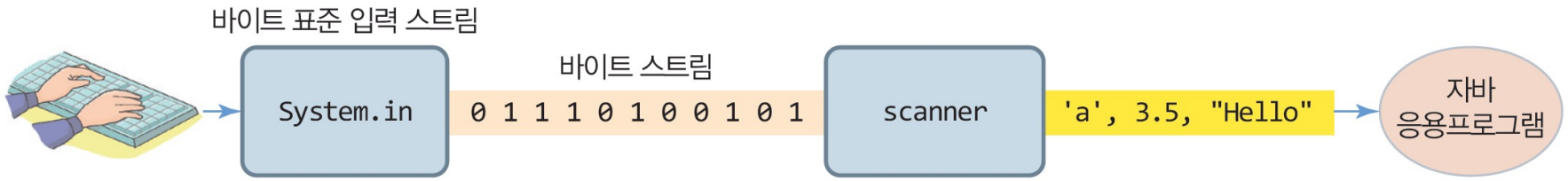
- **System.in**
  - 키보드로부터 직접 읽는 자바의 표준 입력 스트림
  - 키 값을 바이트(문자 아님)로 리턴
- **System.in을 사용할 때 문제점**
  - 키 값을 바이트 데이터로 넘겨주므로 응용프로그램이 문자 정보로 변환해야 함



Scanner로 쉽게 키 입력

- **Scanner 클래스**
  - System.in에게 키를 읽게 하고, 읽은 바이트를 문자, 정수, 실수, 불린, 문자열 등 다양한 타입으로 변환하여 리턴
    - java.util.Scanner 클래스
- **객체 생성**

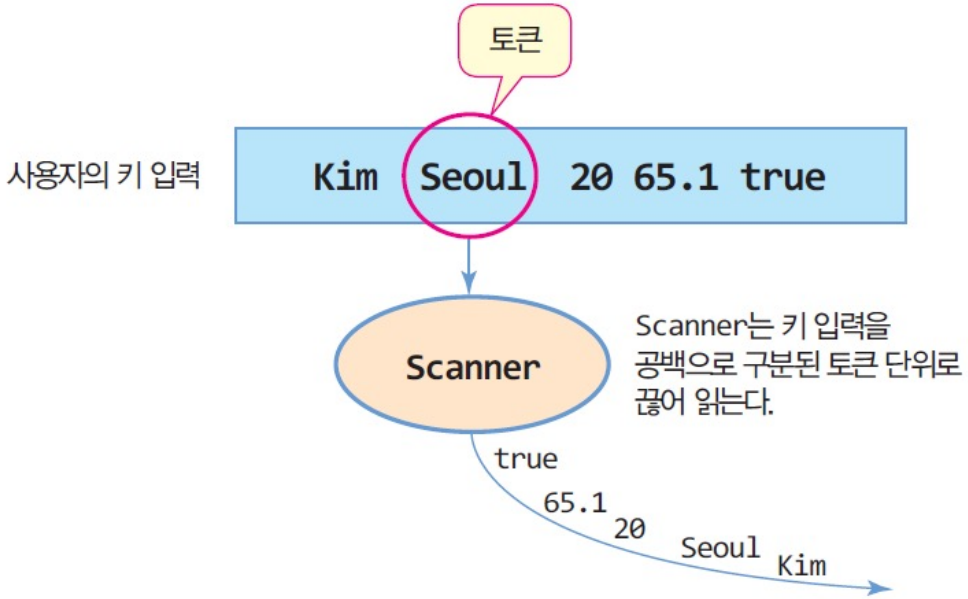
```
import java.util.Scanner; // import 문 필요
...
Scanner a = new Scanner(System.in); // Scanner 객체 생성
```



  - System.in에게 키를 읽게 하고, 원하는 타입으로 변환하여 리턴

Scanner를 이용한 키 입력

- Scanner에서 키 입력 받기
  - Scanner는 입력되는 키 값을 공백으로 구분되는 아이템 단위로 읽음
  - 공백 문자 : '₩t', '₩f', '₩r', ' ', '₩n'
- 개발자가 원하는 다양한 타입의 값으로 바꾸어 읽을 수 있음




```
Scanner scanner = new Scanner(System.in);

String name = scanner.next();          // "Kim"
String city = scanner.next();          // "Seoul"
int age = scanner.nextInt();           // 20
double weight = scanner.nextDouble(); // 65.1
boolean single = scanner.nextBoolean(); // true
```

Scanner 주요 메소드

메소드	설명
String next()	다음 토큰을 문자열로 리턴
byte nextByte()	다음 토큰을 byte 타입으로 리턴
short nextShort()	다음 토큰을 short 타입으로 리턴
int nextInt()	다음 토큰을 int 타입으로 리턴
long nextLong()	다음 토큰을 long 타입으로 리턴
float nextFloat()	다음 토큰을 float 타입으로 리턴
double nextDouble()	다음 토큰을 double 타입으로 리턴
boolean nextBoolean()	다음 토큰을 boolean 타입으로 리턴
String nextLine()	'\n'을 포함하는 한 라인을 읽고 '\n'을 버린 나머지 문자열 리턴
void close()	Scanner의 사용 종료
boolean hasNext()	현재 입력된 토큰이 있으면 true, 아니면 입력 때까지 무한정 대기, 새로운 입력이 들어올 때 true 리턴. crtl-z 키가 입력되면 입력 끝이므로 false 리턴

예제 : Scanner를 이용한 키 입력 연습

Scanner를 이용하여
이름, 도시, 나이, 체중,
독신 여부를 입력 받고
다시 출력하는
프로그램을 작성하라.

```
import java.util.Scanner;

public class ScannerEx {
    public static void main(String args[]) {
        System.out.println("이름, 도시, 나이, 체중, 독신 여부를 빈칸으로 분리하여 입력하세요");
        Scanner scanner = new Scanner(System.in);

        String name = scanner.next(); // 문자열 읽기
        System.out.print("이름은 " + name + ", ");

        String city = scanner.next(); // 문자열 읽기
        System.out.print("도시는 " + city + ", ");

        int age = scanner.nextInt(); // 정수 읽기
        System.out.print("나이는 " + age + "살, ");

        double weight = scanner.nextDouble(); // 실수 읽기
        System.out.print("체중은 " + weight + "kg, ");

        boolean single = scanner.nextBoolean(); // 논리값 읽기
        System.out.println("독신 여부는 " + single + "입니다.");

        scanner.close(); // scanner 닫기
    }
}
```

```
이름, 도시, 나이, 체중, 독신 여부를 빈칸으로 분리하여 입력하세요.
Kim Seoul 20 65.1 true
이름은 Kim, 도시는 Seoul, 나이는 20살, 체중은 65.1kg, 독신 여부는 true입니다.
```

예제 : /와 % 산술 연산

초 단위의 정수를 입력받고, 몇 시간, 몇 분, 몇 초인지 출력하는 프로그램을 작성하라.

```
import java.util.Scanner;

public class ArithmeticOperator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("정수를 입력하세요: ");
        int time = scanner.nextInt();       // 정수 입력
        int second = time % 60;             // 60으로 나눈 나머지는 초
        int minute = (time / 60) % 60;      // 60으로 나눈 몫을 다시 60으로 나눈 나머지는 분
        int hour = (time / 60) / 60;        // 60으로 나눈 몫을 다시 60으로 나눈 몫은 시간

        System.out.print(time + "초는 ");
        System.out.print(hour + "시간, ");
        System.out.print(minute + "분, ");
        System.out.println(second + "초입니다.");

        scanner.close();
    }
}
```

```
정수를 입력하세요:5000
5000초는 1시간, 23분, 20초입니다.
```

예제 : 비교 연산자와 논리 연산자 사용하기

다음 소스의 실행 결과는 무엇인가?

```
public class LogicalOperator {
   public static void main (String[] args) {
      // 비교 연산
      System.out.println('a' > 'b');
      System.out.println(3 >= 2);
      System.out.println(-1 < 0);
      System.out.println(3.45 <= 2);
      System.out.println(3 == 2);
      System.out.println(3 != 2);
      System.out.println(!(3 != 2));

      // 비교 연산과 논리 연산 복합
      System.out.println((3 > 2) && (3 > 4));
      System.out.println((3 != 2) || (-1 > 0));
      System.out.println((3 != 2) ^ (-1 > 0));
   }
}
```

```
false
true
true
false
false
true
false
false
true
true
```

예제 : if문 사용하기 (Scanner) 사용

시험 점수가 80점이 이상이면 합격 판별을 하는 프로그램을 작성하시오.

```
import java.util.Scanner;

public class SuccessOrFail {
   public static void main (String[] args) {
      Scanner scanner = new Scanner(System.in);

      System.out.print("점수를 입력하시오: ");
      int score = scanner.nextInt();
      if (score >= 80)
         System.out.println("축하합니다! 합격입니다.");

      scanner.close();
   }
}
```

```
점수를 입력하시오: 95
축하합니다! 합격입니다.
```

예제 : if-else 사용하기

입력된 수가 3의 배수인지 판별하는 프로그램을 작성하시오.

```
import java.util.Scanner;

public class MultipleOfThree {
   public static void main (String[] args) {
      Scanner in = new Scanner(System.in);

      System.out.print("수를 입력하시오: ");
      int number = in.nextInt();

      if (number % 3 == 0)
         System.out.println("3의 배수입니다.");
      else
         System.out.println("3의 배수가 아닙니다.");

      scanner.close();
   }
}
```

```
수를 입력하시오: 129
3의 배수입니다.
```

예제 2-12 : 다중 if-else로 학점 매기기

다중 if-else문을 이용하여 입력받은 성적에 대해
학점을 부여하는 프로그램을 작성해보자.

```
import java.util.Scanner;
public class Grading {
    public static void main(String[] args) {
        char grade;
        Scanner scanner = new Scanner(System.in);

        System.out.print("점수를 입력하세요(0~100): ");
        int score = scanner.nextInt(); // 점수 읽기
        if(score >= 90) // score가 90 이상
            grade = 'A';
        else if(score >= 80) // score가 80 이상 90 미만
            grade = 'B';
        else if(score >= 70) // score가 70 이상 80 미만
            grade =  ' C ' ;
        else if(score >= 60) // score가 60 이상 70 미만
            grade = 'D';
        else // score가 60 이만
            grade = 'F';
        System.out.println("학점은 "+ grade + "입니다.");

        scanner.close();
    }
}
```

```
점수를 입력하세요(0~100): 89
학점은 B입니다.
```

예제 : 중첩 if-else 문 사례

점수와 학년을 입력 받아 60점 이상이면 합격,
미만이면 불합격을 출력한다. 4학년의 경우 70점 이상이어야 합격이다.

```
import java.util.Scanner;
public class NestedIf {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("점수를 입력하세요(0~100): ");
        int score = scanner.nextInt();

        System.out.print("학년을 입력하세요(1~4): ");
        int year = scanner.nextInt();

        if(score >= 60) { // 60점 이상
            if(year != 4)
                System.out.println("합격!"); // 4학년 아니면 합격
            else if(score >= 70)
                System.out.println("합격!"); // 4학년이 70점 이상이면 합격
        else
            System.out.println("불합격!"); // 4학년이 70점 미만이면 불합격
        }
        else // 60점 미만 불합격
            System.out.println("불합격!");

        scanner.close();
    }
}
```

```
점수를 입력하세요(0~100): 65
학년을 입력하세요(1~4): 4
불합격!
```

예제 : switch 문 활용

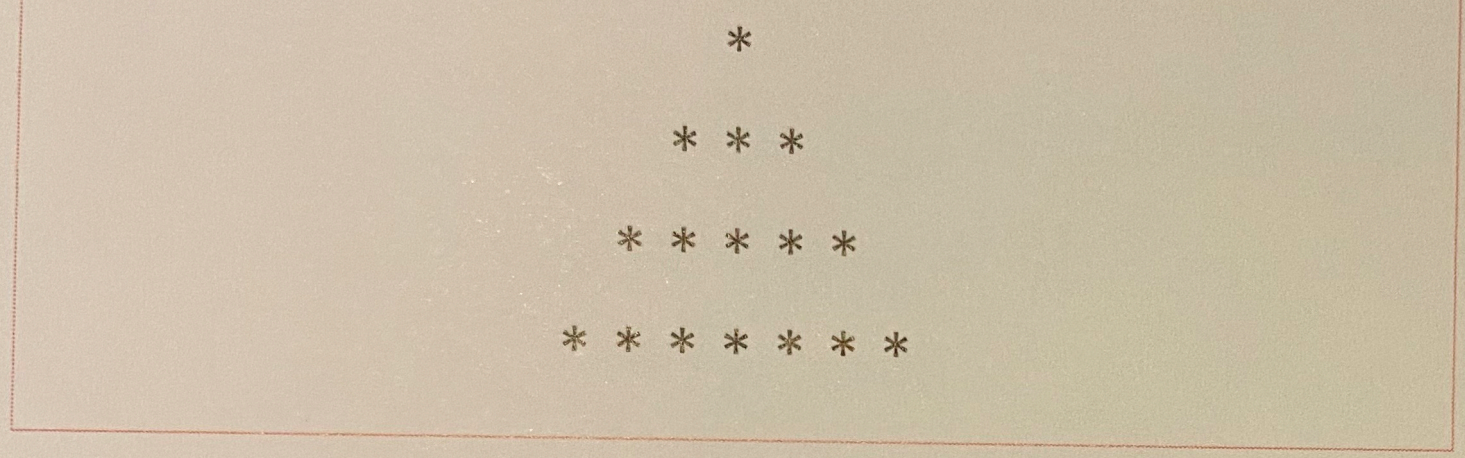
switch 문을 이용하여 커피 메뉴의 가격을 알려주는 프로그램을 작성하라.
에스프레소, 카푸치노, 카페라떼는 3500원이고, 아메리카노는 2000원이다.

```
import java.util.Scanner;
public class CoffeePrice {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("무슨 커피 드릴까요? ");
        String order = scanner.next();
        int price=0;
        switch (order) {
            case "에스프레소":
            case "카푸치노":
            case "카페라떼":
                price = 3500;
                break;
            case "아메리카노" :
                price = 2000;
                break;
            default:
                System.out.println("메뉴에 없습니다!");
        }
        if(price != 0)
            System.out.print(order + "는 " + price + "원입니다");
        scanner.close();
    }
}
```

```
무슨 커피 드릴까요? 에스프레소
에스프레소는 3500원입니다
```

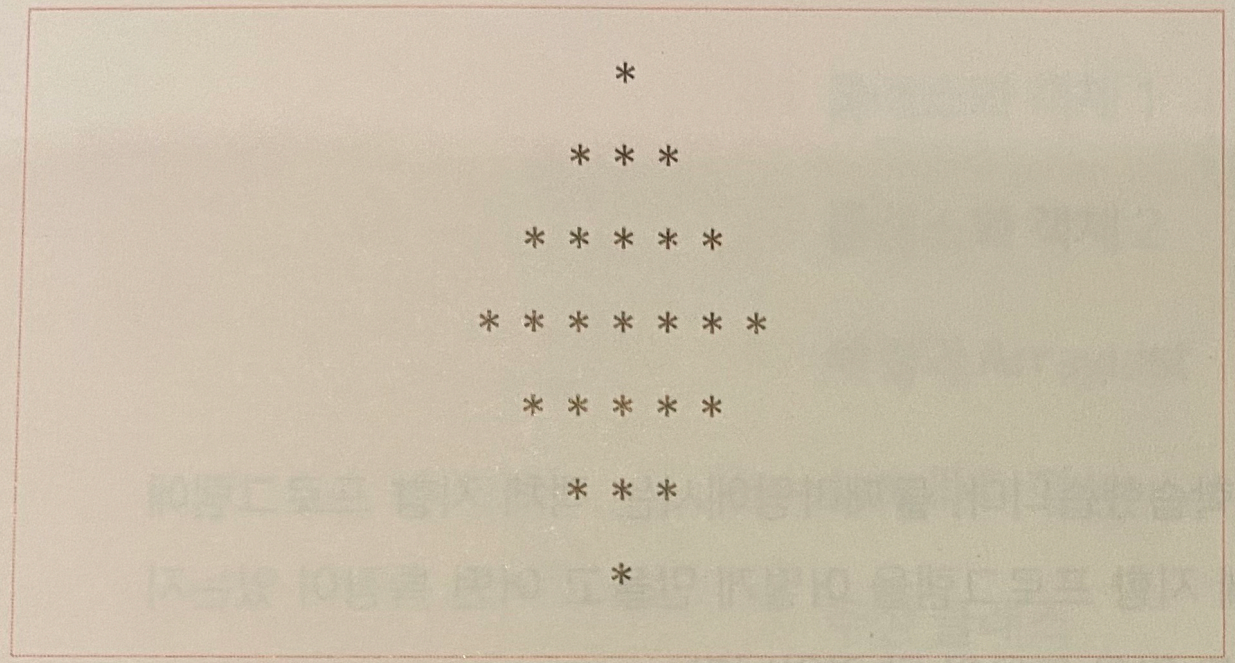
Q4

Q4 반복문을 사용하여 다음 모양을 출력하는 프로그램을 만들어 보세요.



Q5

Q5 반복문과 조건문을 사용하여 다음 모양을 출력하는 프로그램을 만들어 보세요.



객체 지향 프로그래밍

- 1. 클래스와 객체

객체 지향프로그래밍과 클래스

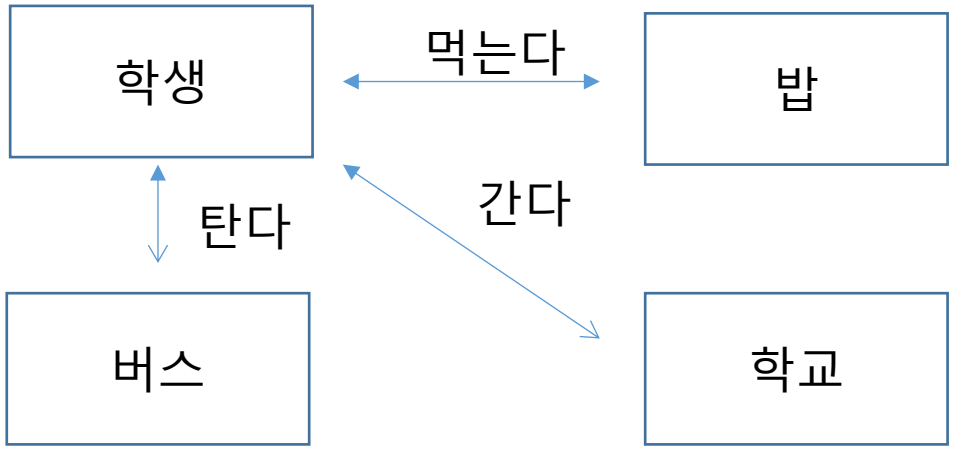
- **객체(Object)란 ?**
  - “의사나 행위가 미치는 대상” – 사전적 의미
  - 구체적, 추상적 데이터 단위

- **객체지향 프로그래밍(Object Oriented Programming, OOP)**
  - 객체를 기반으로 하는 프로그래밍
  - cf. 절차 프로그래밍 (Procedural Prigramming , 예) C 언어)
-

생활 속의 객체 예

- **학교 가는 과정에 대한 절차적 프로그래 밍**
  - -일어난다 -> 씻는다-> 밥을 먹는다 -> 버스를 탄다-> 요금을 지불한다 -> 학교에 도착한다.
  - 시간의 흐름에 따른 프로그래밍

- **학교 가는 과정에 대한 객체 지향 프로그래밍**
  - 객체를 정의
  - 객체의 기능 구현
  - 객체 사이의 협력 구현



클래스(class)

- **클래스란?**
  - 객체에 대한 속성과 기능을 코드로 구현 한 것
  - "클래스를 정의 한다" 라고 함
  - 객체에 대한 청사진(blueprint)

- **객체의 속성**
  - 객체의 특성, 속성, 멤버 변수
  - property, attribute, member variable

- **객체의 기능**
  - 객체가 하는 기능들을 메서드로 구현
  - method, member function

클래스 정의 하기

(접근 제어자) class 클래스 이름{

멤버 변수;

메서드;

}

- **학생 클래스의 예**
  - 속성 : 학번, 이름, 학년, 사는 곳 등등...
  - 기능 : 수강신청, 수업듣기, 시험 보기 등등...

클래스 정의 하기

- class는 대부분 대문자로 시작
- 하나의 java 파일에 하나의 클래스를 두는 것이 원칙이나, 여러 개의 클래스가 같이 있는 경우 public 클래스는 단 하나이며, public 클래스와 자바 파일의 이름은 동일해야 함

- 자바의 모든 코드는 class 내부에 위치

학생 클래스 만들기

```
package classpart;

public class Student {
    int studentID;          //학번
    String studentName;     //학생 이름
    int grade;              //학년
    String address;         //사는 곳
}
```

클래스를 만드는 예약어

클래스 이름

멤버 변수

속성	자료형	변수 이름	설명
학번	int	studentID	학번은 정수로 나타낼 수 있기 때문에 int형으로 선언합니다.
이름	String	studentName	학생 이름은 문자로 되어 있습니다. 그런데 이름은 A 같은 하나의 문자가 아니라 여러 개의 문자로 이루어진 문자열로 표현합니다. 문자열은 자바에서 제공하는 String 클래스를 사용합니다.
학년	int	grade	학년은 정수로 나타낼 수 있기 때문에 int형으로 선언합니다.
사는 곳	String	address	문자열을 나타내기 위해 String을 사용합니다.

클래스의 속성

- 클래스의 특징을 나타냄
- property, attribute 라고도 함
- 자료형을 이용하여 멤버 변수로 선언

속성	자료형	변수 이름	설명
학번	int	studentID	학번은 정수로 나타낼 수 있기 때문에 int형으로 선언합니다.
이름	String	studentName	학생 이름은 문자로 되어 있습니다. 그런데 이름은 A 같은 하나의 문자가 아니라 여러 개의 문자로 이루어진 문자열로 표현합니다. 문자열은 자바에서 제공하는 String 클래스를 사용합니다.
학년	int	grade	학년은 정수로 나타낼 수 있기 때문에 int형으로 선언합니다.
사는 곳	String	address	문자열을 나타내기 위해 String을 사용합니다.

클래스의 기능

- 메서드(method)로 구현
- 멤버 함수(member function) 라고도 함
- 객체가 수행하는 기능을 구현

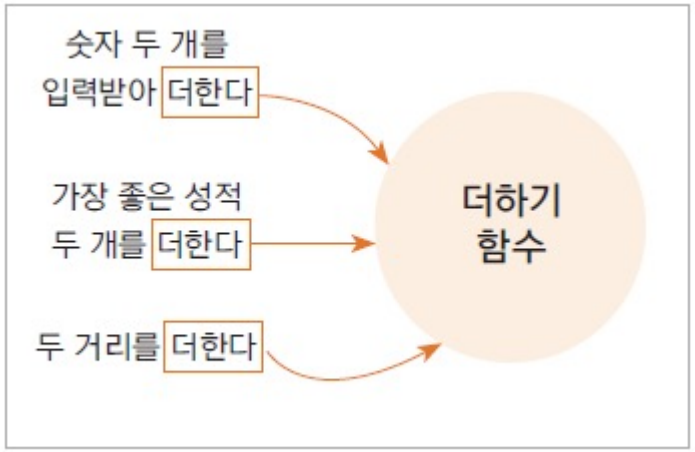
```
package classpart;

public class Student {
   int studentID;
   String studentName;
   int grade;
   String address;

   public void showStudentInfo( ) {
      System.out.println(studentName + "," + address);   //이름, 주소 출력
   }
}
```

메서드 추가

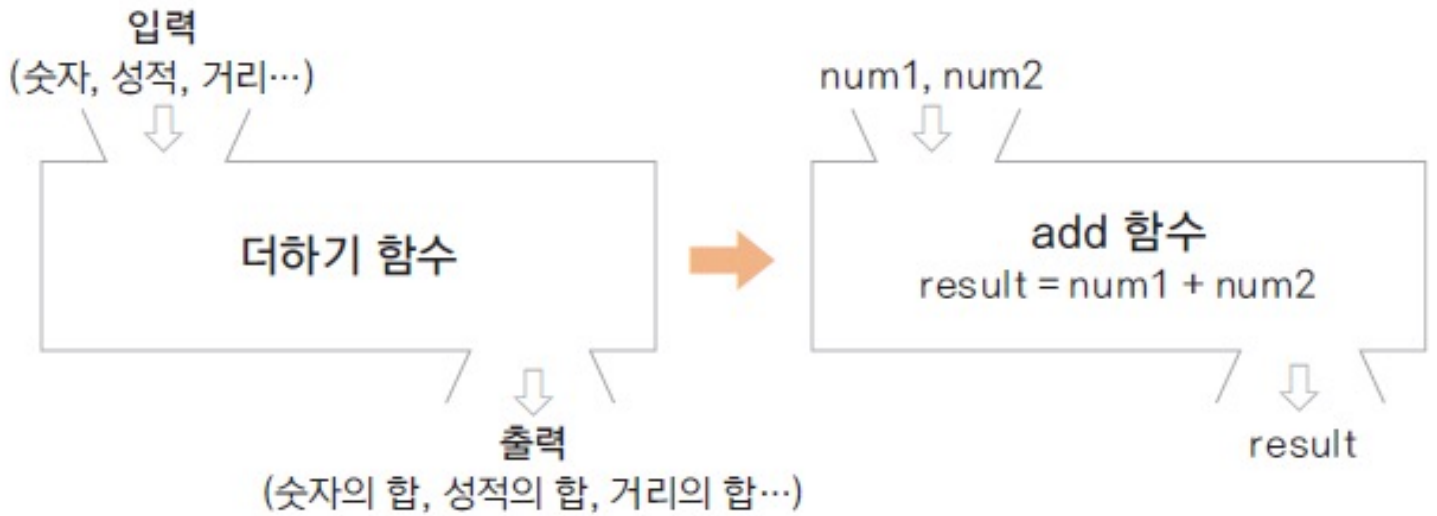
메서드

- **함수의 일종**
- **객체의 기능을 제공하기 위해 클래스 내부에 구현되는 함수**

- **함수란 ?**
  - 하나의 기능을 수행하는 일련의 코드
  - 중복되는 기능은 함수로 구현하여
  - 함수를 호출하여 사용함

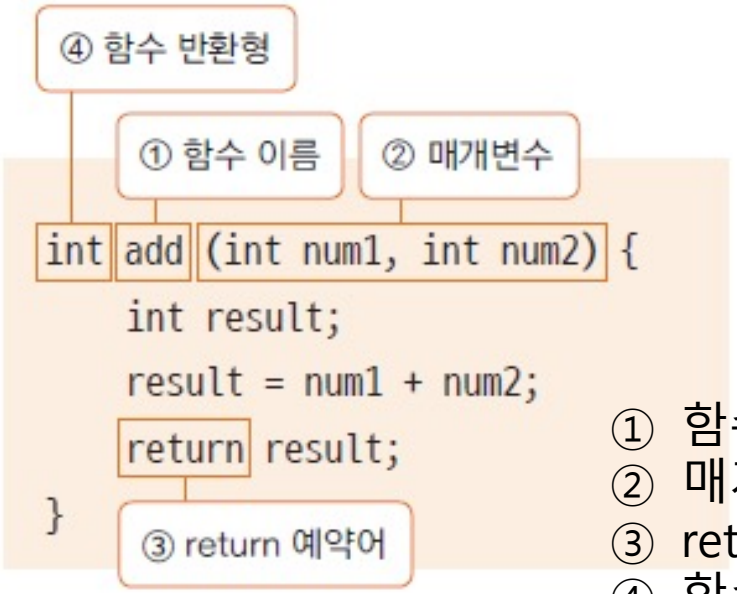


함수


입력
(숫자, 성적, 거리…)
더하기 함수
출력
(숫자의 합, 성적의 합, 거리의 합…)
num1, num2
add 함수
result = num1 + num2
result

함수 정의 하기

- 함수를 코드로 구현
- 함수의 이름, 매개변수, 반환 값을 선언하고 코드를 구현함



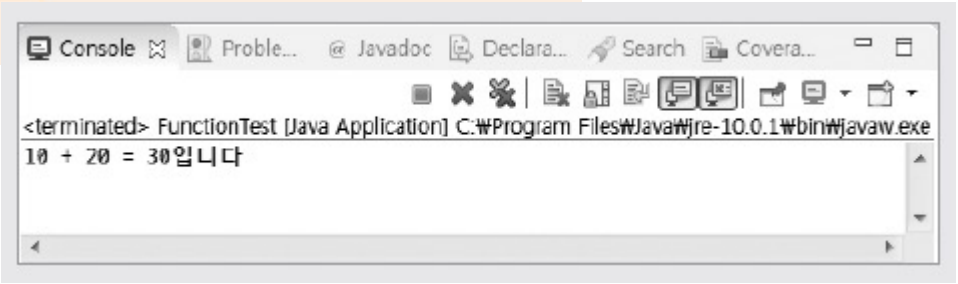
① 함수 이름: 함수의 기능과 관련하여 명명
② 매개 변수 : 함수의 수행을 위해 필요한 변수
③ return : 함수 수행 결과를 반환하기 위한 예약어
④ 함수 반환 형: 반환 값의 자료형을 나타냄
반환 값이 없는 경우 void라고 씀

함수 구현하고 호출하기

```
package classpart;

public class FunctionTest {
   public static void main(String[ ] args) {
      int num1 = 10;
      int num2 = 20;

      int sum = add(num1, num2);
      System.out.println(num1 + " + " + num2 + " = " + sum + "입니다");
   }

   public static int add(int n1, int n2) {
      int result = n1 + n2;
      return result;  //결과 값 반환
   }
}
```

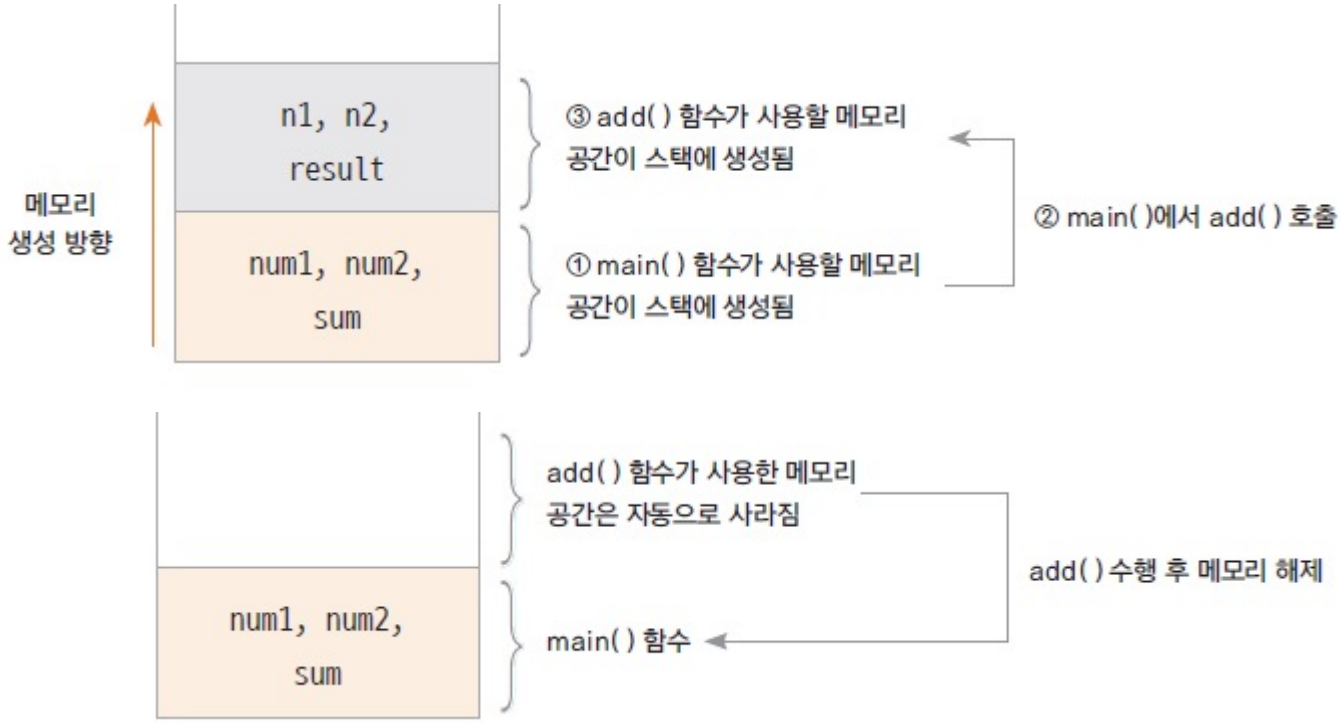
add( ) 함수 호출

add( ) 함수



함수와 스택 메모리

- 함수가 호출될 때 사용하는 메모리 – 스택(stack)
- 함수의 기능 수행이 끝나면 자동으로 반환되는 메모리
- 함수 호출과 스택 메모리 구조



클래스에 메서드 구현하기

- 클래스의 메서드는 멤버 변수를 사용하여 기능 구현
- 학생의 이름을 반환하는 메서드

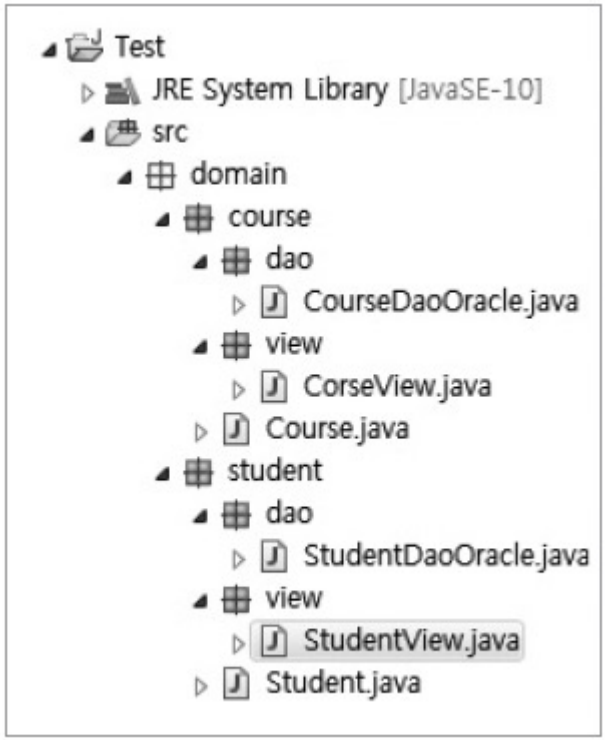
```
package classpart;

public class Student {
   int studentID;
   String studentName;
   int grade;
   String address;

   public String getStudentName( ) {
      return studentName;
   }
}
```

학생의 이름을 반환하는 메서드

패키지

- 소스의 묶음 카테고리
- 계층구조로 구성되며 소문자로 명명
- 패키지를 정의 하는 것은 소스 코드를 어떻게 구분하여 관리할 것인지를 정의 하는 것

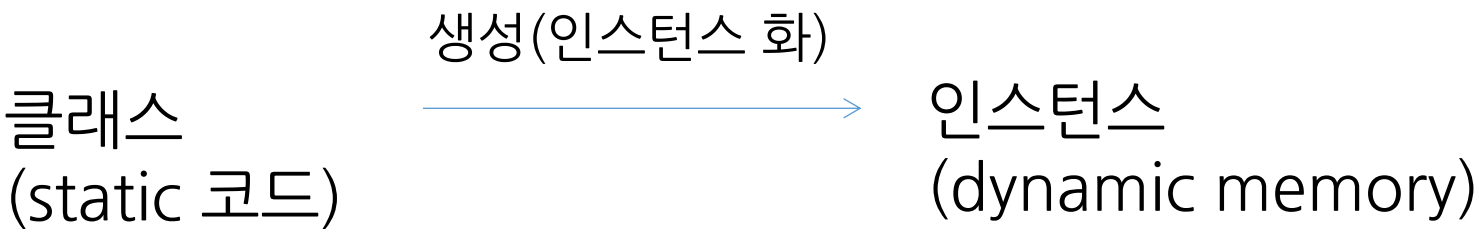
- 기본 클래스와
- 각 기능을 하는 클래스들을
- 패키지로 구분하여 관리 하는 예



naming convension

- 클래스 이름은 대문자로 시작
- 패키지의 이름은 소문자로
- 메서드나 변수의 이름은 소문자로 시작해서 단어마다 대문자로 씀 (camel notation)
- (반드시 지켜야 하는 사항은 아니지만, 일종의 관례)

class & instance



클래스 생성하기

- 클래스를 사용하기 위해서는 클래스를 생성하여야 함
- new 예약어를 이용하여 클래스 생성

- 클래스형 변수이름 = new 생성자;

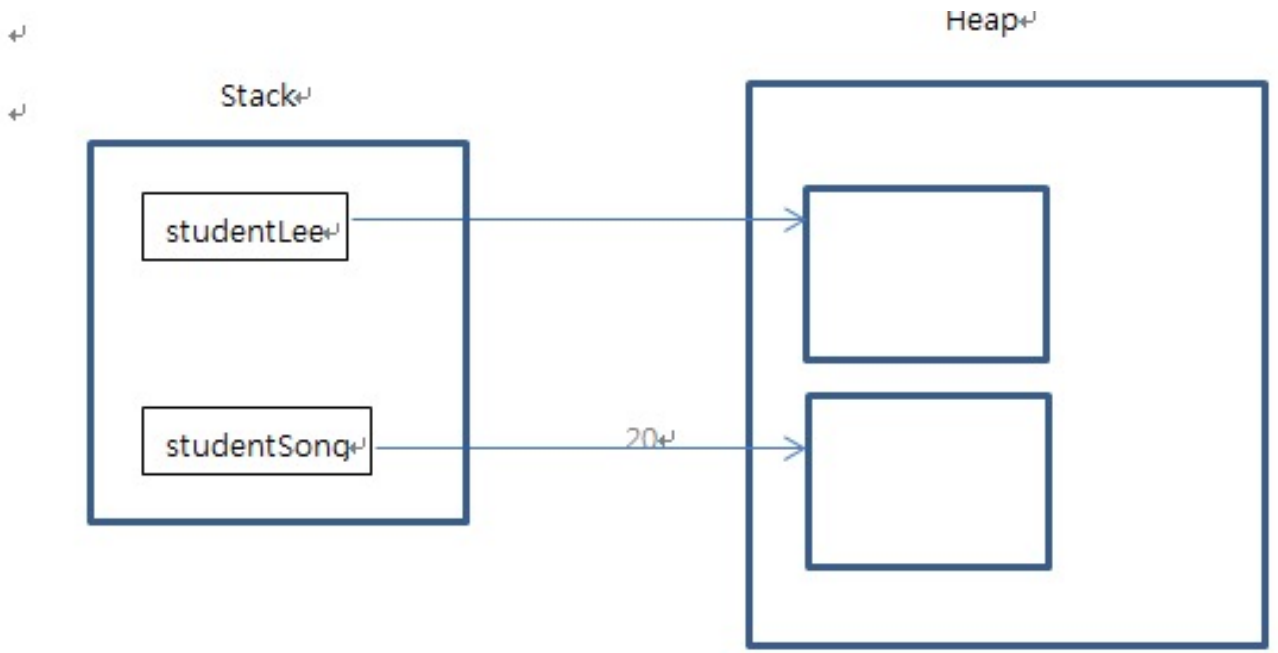
- Student studentA = new Student();

인스턴스 여러 개 생성하기

```java
package classpart;

public class StudentTest1 {

    public static void main(String[] args) {

        Student student1 = new Student();
        student1.studentName = "홍길동";

        System.out.println(student1.getStudentName());

        Student student2 = new Student();
        student2.studentName = "이순신";

        System.out.println(student2.getStudentName());
    }
}
```

인스턴스와 힙(heap) 메모리

- 하나의 클래스 코드로 부터 여러 개의 인스턴스를 생성
- 인스턴스는 힙(Heap) 메모리에 생성됨
- 각각의 인스턴스는 다른 메모리에 다른 값을 가짐



참조 변수와 참조 값

- 참조 변수 : 인스턴스 생성시 선언하는 변수
- 참조 값: 인스턴스 가 생성되는 힙 메모리 주소
- 참조 변수를 이용한 참조 값 확인 코드

```
public class StudentTest2 {

    public static void main(String[] args) {

        Student student1 = new Student();
        student1.studentName = "홍길동";

        Student student2 = new Student();
        student2.studentName = "이순신";

        System.out.println(student1);
        System.out.println(student2);
    }
}
```

```
Console  Problems  Javadoc  Declarati
<terminated> StudentTest2 [Java Application] C:₩Progran
classpart.Student@16f65612
classpart.Student@311d617d
```

클래스, 인스턴스, 참조변수, 참조값

용어	설명
객체	객체 지향 프로그램의 대상, 생성된 인스턴스
클래스	객체를 프로그래밍하기 위해 코드로 만든 상태
인스턴스	클래스가 메모리에 생성된 상태
멤버 변수	클래스의 속성, 특성
메서드	멤버 변수를 이용하여 클래스의 기능을 구현
참조 변수	메모리에 생성된 인스턴스를 가리키는 변수
참조 값	생성된 인스턴스의 메모리 주소 값

생성자(constructor)

- 인스턴스 생성 시 new 키워드와 함께 사용했던 생성자

```
package constructor;

public class Person {
   String name;
   float height;
   float weight;
}
```

```
package constructor;

public class PersonTest {
   public static void main(String[ ] args) {
      Person personLee = new Person( );
   }
}
```

생성자

생성자 (constructor)

- **생성자 기본 문법**
- <modifiers> <class_name>([<argument_list>])
- {
- [<statements>]
- }
- **생성자는 인스턴스를 초기화 할 때의 명령어 집합**
- **생성자의 이름은 그 클래스의 이름과 같음**
- **생성자는 메소드가 아님. 상속되지 않으며, 리턴 값은 없음**

디폴트 생성자 (default constructor)

- 하나의 클래스에는 반드시 적어도 하나 이상의 Constructor가 존재

- 프로그래머가 Constructor 를 기술하지 않으면 Default Constructor 가 자동으로 생김 (컴파일러가 코드에 넣어 줌)
- Default Constructor는 매개 변수가 없음
- Default Constructor는 구현부가 없음

- 만약 클래스에 매개변수가 있는 생성자를 추가하면 디폴트 생성자는 제공되지 않음

생성자 오버로드(constructor overload)

- 필요에 의해 생성자 추가 하는 경우 여러 개의 생성자가 하나의 클래스에 있음 (overload)

```
package constructor;

public class Person {
    String name;
    float height;
    float weight;

    public Person( ) { }        디폴트 생성자

    public Person(String pname) {
        name = pname;
    }                           이름을 매개변수로 입력받는 생성자

    public Person(String pname, float pheight, float pweight) {
        name = pname;
        height = pheight;
        weight = pweight;
    }                           이름, 키, 몸무게를 매개변수로 입력 받는 생성자
}
```

참조 자료형 (reference data type)

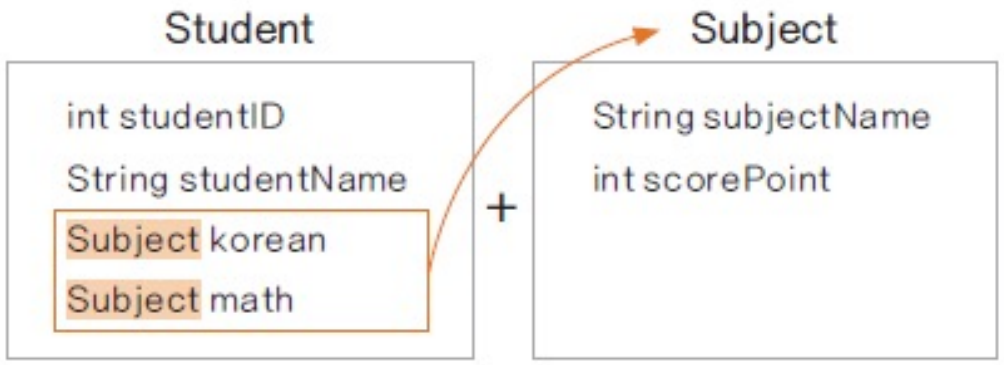
- 변수의 자료형

- 클래스 형으로 선언하는 자료형

참조 자료형의 예

- 학생의 속성 중 수업에 대한 부분
- 수업에 대한 각 속성을 학생 클래스에 정의 하지 않고
- 수업이라는 클래스로 분리해서 사용
- 이때 과목은 참조 자료형으로 선언

참조 자료형의 예




```
public class Student3 {
  int studentID;
  String studentName;
  Subject korean;
  Subject math;
}
```

Subject형을 사용하여 선언

```
public class Subject {
  String SubjectName;
  int scorePoint;
}
```

정보은닉 (information hiding)

- private 접근 제어자
- 클래스의 외부에서 클래스 내부의 멤버 변수나 메서드에
- 접근(access)하지 못하게 하는 경우 사용
- 멤버 변수나 메서드를 외부에서 사용하지 못하도록 하여 오류를
- 줄일 수 있음
- 변수에 대해서는 필요한 경우 get(), set() 메서드를 제공
-

정보 은닉의 예 public 으로 선언한 경우

```
public class MyDate {
   public int day;
   public int month;
   public int year;
}
```

```
public class MyDateTest {
   public static void main(String[ ] args) {
      MyDate date = new MyDate( );
      date.month = 2;
      date.day = 31;
      date.year = 2018;
   }
}
```

2월에 31일이 존재하지 않는데 오류를 막을 수 없음

정보 은닉의 예 - private 으로 선언한 경우

```
public class MyDate {
   private int day;
   private int month;
   private int year;

   public setDay(int day) {
      if(month == 2) {
          if(day < 1 || day > 28) {
            System.out.println("오류입니다");
          } else {
            this.day = day;
          }
       }
   }
}
```

날짜가 변수에 저장되기 전에 체크하여 오류를 방지 할 수 있음

감사합니다.

끝