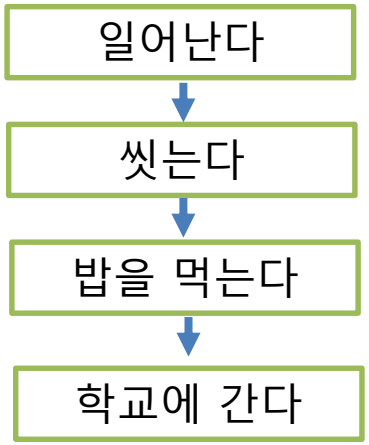
5장. 클래스와 객체

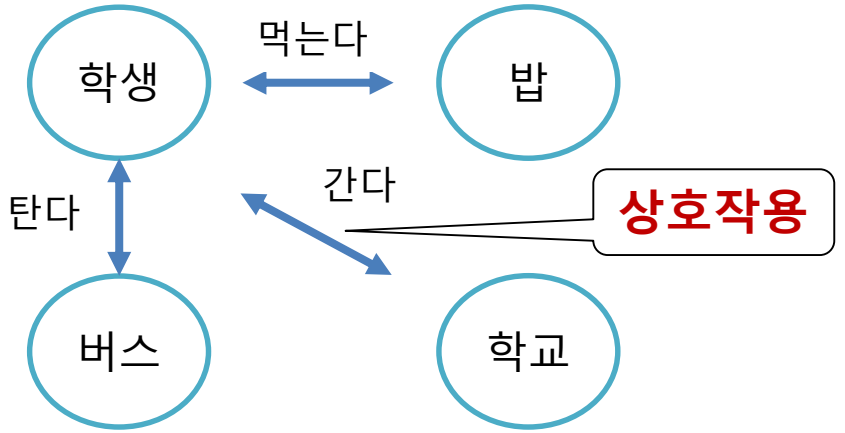
Class & Instance

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.

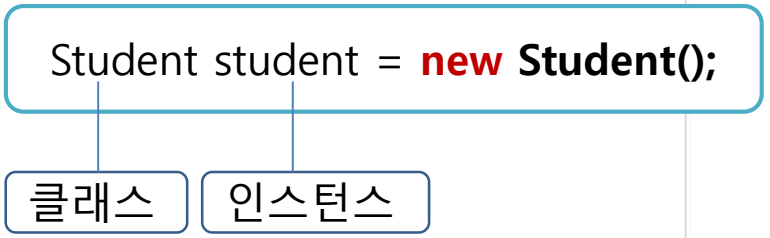
```
(접근제어자) class 클래스 이름{
        멤버 변수;
        메서드;
}
```

Student 클래스 →

```
package classpart;

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

객체(Object)

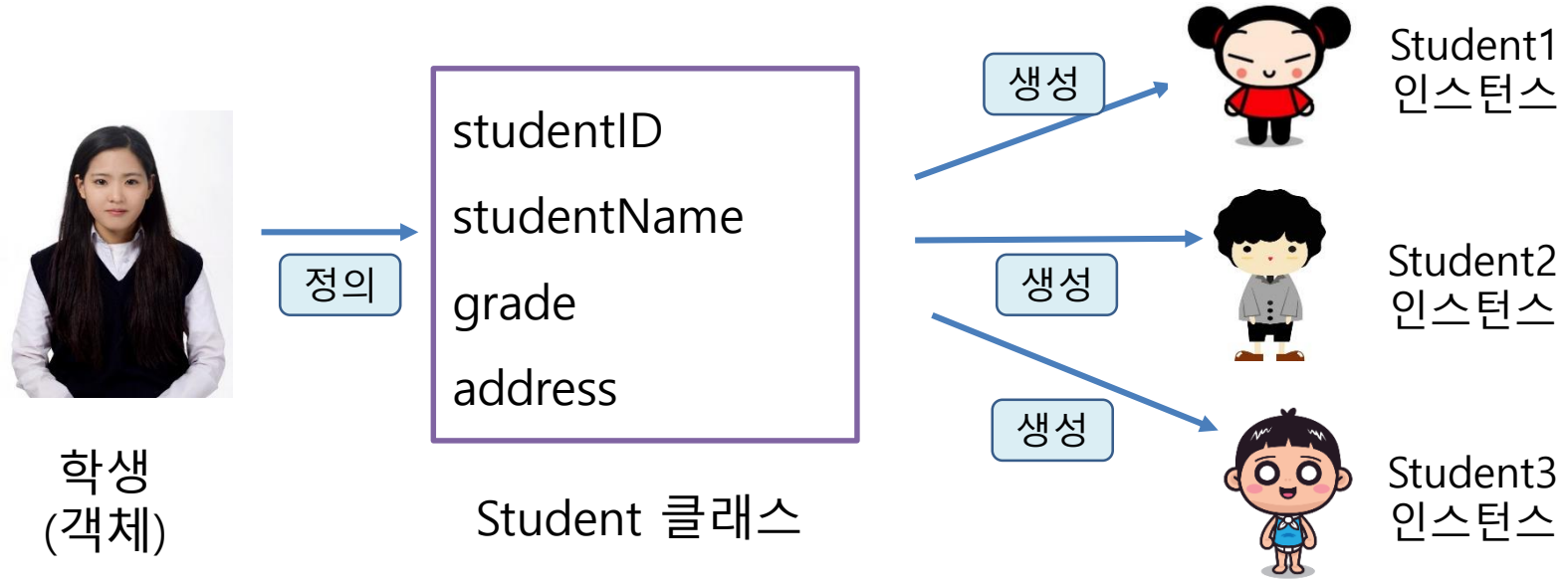
- **학생 클래스의 사용**
  - 메인 메소드(함수)가 있는 클래스에서 실행 사용할 수 있음
  - 클래스에서 **new** 연산자를 사용하여 객체를 생성해야 함.
  - 객체변수.멤버변수->점(.) 연산자를 사용하여 접근함




```java
public class StudentTest {

    public static void main(String[] args) {
        // Stuednt 클래스 사용하기
        Student student = new Student(); //student 객체 생성

        student.studentID = 1001;
        student.studentName = "홍길동";
        student.grade = 3;
        student.address = "서울시 구로구";

        System.out.println("학번 : " + student.studentID);
        System.out.println("이름 : " + student.studentName);
        System.out.println("학년 : " + student.grade);
        System.out.println("주소 : " + student.address);
    }
}
```

클래스와 인스턴스

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



인스턴스와 참조 변수

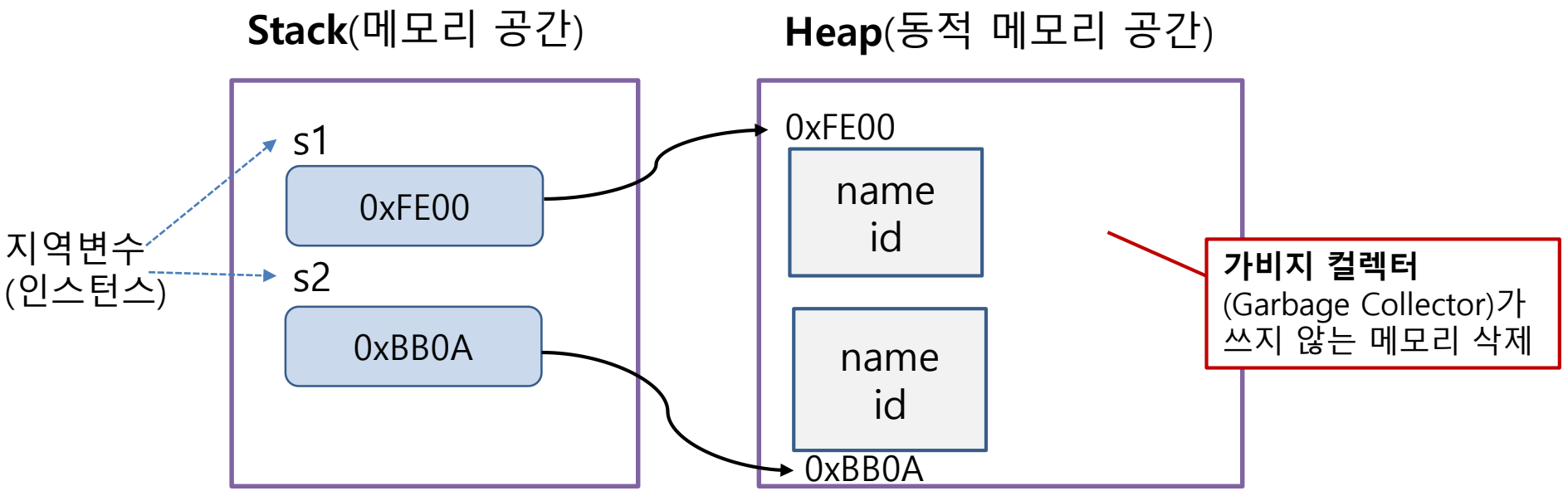
- 인스턴스 여러 개 생성하기

```
Student s1 = new Student(); //s1 인스턴스 생성
Student s2 = new Student(); //s2 인스턴스 생성

s1.studentID = 1001;
s1.studentName = "홍길동";
s1.grade = 3;
s1.address = "서울시 구로구";

s2.studentID = 1002;
s2.studentName = "이순신";
s2.grade = 2;
s2.address = "경기도 평택시";

System.out.println("학번 : " + s1.studentID);
System.out.println("이름 : " + s1.studentName);
System.out.println("학년 : " + s1.grade);
System.out.println("주소 : " + s1.address);

System.out.println("------------------------");
System.out.println("학번 : " + s2.studentID);
System.out.println("이름 : " + s2.studentName);
System.out.println("학년 : " + s2.grade);
System.out.println("주소 : " + s2.address);
```

```
학번 : 1001
이름 : 홍길동
학년 : 3
주소 : 서울시 구로구
----------------------
학번 : 1002
이름 : 이순신
학년 : 2
주소 : 경기도 평택시
```

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**

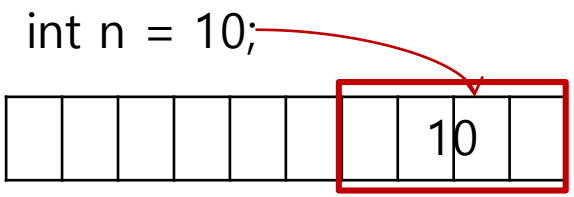
기본 자료형(Primitive)

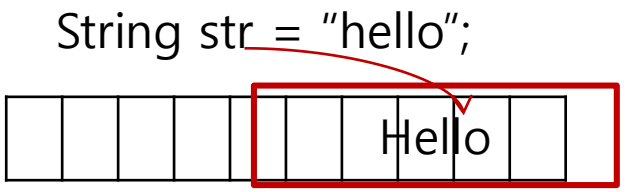
Java 언어에 이미 존재하고 있는 데이터 타입, 주로 간단한 데이터들이다.

(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. (예:classpart.Student)

```
package classpart;

public class StudentTest {

    public static void main(String[] args) {
        // Stuednt 클래스 사용하기
        Student student = new Student(); //student 객체 생성

        student.studentID = 1001;
        student.studentName = "홍길동";
        student.grade = 3;
        student.address = "서울시 구로구";

        System.out.println("학번 : " + student.studentID);
        System.out.println("이름 : " + student.studentName);
        System.out.println("학년 : " + student.grade);
        System.out.println("주소 : " + student.address);

        System.out.println("-------------------------");
        System.out.println(student);  //인스턴스의 주소
    }
}
```

패키지 이름

```
학번 : 1001
이름 : 홍길동
학년 : 3
주소 : 서울시 구로구
-------------------------
classpart.Student@7d6f77cc
```

생성자(constructor)

❖ 생성자(Constructor)

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

  생략하여도 컴파일러가 자동으로 생성해 준다.

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

    public Person() {

    }

    public void showInfo() {
        System.out.println("이름 : " + name + ", 키 : " +
                height + ", 몸무게 : " + weight);
    }
}
```

생성자

생성자(constructor)

❖ 기본 생성자

```
public class PersonTest {

    public static void main(String[] args) {
        Person person = new Person();
        person.name = "손흥민";
        person.height = 183.2F;
        person.weight = 76.7F;

        person.showInfo();

        System.out.println(person);
    }
}
```

생성자

```
이름 : 손흥민, 키 : 183.2, 몸무게 : 76.7
constructor.Person@33833882
```

생성자(constructor)

❖ 매개변수가 있는 생성자

- 멤버 변수에 대한 값을 매개 변수로 받아서 멤버 변수 값을 초기화함

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

    public Person() {

    }

    public Person(String n) {
        name = n;
    }

    public void showInfo() {
        System.out.println("이름 : " + name + ", 키 : " +
                height + ", 몸무게 : " + weight);
    }
}
```

```java
public class PersonTest2 {

    public static void main(String[] args) {
        Person person = new Person("손흥민");
        //이름을 생성자에서 직접 지정함.
        person.height = 183.2F;
        person.weight = 76.7F;

        person.showInfo();
    }
}
```

생성자 오버로드

❖ 생성자 오버로드(overload)

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

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

    public Person() {

    }

    public Person(String n) {
        name = n;
    }

    public Person(String n, float h, float w) {
        name = n;
        height = h;
        weight = w;
    }

    public void showInfo() {
        System.out.println("이름 : " + name + ", 키 : " +
                height + ", 몸무게 : " + weight);
    }
}
```

생성자 오버로드

❖ 생성자 오버로드(overload)

```
public class PersonTest3 {

    public static void main(String[] args) {
        //기본 생성자로 생성
        Person son = new Person();
        son.name = "손흥민";
        son.height = 183.2F;
        son.weight = 76.7F;

        //매개변수가 있는 생성자
        Person chu = new Person("추신수", 180.3F, 90.0F);

        son.showInfo();
        chu.showInfo();
    }
}
```

```
이름 : 손흥민, 키 : 183.2, 몸무게 : 76.7
이름 : 추신수, 키 : 180.3, 몸무게 : 90.0
```

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

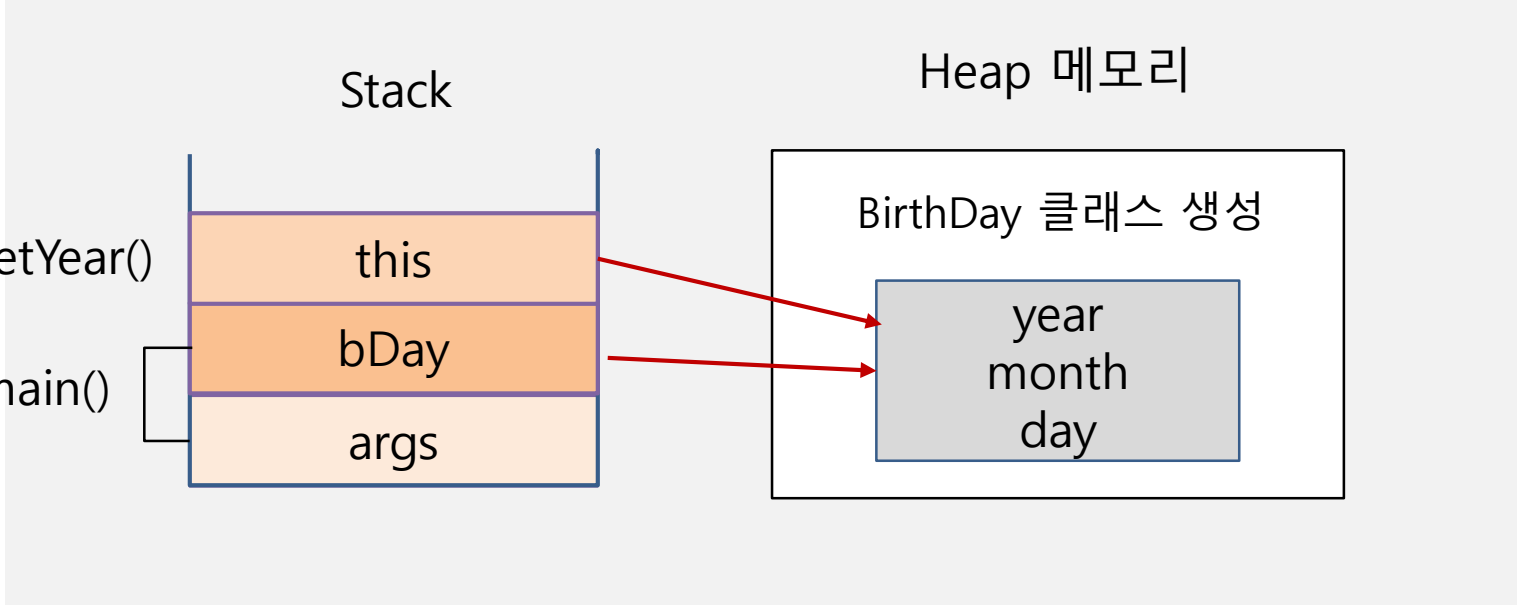
```
public class ThisTest {

    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

❖ 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

- **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

- 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

- **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

- **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

- **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]);
        }
```

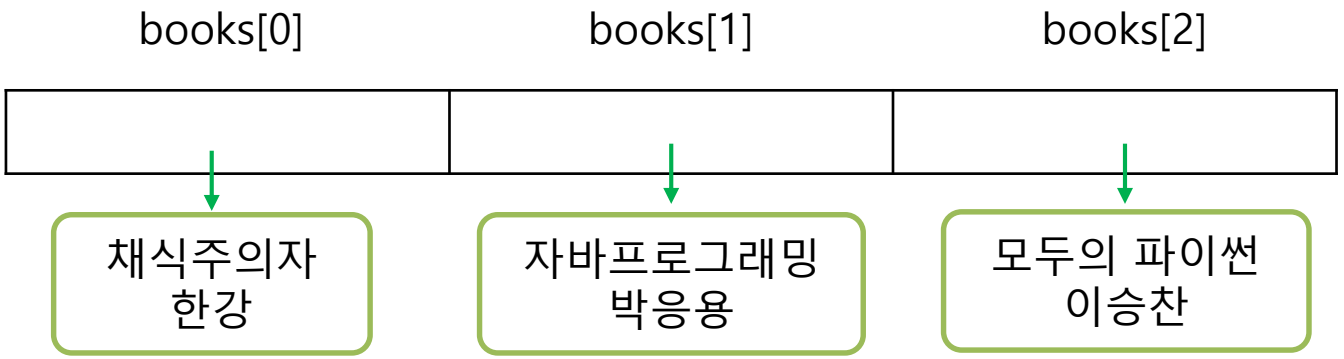
books[0]	books[1]	books[2]
null	null	null

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")
};
```

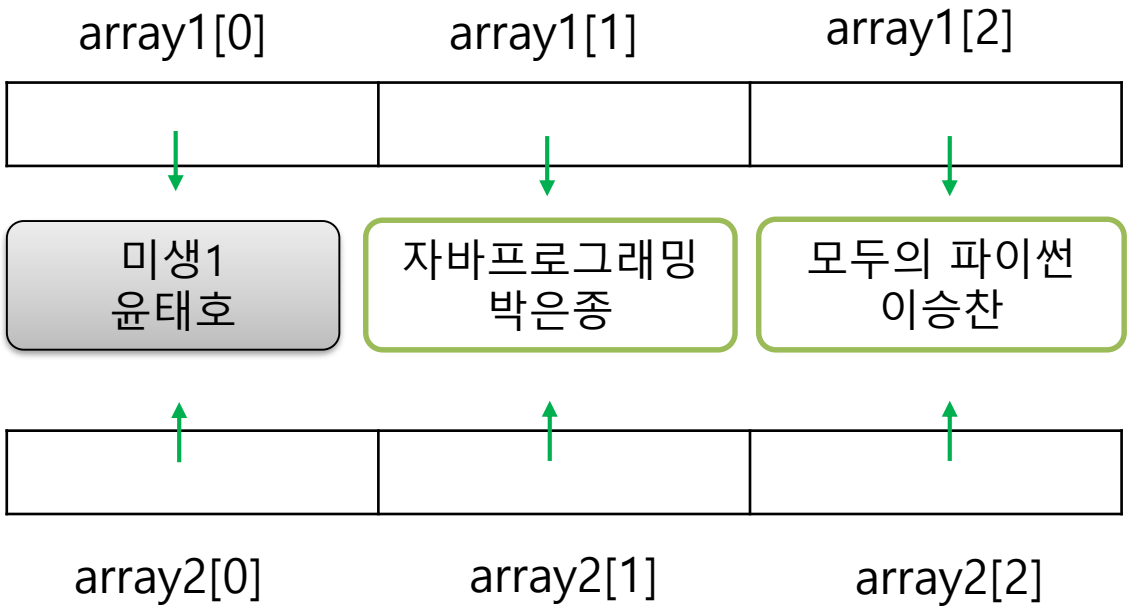
객체 배열 복사하기

- **객체 배열 복사하기**

```
public class ObjectCopy {

    public static void main(String[] args) {
        //객체 배열 생성 방법 3
        Book[] array1 = new Book[3];
        Book[] array2 = new Book[3];

        //Book 객체 생성
        array1[0] = new Book(100, "채식주의자", "한강");
        array1[1] = new Book(101, "자바프로그래밍 입문", "박은종");
        array1[2] = new Book(102, "모두의 파이썬", "이승찬");

        // array1을 array2에 복사
        for(int i=0; i<array1.length; i++) {
            array2[i] = array1[i];
        }

        System.out.println("========== array2 출력 ==========");
        for(int i=0; i<array1.length; i++) {
            array2[i].showBookInfo();
        }
    }
}
```

객체 배열 – 얕은 복사

- 객체 배열의 얕은 복사(shallow copy)



객체 배열 – 얕은 복사

- 객체 배열의 얕은 복사

```
public class ShallowCopy {

    public static void main(String[] args) {
        //객체 배열 생성 방법 3
        Book[] array1 = new Book[3];
        Book[] array2 = new Book[3];

        //Book 객체 생성
        array1[0] = new Book(100, "채식주의자", "한강");
        array1[1] = new Book(101, "자바프로그래밍 입문", "박은종");
        array1[2] = new Book(102, "모두의 파이썬", "이승찬");

        //array1의 첫번째 요소값 수정
        array1[0].setBookName("미생1");
        array1[0].setAuthor("윤태호");
```

객체 배열 – 얕은 복사

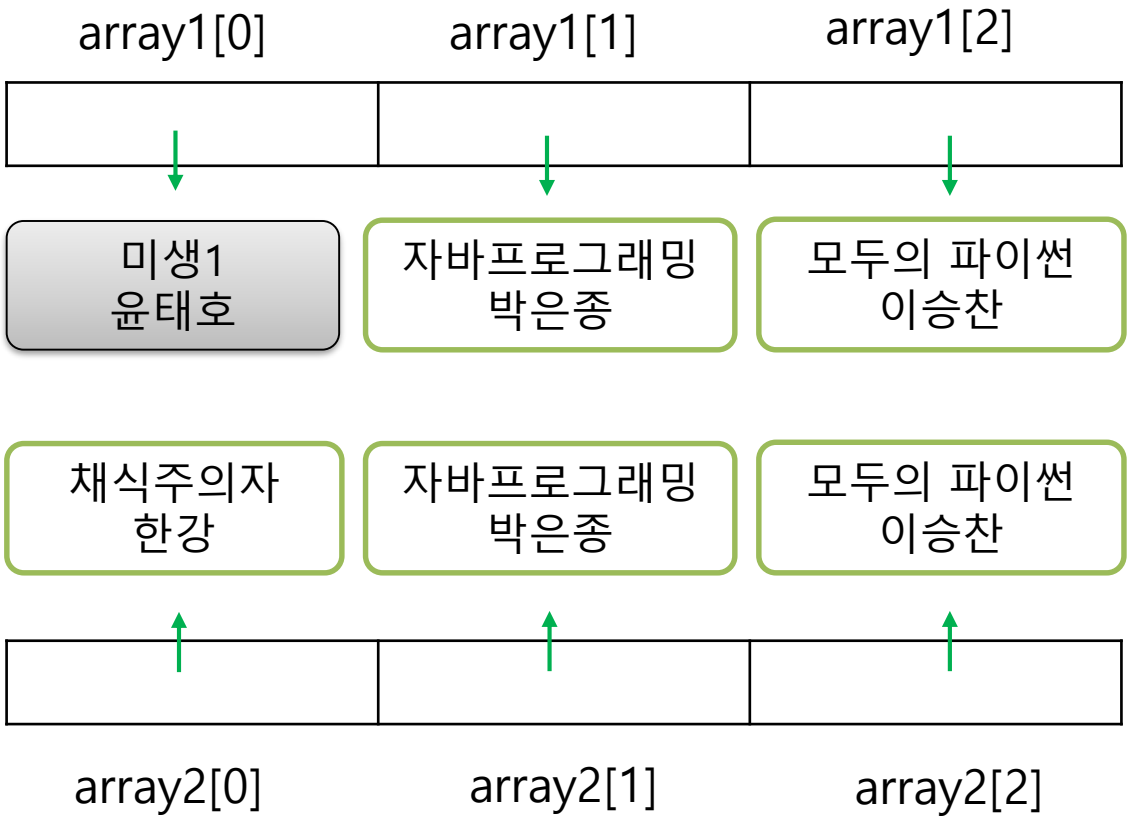
- 객체 배열의 얕은 복사

```
        //얕은 복사
        for(int i=0; i<array1.length; i++) {
            array2[i] = array1[i];
        }

        System.out.println("========== array1 출력 ==========");
        for(int i=0; i<array1.length; i++) {
            array1[i].showBookInfo();
        }

        System.out.println("========== array2 출력 ==========");
        for(int i=0; i<array2.length; i++) {
            array2[i].showBookInfo();
        }
    }
}
```

```
========== array1 출력 ==========
100: 미생1, 윤태호
101: 자바프로그래밍 입문, 박은종
102: 모두의 파이썬, 이승찬
========== array2 출력 ==========
100: 미생1, 윤태호
101: 자바프로그래밍 입문, 박은종
102: 모두의 파이썬, 이승찬
```

객체 배열 – 깊은 복사

- **객체 배열의 깊은 복사(deep copy)**



객체 배열 – 깊은 복사

- **객체 배열의 깊은 복사**

```
public class DeepCopy {

    public static void main(String[] args) {
        //객체 배열 생성 방법 3
        Book[] array1 = new Book[3];
        Book[] array2 = new Book[3];

        //Book 객체 생성
        array1[0] = new Book(100, "채식주의자", "한강");
        array1[1] = new Book(101, "자바프로그래밍 입문", "박은종");
        array1[2] = new Book(102, "모두의 파이썬", "이승찬");

        //기본 생성자로 array2 배열 인스턴스 생성
        array2[0] = new Book();
        array2[1] = new Book();
        array2[2] = new Book();

        //깊은 복사
        for(int i=0; i<array1.length; i++) {
            array2[i].setBookNumber(array1[i].getBookNumber());
            array2[i].setBookName(array1[i].getBookName());
            array2[i].setAuthor(array1[i].getAuthor());
        }
```

객체 배열 – 깊은 복사

- **객체 배열의 깊은 복사**

```
        //array1의 첫번째 요소값 수정
        array1[0].setBookName("미생1");
        array1[0].setAuthor("윤태호");

        System.out.println("========== array1 출력 ==========");
        for(int i=0; i<array1.length; i++) {
            array1[i].showBookInfo();
        }

        System.out.println("========== array2 출력 ==========");
        for(int i=0; i<array2.length; i++) {
            array2[i].showBookInfo();
        }
    }
}
```

```
========== array1 출력 ==========
100: 미생1, 윤태호
101: 자바프로그래밍 입문, 박은종
102: 모두의 파이썬, 이승찬
========== array2 출력 ==========
100: 채식주의자, 한강
101: 자바프로그래밍 입문, 박은종
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

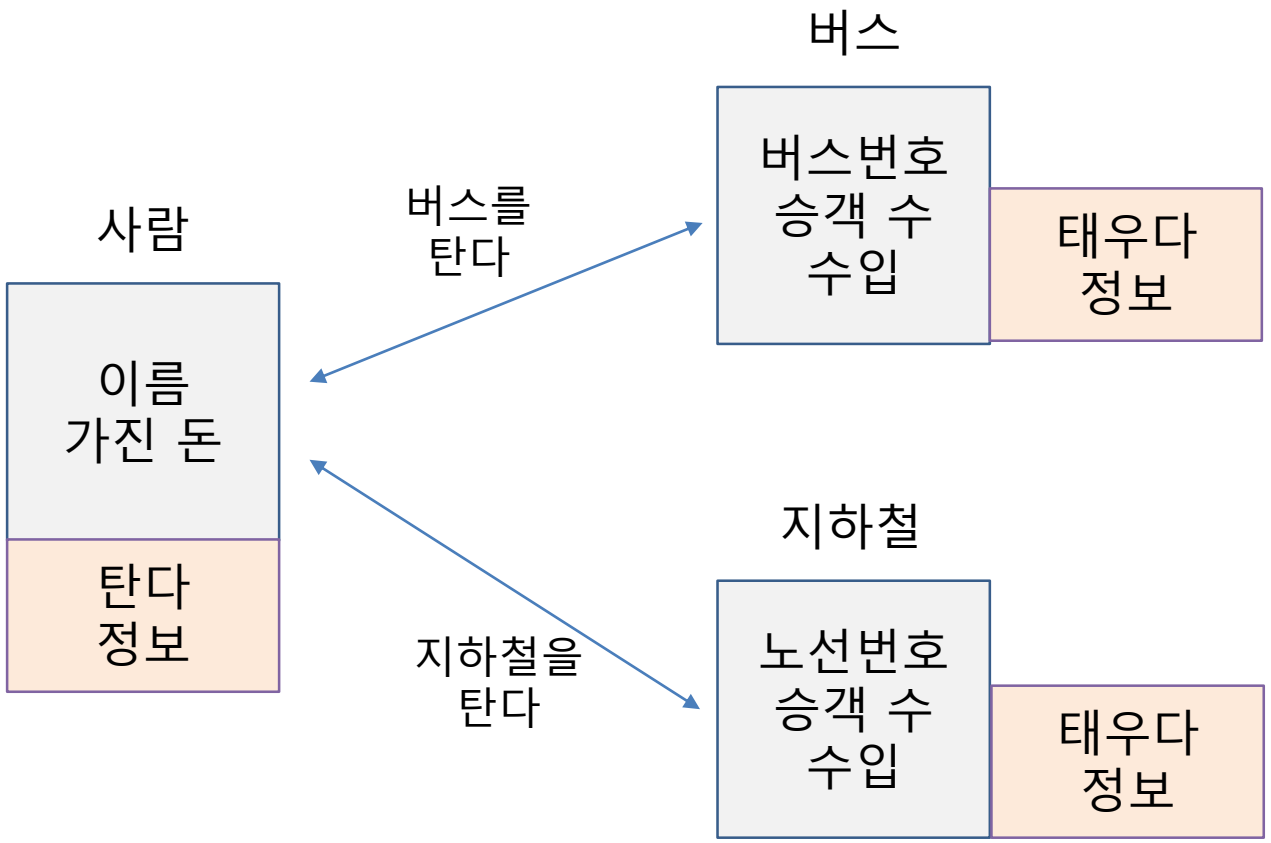
- Circle 테스트

```
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.
```

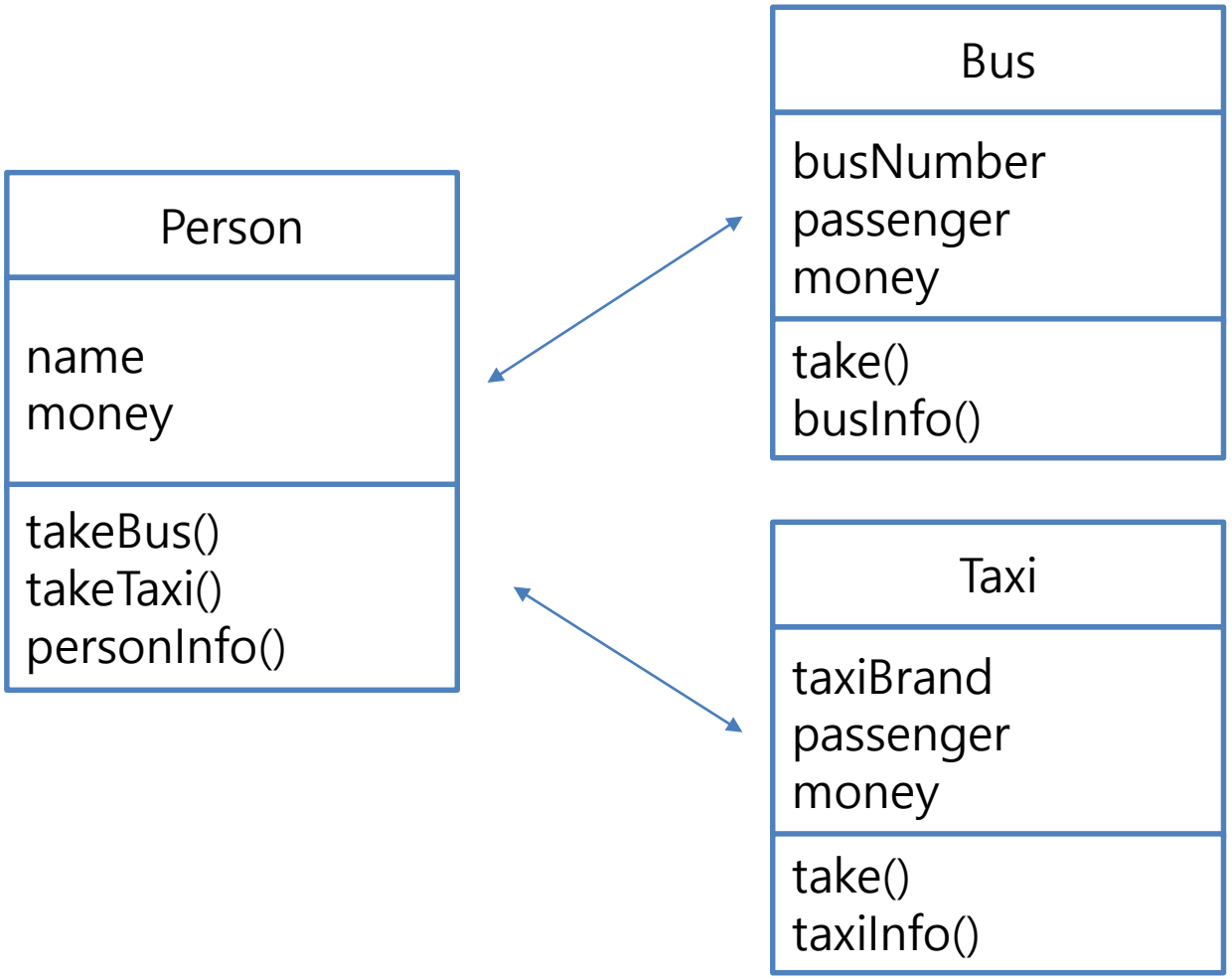
객체 간 협력

- 사람이 버스나 지하철을 타는 상황을 객체 지향으로 프로그래밍하기



객체 간 협력

- 학생, 버스, 지하철 클래스



객체 간 협력

- 버스 클래스

```
package transport;

public class Bus {
    private int busNumber;   //버스 번호
    private int passenger;   //승객수
    private int money;       //버스의 수입

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

    public void take(int money) {
        this.money += money;
        passenger++;
    }

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

객체 간 협력

- **Person 클래스**

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

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

    public void takeBus(Bus bus, int fee) {
        bus.take(fee);
        this.money -= fee;
    }

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

객체 간 협력

```java
public class TransportMain {

    public static void main(String[] args) {
        // Bus 객체 생성
        Bus bus471 = new Bus(471);
        int feeOfBus = 1500;  //버스 요금

        //승객 1명 태움
        bus471.take(feeOfBus);
        //승객 2명 태움
        bus471.take(feeOfBus);

        //버스 정보 출력
        bus471.busInfo();

        // Person 객체 1명 생성
        Person p1 = new Person("허미미", 10000);
        p1.takeBus(bus471, feeOfBus);
        p1.personInfo();

        // 사람 2명 생성
        Person p2 = new Person("이강인", 20000);
        p2.takeBus(bus471, feeOfBus);
        p2.personInfo();

    }
}
```

```
471번 버스의 수입은 3000원이고, 승객 수는 2명 입니다.
허미미님의 남은 돈은 8500원 입니다.
이강인님의 남은 돈은 18500원 입니다.
```

실습 문제 1 – 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
```