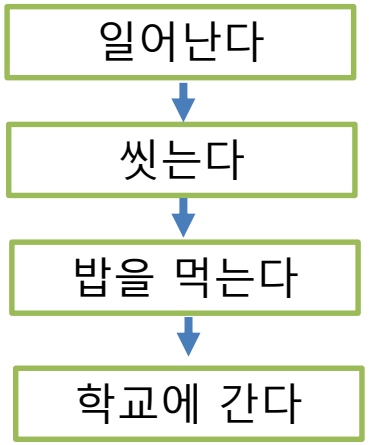
5장. 클래스와 객체

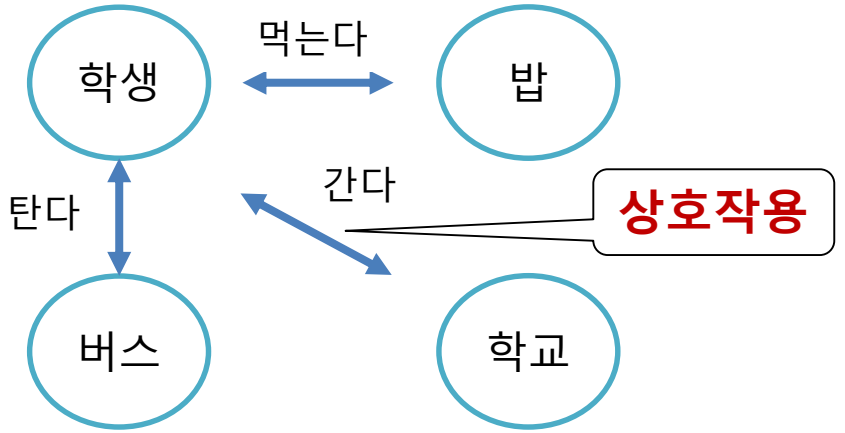
class & instance

객체 지향 프로그래밍

- **객체(Object)란?**
  - 의사나 행위가 미치는 대상 -> 사전적 의미
  - 구체적, 추상적 데이터 단위 (구체적- 책상, 추상적-회사)
- **객체지향 프로그래밍(Objected Oriented Programming, OOP)**
  - 객체를 기반으로 하는 프로그래밍
  - 먼저 객체를 만들고, 객체 사이에 일어나는 일을 구현함.



<절차지향 –C언어>



<객체지향 –Java>

클래스(class)

- **클래스란?**

  객체에 대한 속성과 기능을 코드로 구현 한 것

  "클래스를 정의 한다"라고 하고, 객체에 대한 설계도 또는 청사진.

- **객체의 속성과 기능**
  - 객체의 특성(property), 속성(attribute) -> **멤버 변수**
  - 객체가 하는 기능 -> **메서드(멤버 함수)**

학생 클래스
- 속성(멤버변수) : 이름, 나이, 학년, 사는 곳 등..
- 기능(메서드) : 수강신청, 수업듣기, 시험 보기 등..

클래스(class)

▪ 클래스 정의하기

- 하나의 java파일에 하나의 클래스를 두는 것이 원칙이나, 여러 개의 클래스가 같이 있는 경우 **public** 클래스는 단 하나이다.
- public 클래스와 java파일의 이름은 동일해야 하고, 클래스 이름은 대문자로 시작한다.

```
(접근제어자) class 클래스 이름{
        멤버 변수;
        메서드;
}
```

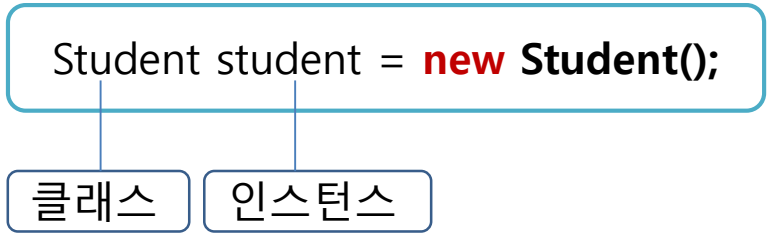
클래스의 정의와 사용

- 학생 클래스 정의

```
package classes;

public class Student {
    int studentId;     //학번
    String name;       //이름
    int grade;         //학년

    //학생 정보 출력 메서드
    void displayInfo() {
        System.out.println("학번 : " + studentId);
        System.out.println("이름 : " + name);
        System.out.println("학년 : " + grade);
    }
}
```

클래스의 정의와 사용

- **학생 클래스의 사용**
  - 메인 메소드(함수)가 있는 클래스에서 실행 사용할 수 있음
  - 클래스에서 **new** 연산자를 사용하여 객체를 생성해야 함.
  - 객체변수.멤버변수->점(.) 연산자를 사용하여 접근함



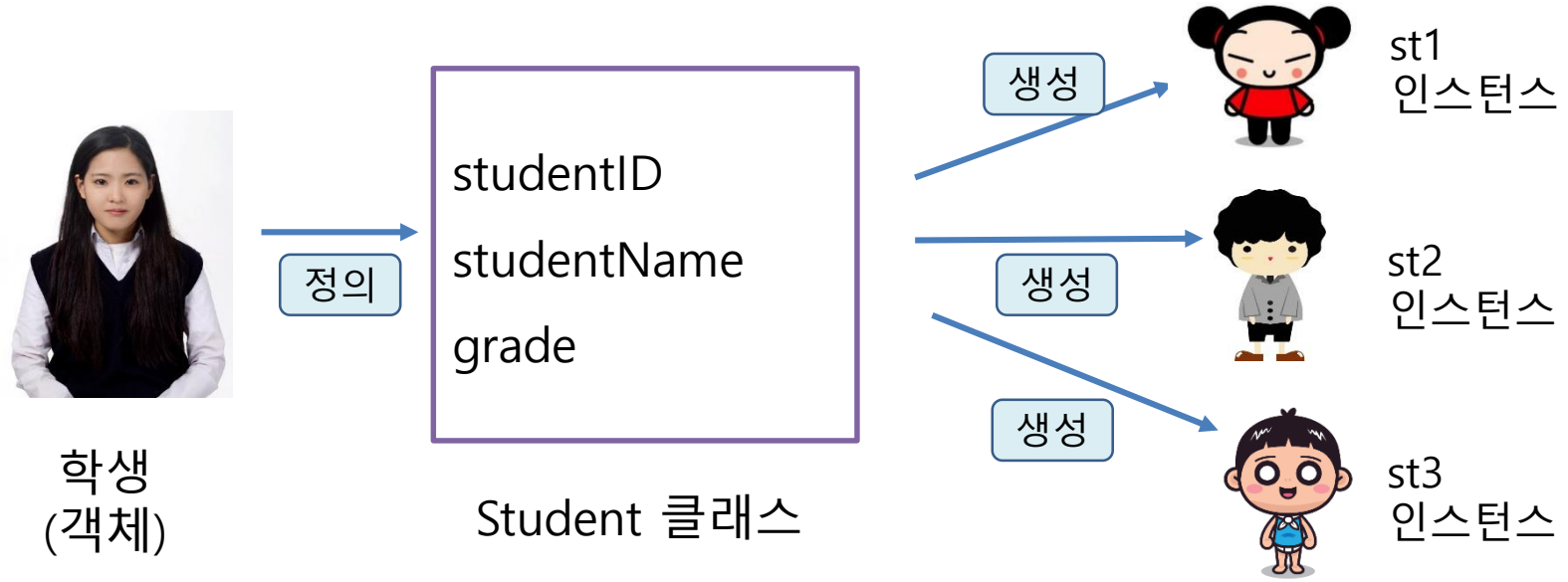
클래스의 정의와 사용

- 학생 클래스 테스트

```
public class StudentTest {
    public static void main(String[] args) {
        //Student 클래스의 인스턴스 생성
        Student s1 = new Student();

        s1.studentId = 1001;
        s1.name = "정은하";
        s1.grade = 3;

        /*System.out.println("학번 : " + s1.studentId);
        System.out.println("이름 : " + s1.name);
        System.out.println("학년 : " + s1.grade);*/

        //학생 정보 출력 - 메서드 호출
        s1.displayInfo();

        //인스턴스 정보 출력
        System.out.println(s1);
    }
}
```

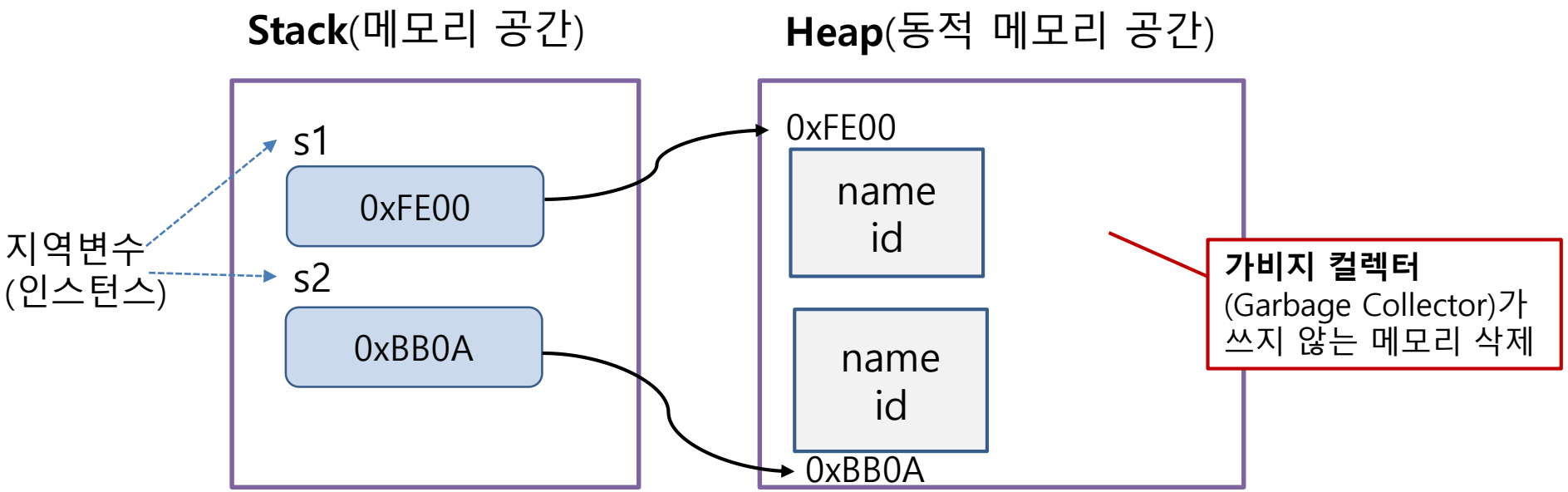
클래스와 인스턴스

- **객체, 클래스, 인스턴스**
  - 객체 : '의사나 행위가 미치는 대상'
  - 클래스 : 객체를 코드로 구현한 것
  - 인스턴스 : 클래스가 메모리 공간에 생성된 상태.



인스턴스와 참조변수

- **인스턴스와 힙 메모리**
  - 하나의 클래스 코드로부터 여러 개의 인스턴스를 생성
  - 인스턴스는 힙(Heap) 메모리에 생성됨
  - 각각의 인스턴스는 다른 메모리에 다른 참조값을 가짐(주소값으로 해시 코드[hash code]값이라고도 한다.)



객체 자료형

- **변수의 자료형**

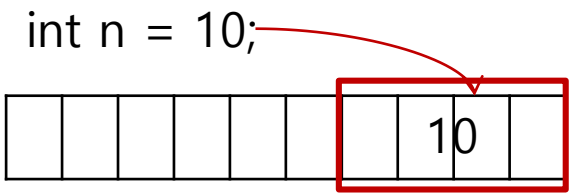
기본 자료형(Primitive)

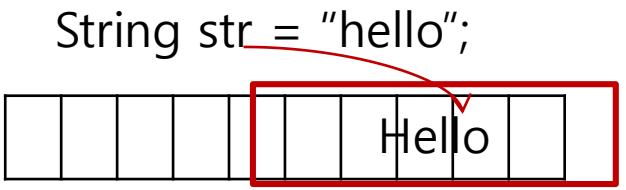
Java 언어에 이미 존재하고 있는 데이터 타입, 주로 간단한 데이터들이다.

(int, double, boolean, char 등)

객체 자료형(Object)

여러가지 데이터 타입으로 구성된 자료형(클래스)으로 기본 자료형에 비해 크기가 크다.(String, System, ArrayList 등)





패키지(package)

- **패키지란?**
  - 클래스 파일의 묶음이다.
  - 패키지를 만들면 프로젝트 하위에 물리적으로 디렉터리가 생성된다.
  - 클래스의 실제 이름은 패키지이름.클래스이름 이다.

```
package classes;

public class StudentTest {
    public static void main(String[] args) {
        //Student 클래스의 인스턴스 생성
        Student s1 = new Student();

        //인스턴스 정보 출력
        System.out.println(s1);
```

패키지 이름

클래스 이름

```
classes.Student@433c675d
```

생성자(constructor)

❖ 생성자(Constructor)

- 생성자는 클래스를 생성할 때만 호출한다.
- 생성자 이름은 클래스 이름과 같고, 생성자는 반환값(return)이 없다.
- 매개변수가 없는 생성자를 ***기본 생성자***라 하며, 생략할 수 있다.

  생략하여도 컴파일러가 자동으로 생성해 준다.

```
package constructor;

public class Student {
    int studentId;          //학번
    String name;            //이름
    int grade;              //학년

    Student(){}    //기본 생성자(생략 가능)
```

생성자(constructor)

❖ 생성자 오버로딩(overloading)

- 클래스에 생성자가 두 개 이상 제공되는 경우를 말한다.
- 이름은 같고, 매개 변수가 다른 생성자를 여러 개 만들수 있다.

```
public class Student {
    int studentId;          //학번
    String name;            //이름
    int grade;              //학년

    Student(){}    //기본 생성자(생략 가능)

    //매개 변수를 가진 생성자
    Student(int _studentId, String _name, int _grade){
        studentId = _studentId;
        name = _name;
        grade = _grade;
    }
```

생성자(constructor)

❖ 생성자 오버로딩(overloading)

```
public class StudentTest {

    public static void main(String[] args) {
        //Student 클래스의 인스턴스 s1 생성
        Student s1 = new Student();  //기본 생성자로 생성

        s1.studentId = 1001;
        s1.name = "정은하";
        s1.grade = 3;

        //매개값을 입력한 인스턴스 s2 .todtjd
        Student s2 = new Student(1002, "이우주", 1);

        //정보 출력
        s1.displayInfo();
        s2.displayInfo();
    }
}
```

회원 로그인 서비스 클래스

❖ 로그인/로그아웃을 서비스하는 클래스

```
public class MemberService {

    //로그인 일치 여부를 반환하는 메서드
    public boolean login(String id, String password) {
        if(id.equals("hangang") && password.equals("k2025"))
            return true;
        return false;
    }

    //로그아웃을 실행하는 메서드
    public void logout(String id) {
        System.out.println("로그아웃 되었습니다.");
    }
}
```

회원 로그인 서비스 클래스

❖ 로그인/로그아웃을 서비스하는 클래스

```
public class MemberServiceTest {

    public static void main(String[] args) {
        //memberService 객체 생성
        MemberService memberService = new MemberService();

        //로그인을 위해 아이디, 비밀번호 입력
        boolean result = memberService.login("hangang", "k2025");
        if(result) {
            System.out.println("로그인 되었습니다.");
            memberService.logout("hangang");
        }else {
            System.out.println("id 또는 password가 올바르지 않습니다.");
        }
    }
}
```

this 예약어

- **자신의 메모리를 가리키는 this**

  생성된 인스턴스 스스로를 가리키는 예약어

```
package thissample;
class BirthDay{
    int day;
    int month;
    int year;

    public void setYear(int year) {
        this.year = year;
    }

    public void printThis() {
        System.out.println(this);
    }
}
```

this 예약어

- 자신의 메모리를 가리키는 this

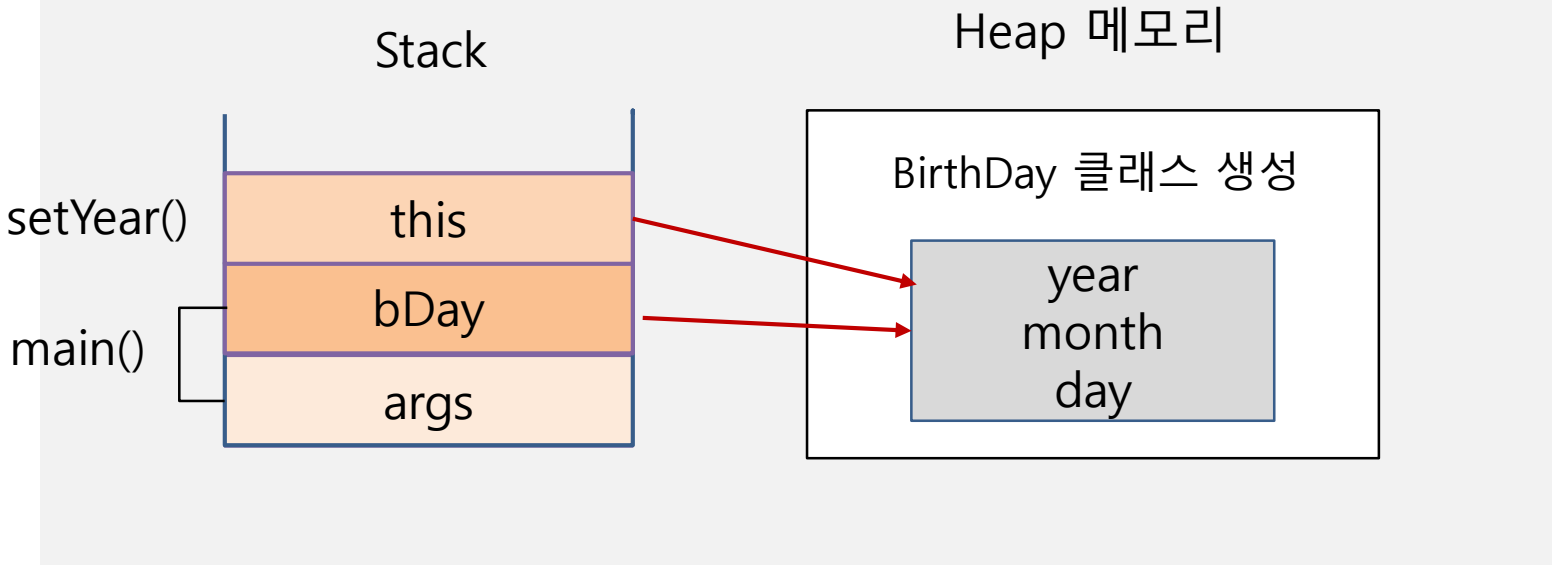
```
public class ThisTest {

    public static void main(String[] args) {
        BirthDay bDay = new BirthDay();
        bDay.setYear(2020);

        System.out.println(bDay);
        bDay.printThis();

        //인스턴스를 출력하면 클래스이름@메모리 주소
    }
}
```

클래스이름@메모리 주소 →

```
thissample.BirthDay@7d6f77cc
thissample.BirthDay@7d6f77cc
```

this 예약어

- **this 주소(참조값) 확인**



main() 함수에서 bDay 변수가 가리키는 인스턴스와 Birthday 클래스의 setYear() 메서드에서 this가 가리키는 인스턴스가 같은 곳에 있음을 알 수 있다.

this 예약어

- 생성자에서 다른 생성자를 호출하는 this

```
package thissample;
class Person{
    String name;
    int age;

    Person(){  //this를 사용해 Person(String, int) 생성자 호출
        this("이름 없음", 1);
    }

    Person(String name, int age){
        this.name = name;
        this.age = age;
    }

    Person returnItSelf() {  //반환형은 클래스형
        return this;
    }
}
```

this 예약어

- 생성자에서 다른 생성자를 호출하는 this

```
public class CallAnotherConst {

    public static void main(String[] args) {
        Person noName = new Person();
        System.out.println(noName.name);
        System.out.println(noName.age);

        Person p = noName.returnItSelf();

        System.out.println(p);
        System.out.println(noName);
    }
}
```

```
이름 없음
1
thissample.Person@7d6f77cc
thissample.Person@7d6f77cc
```

정보 은닉(캡슐화)

- **정보 은닉(Information Hiding)**
  - **접근 제어자 : 접근 권한 지정**
    - **public : 외부 클래스에서 접근 가능**
    - **private : 클래스의 외부에서 클래스 내부의 멤버 변수나 메서드에 접근 못하게 하는 경우 사용**
  - **변수에 대해서는 필요한 경우 get(), set() 메서드를 제공**

접근 제어자	설 명
public	외부 클래스 어디에서나 접근 할수 있다.
protected	같은 패키지 내부와 상속 관계의 클래스에서만 접근(다른 패키지에서도 가능)
없는 경우	default이며 같은 패키지 내부에서만 접근 가능
private	같은 클래스 내부 가능, 그 외 접근 불가

정보 은닉(캡슐화)

- **private 접근 제한자**

```
public class Account {
    private String ano;     //계좌 번호
    private String owner;   //계좌주
    private int balance;    //잔액
```

```
public class AccountTest {

    public static void main(String[] args) {
        Account account1 = new Account();
        //account.ano = "100-1000";
        //private 멤버는 접근 불가
```

정보 은닉(캡슐화)

- get(), set() 메서드 사용하여 private 변수에 접근가능

set + 멤버변수이름(){ }

get + 멤버변수이름(){ };

```
public String getAno() {
    return ano;
}

public void setAno(String ano) {
    this.ano = ano;
}

public String getOwner() {
    return owner;
}

public void setOwner(String owner) {
    this.owner = owner;
}

public int getBalance() {
    return balance;
}

public void setBalance(int balance) {
    this.balance = balance;
}
```

정보 은닉(캡슐화)

- **Account 객체 생성**

```
//Account 객체 생성 - 기본 생성자
Account account1 = new Account();
//account1.ano = "111-222";  //private 멤버에 접근 불가

//데이터 입력
account1.setAno("111-222");
account1.setOwner("나저축");
account1.setBalance(10000);

//데이터 출력
System.out.println("계좌번호: " + account1.getAno());
System.out.println("계좌주: " + account1.getOwner());
System.out.println("잔고: " + account1.getBalance());
System.out.println("========================");
```

정보 은닉(캡슐화)

- **매개변수 있는 생성자 만들기**

매개변수 이름과 this 멤버 이름이 같아야 한다.

```
public class Account {
    private String ano;
    private String owner;
    private int balance;

    public Account() {};  //기본 생성자

    //매개 변수가 있는 생성자 - this로 멤버 초기화
    public Account(String ano, String owner, int balance) {
        this.ano = ano;
        this.owner = owner;
        this.balance = balance;
    }
```

정보 은닉(캡슐화)

- **Account 클래스 테스트**

```
//생성자 매개변수 외부 입력으로 객체 생성
Account account2 = new Account("333-444", "최금리", 20000);

//데이터 출력
System.out.println("계좌번호: " + account2.getAno());
System.out.println("계좌주: " + account2.getOwner());
System.out.println("잔고: " + account2.getBalance());
System.out.println("========================");
```

```
계좌번호: 111-222
계좌주: 나저축
잔고: 10000
========================
계좌번호: 333-444
계좌주: 최금리
잔고: 20000
========================
```

객체 배열 만들기

- **객체 배열**

  동일한 기본 자료형(int 등) 변수 여러 개를 배열로 사용할 수 있듯이 참조 자료형 변수도 여러 개를 배열로 사용할 수 있다.

```
package objects;

public class Book {
    private int bookNumber;
    private String bookName;
    private String author;

    public Book(int bookNumber, String bookName, String author) {
        this.bookNumber = bookNumber;
        this.bookName = bookName;
        this.author = author;
    }

    public void showBookInfo() {
        System.out.println(bookNumber + ": " + bookName + ", " + author);
    }
}
```

객체 배열

- **객체 배열 만들기**
  - 배열만 생성한 경우 요소는 null로 초기화 됨

```
public class BookArray {

    public static void main(String[] args) {
        //객체 배열 생성 방법 1
        Book[] books = new Book[3];

        //null 출력
        for(int i=0; i<books.length; i++) {
            System.out.println(books[i]);
        }
```

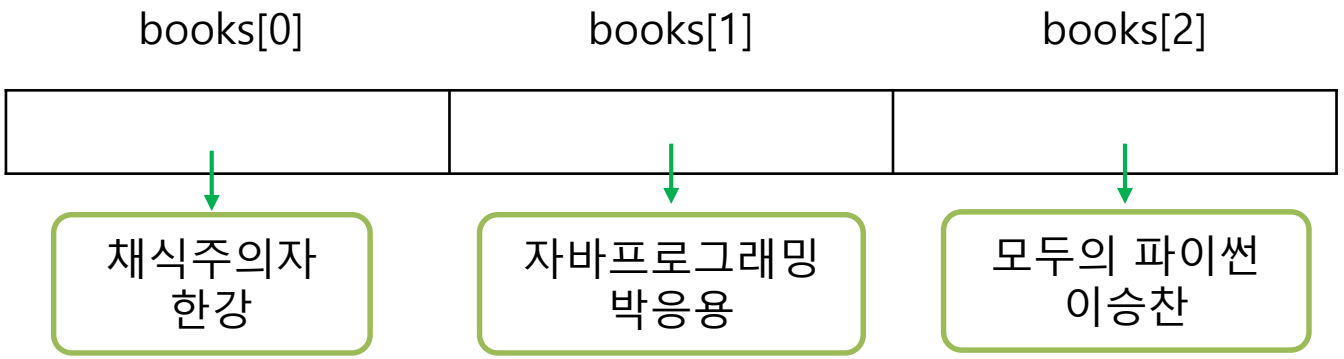
books[0]	books[1]	books[2]
null	null	null

객체 배열 만들기

- **객체 배열**

```
//Book 객체 생성
Book book1 = new Book(100, "채식주의자", "한강");
Book book2 = new Book(101, "자바프로그래밍 입문", "박은종");
Book book3 = new Book(102, "모두의 파이썬", "이승찬");

books[0] = book1;
books[1] = book2;
books[2] = book3;

//특정 객체 검색
books[0].showBookInfo();

//전체 출력
for(int i=0; i<books.length; i++) {
    books[i].showBookInfo();
}
```

```
100: 채식주의자, 한강
101: 자바프로그래밍 입문, 박은종
102: 모두의 파이썬, 이승찬
```



객체 배열 만들기

- **객체 배열**

```
//객체 배열 생성 방법 2
Book[] books = new Book[3];

//배열의 저장
books[0] = new Book(100, "채식주의자", "한강");
books[1] = new Book(101, "자바프로그래밍 입문", "박은종");
books[2] = new Book(102, "모두의 파이썬", "이승찬");
```

```
//객체 배열 생성 방법 3
Book[] books = {
        new Book(100, "채식주의자", "한강"),
        new Book(101, "자바프로그래밍 입문", "박은종"),
        new Book(102, "모두의 파이썬", "이승찬")
};
```

클래스(객체)간 참조

- **클래스 간 참조**

Point 클래스

```
public class Point { //점
    int x;
    int y;
}
```

Circle 클래스

```
public class Circle { //원
    Point center;  //중심점
    int radius;    //반지름
}
```

Point 클래스(자료형)를 참조

클래스(객체)간 참조

- Point 클래스

```
package reference;

public class Point {
    private int x;
    private int y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }
}
```

클래스(객체)간 참조

- Circle 클래스

```
public class Circle {
    Point center;  //중심점
    int radius;    //반지름

    public Circle(int x, int y, int radius) {
        center = new Point(x, y); //Point 객체 생성
        this.radius = radius;
    }

    public void showInfo() {
        System.out.println("원의 중심은 (" + center.getX() + ", " +
                center.getY() + ")이고, 반지름은 " + radius + "입니다.");
    }
}
```

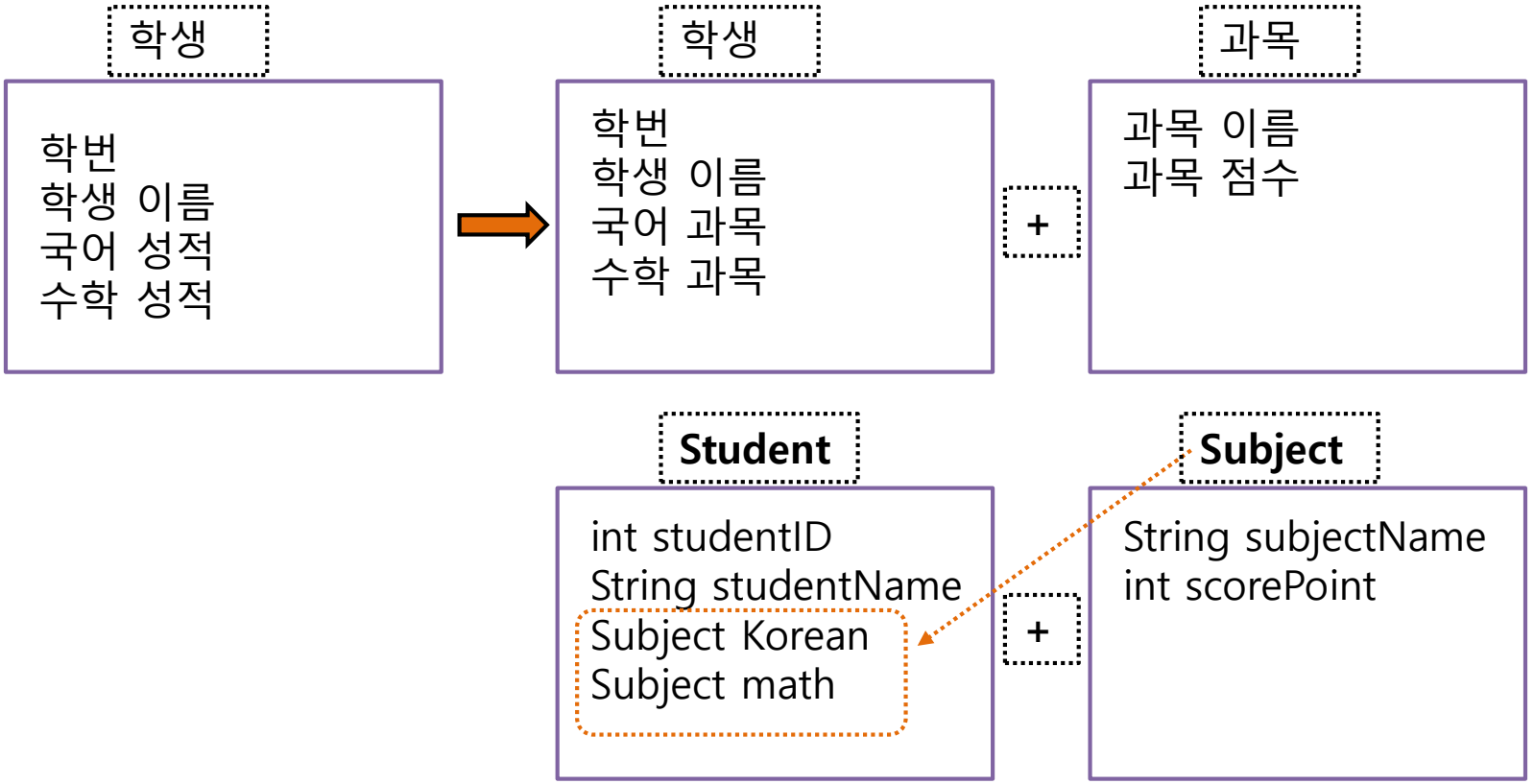
클래스(객체)간 참조

- Circle 테스트

```
public class CircleTest {

    public static void main(String[] args) {
        // Circle 객체 생성
        Circle c1 = new Circle(2, 3, 5);
        Circle c2 = new Circle(8, 8, 10);

        System.out.println("=============== 원의 정보 ===============");
        c1.showInfo();
        c2.showInfo();
    }
}
```

```
=============== 원의 정보 ===============
원의 중심은 (2, 3)이고, 반지름은 5입니다.
원의 중심은 (8, 8)이고, 반지름은 10입니다.
```

클래스(자료형) 참조



문제점 : 이 클래스는 학생에 대한 클래스인데 과목 변수가 계속 늘어남

해결책 : 과목이름과 성적을 과목(Subject) 클래스로 분리함.

클래스(자료형) 참조

- 과목 클래스

```
public class Subject {
    private String subjectName;   //과목명
    private int scorePoint;       //점수

    //과목 설정
    public void setSubjectName(String subjectName) {
        this.subjectName = subjectName;
    }

    public String getSubjectName() {
        return subjectName;
    }

    //점수 설정
    public void setScorePoint(int scorePoint) {
        this.scorePoint = scorePoint;
    }

    public int getScorePoint() {
        return scorePoint;
    }
}
```

클래스(자료형) 참조

- 학생 클래스

```
public class Student {
    private int studentId;       //학번
    private String studentName; //이름
    private Subject korean;      //국어
    private Subject math;        //수학

    public Student(int studentId, String studentName) {
        this.studentId = studentId;
        this.studentName = studentName;
        korean = new Subject();
        math = new Subject();
    }

    //국어 점수 설정
    public void setKoreanSubject(String name, int score) {
        korean.setSubjectName(name);
        korean.setScorePoint(score);
    }
```

클래스(자료형) 참조

- 학생 클래스

```
    //수학 점수 설정
    public void setMathSubject(String name, int score) {
        math.setSubjectName(name);
        math.setScorePoint(score);
    }

    //학생의 정보
    public void showInfo() {
        System.out.println(
                        "학번: " + studentId +
                        "\n이름: " + studentName +
                        "\n국어 점수: " + korean.getScorePoint() +
                        "\n수학 점수: " + math.getScorePoint());
        System.out.println("-----------------------");
    }
}
```

클래스(자료형) 참조

- ScoreMain 테스트

```
public class ScoreMain {

    public static void main(String[] args) {
        //학생 객체 생성
        Student lee = new Student(1001, "이정후");
        lee.setKoreanSubject("국어", 90);
        lee.setMathSubject("수학", 85);
        lee.showInfo();

        Student shin = new Student(1002, "신유빈");
        shin.setKoreanSubject("국어", 95);
        shin.setMathSubject("수학", 80);
        shin.showInfo();
    }
}
```

```
학번: 1001
이름: 이정후
국어 점수: 90
수학 점수: 85
------------------------
학번: 1002
이름: 신유빈
국어 점수: 95
수학 점수: 80
------------------------
```

배열로 성적 관리하기

- 학생 성적 출력 프로그램(배열로 구현)

```
public class Student {
    private int studentId;       //학번
    private String studentName; //이름
    private Subject[] subjects; //

    public Student(int studentId, String studentName) {
        this.studentId = studentId;
        this.studentName = studentName;
        subjects = new Subject[10];
    }

    //과목 추가
    public void addSubject(String name, int score) {
        Subject subject = new Subject(); //과목 객체 1개 생성
        subject.setSubjectName(name);
        subject.setScorePoint(score);

        //생성된 과목 객체를 배열에 저장
        for(int i=0; i<subjects.length; i++) {
            if(subjects[i] == null) { //배열 공간이 비어있으면
                subjects[i] = subject; //배열에 저장
                break; //매번 빠져나옴
            }
        }
    }
```

배열로 성적 관리하기

- **학생 성적 출력 프로그램(배열로 구현)**

```
    //학생의 정보와 평균 계산
    public void displayInfo() {
        int total = 0;  //총점
        double avg;     //평균
        int count = 0;  //개수

        System.out.println(   //학생 정보 출력
                "학번: " + studentId +
                "\n이름: " + studentName);
        for(int i=0; i<subjects.length; i++) {
            if(subjects[i] != null) { //배열의 공간이 비어있지 않으면
                total += subjects[i].getScorePoint(); //점수 더하기
                count++;  //배열에 저장된 객체의 개수

                System.out.println(   //과목 점수 출력
                        subjects[i].getSubjectName() +
                        "점수: " + subjects[i].getScorePoint());
            }
        }
        //평균 계산
        avg = (double)total / count;
        System.out.printf("평균 점수: %.1f점", avg);
        System.out.println("\n========================");
    }
}
```

배열로 성적 관리하기

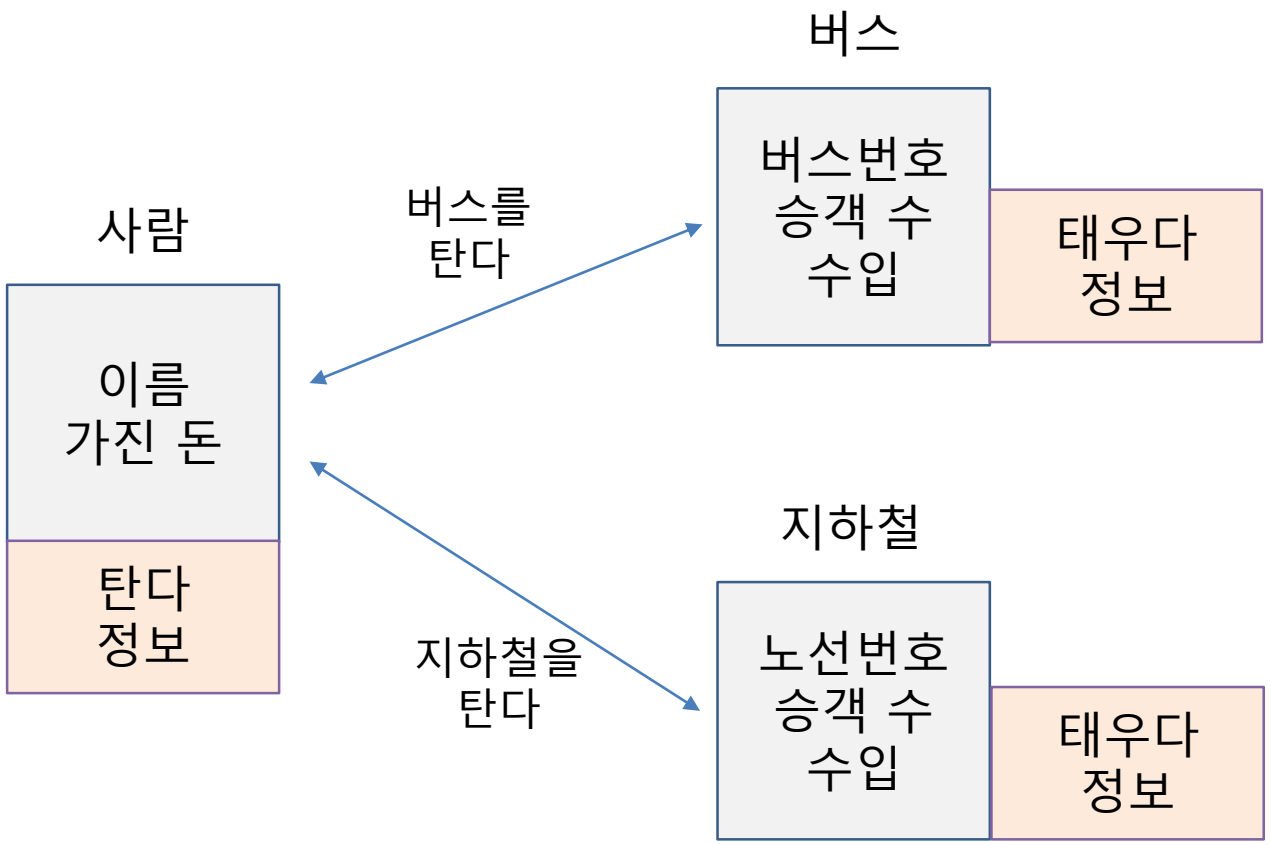
- **ScoreMain 테스트**

```
public class ScoreMain {
    public static void main(String[] args) {

        Student lee = new Student(1001, "이정후");
        //과목 확장
        lee.addSubject("국어", 90);
        lee.addSubject("수학", 85);
        lee.addSubject("과학", 80);

        //정보 출력
        lee.displayInfo();

        Student shin = new Student(1002, "신유빈");

        shin.addSubject("국어", 92);
        shin.addSubject("수학", 80);
        shin.addSubject("과학", 79);

        shin.displayInfo();
    }
}
```

```
학번: 1001
이름: 이정후
국어점수: 90
수학점수: 85
과학점수: 80
평균 점수: 85.0점
========================
학번: 1002
이름: 신유빈
국어점수: 92
수학점수: 80
과학점수: 79
평균 점수: 83.7점
========================
```

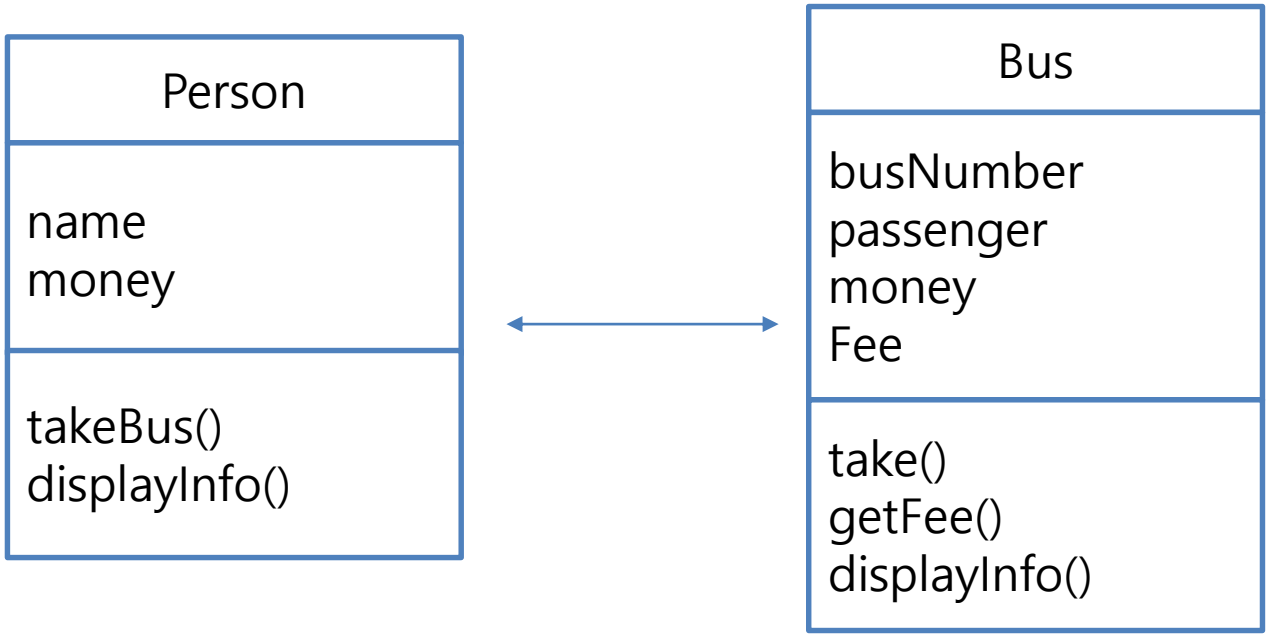
객체 간 협력

- 사람이 버스나 지하철을 타는 상황을 객체 지향으로 프로그래밍하기



객체 간 협력

- 클래스 다이어그램



객체 간 협력

- 버스 클래스

```
package transport;

public class Bus {

    private int busNumber;   //버스 번호
    private int passenger;   //승객수
    private int money;       //버스의 수입
    private static final int FEE = 1500; //요금(상수)

    public Bus(int busNumber) {
        this.busNumber = busNumber;
    }

    public void take() {
        this.money += FEE;   //요금 증가
        passenger++;         //승객 1명 증가
    }

    int getFee() { return FEE;} //요금 반환

    public void displayInfo() {
        System.out.println(busNumber + "번 버스의 수입은 " + money +
                "원이고, 승객수는 " + passenger + "명 입니다.");
    }
}
```

객체 간 협력

- **Person 클래스**

```
public class Person {
    private String name; //이름
    private int money;   //가진 돈

    public Person(String name, int money) {
        this.name = name;
        this.money = money;
    }

    public void takeBus(Bus bus) { //bus 인스턴스를 매개 변수로 전달
        if(this.money >= bus.getFee()) {
            bus.take();
            this.money -= bus.getFee();
        }
        else {
            System.out.println("잔액 부족!!");
        }
    }

    public void displayInfo() {
        System.out.println(name + "님의 남은 돈은 " +
                           money + "원 입니다.");
    }
}
```

객체 간 협력

- **Main 클래스**

```
public class Main {

    public static void main(String[] args) {

        Person lee = new Person("이정우", 10000);
        Person shin = new Person("신유진", 2000);
        Bus bus740 = new Bus(740);

        //버스 탑승
        lee.takeBus(bus740);
        shin.takeBus(bus740);
        shin.takeBus(bus740); //잔액 부족

        //정보 출력
        lee.displayInfo();
        shin.displayInfo();
        bus740.displayInfo();
    }
}
```

```
잔액 부족!!
이정우님의 남은 돈은 8500원 입니다.
신유진님의 남은 돈은 500원 입니다.
740번 버스의 수입은 3000원이고, 승객수는 2명 입니다.
```

실습 문제 – 클래스 구현

회원(Member) 클래스를 정의하고 배열로 객체를 생성하여 테스트하세요.

[파일이름: Member.java, MemberTest.java]

데이터 이름	필드 이름	타입	접근 제어
아이디	id	문자열	private
패스워드	password	문자열	private

☞ 실행 결과

```
********** 회원 현황 **********
id: 정은하, password: j1234
id: 김우주, password: k0000
id: 박하늘, password: p4320
```