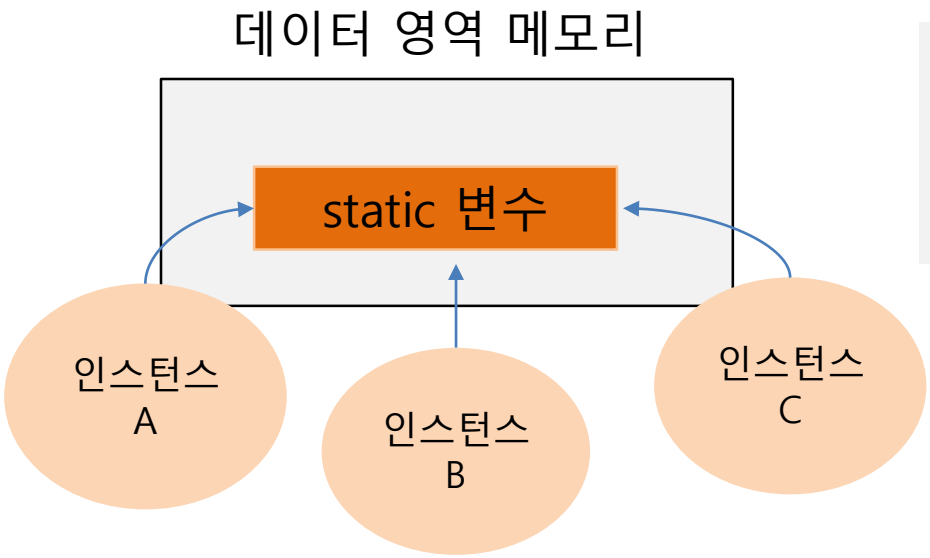
6장. 클래스와 객체2

static

static 변수

▪ static 변수의 정의와 사용 방법

- 다른 멤버변수처럼 인스턴스가 생성될 때마다 새로 생성되는 변수가 아니다.
- 프로그램이 실행되어 **메모리에 적재(load)**될때 메모리 공간이 할당된다.
- 여러 개의 인스턴스가 같은 **메모리의 값을 공유**하기 위해 사용



static 예약어

```
static int serialNum=1000;
```

static 변수

- 차량번호 자동 부여

```
public class Car {

    private static int serialNum = 1000; //정적 변수
    private int carNumber;

    public Car() {
        serialNum++;
        carNumber = serialNum;
    }

    public int getCarNumber() {
        return carNumber;
    }
}
```

static 변수

- 차량번호 자동 부여

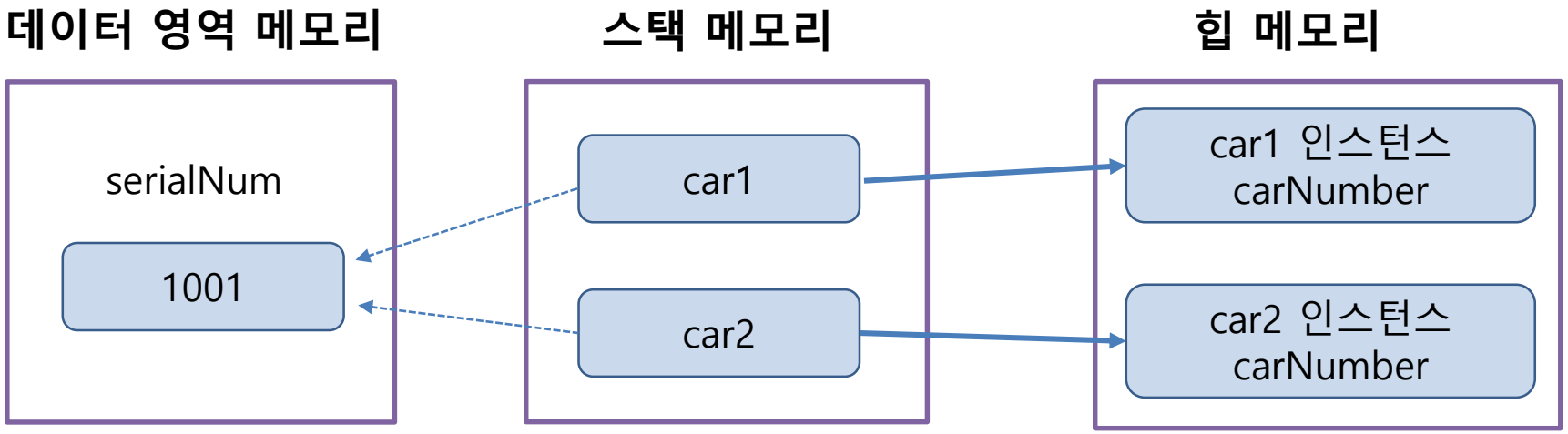
```
public class CarTest {

    public static void main(String[] args) {
        Car car1 = new Car();
        Car car2 = new Car();
        Car car3 = new Car();

        System.out.println("차량번호: " + car1.getCarNumber());
        System.out.println("차량번호: " + car2.getCarNumber());
        System.out.println("차량번호: " + car3.getCarNumber());
    }
}
```

```
차량번호: 1001
차량번호: 1002
차량번호: 1003
```

인스턴스와 참조변수

◆ 차량번호 자동 부여

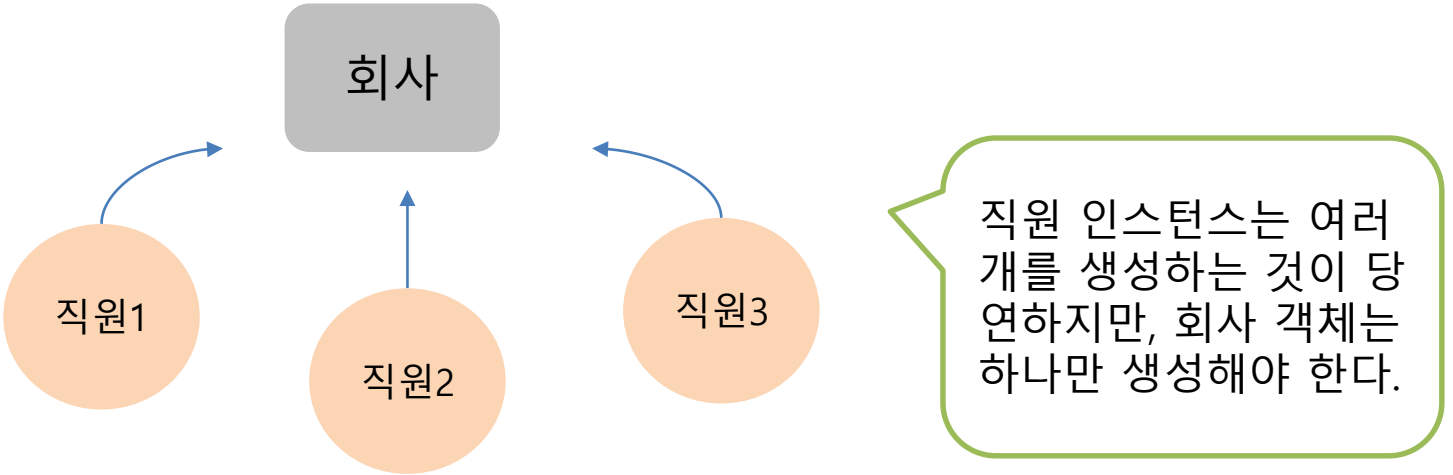
- 차가 생성될 때마다 차량번호가 증가해야 하는 경우
- 기준이 되는 값은 static 변수로 생성하여 유지 함.



static으로 선언한 serialNum 변수는 모든 인스턴스가 공유한다 . 즉 두 개의 참조변수가 동일한 변수의 메모리를 가리키고 있다.

static 응용 : 싱글톤 패턴

- **Single 패턴이란?**
  - 객체지향 프로그램에서 인스턴스를 단 하나만 생성하는 디자인 패턴
  - static을 응용하여 프로그램 전반에서 사용하는 인스턴스를 하나만 구현하는 방식



static 응용 : 싱글톤 패턴

- **Singleton 패턴으로 회사 클래스 구현하기**

  1. 생성자를 private으로 만들기

  2. static으로 유일한 인스턴스 생성하기 – getInstance() 메서드

```
public class Company {
    private static Company instance; //instance 객체 선언

    private Company() {}; //외부에서 생성자를 호출 불가

    public static Company getInstance() { //Company로 직접 접근 가능
        if(instance == null) {
            instance = new Company();
        }
        return instance;
    }
}
```

static 응용 : 싱글톤 패턴

- **Singleton 패턴으로 회사 클래스 구현하기**

```
public class CompanyTest {

    public static void main(String[] args) {
        Company myCompany1 = Company.getInstance();

        Company myCompany2 = Company.getInstance();

        //두 변수가 같은 주소인지 확인
        System.out.println(myCompany1==myCompany2);

        System.out.println(myCompany1);
        System.out.println(myCompany2);
    }
}
```

```
true
singleton.Company@7d6f77cc
singleton.Company@7d6f77cc
```

static 응용 : 싱글톤 패턴

자동차 공장이 1개 있고, 이 공장에서 생산되는 자동차는 제작될 때마다 고유 번호가 부여된다. 자동차번호가 1001부터 시작되어 1002, 1003으로 붙도록 자동차 공장 클래스, 자동차 클래스를 만들어 본다.

```
public class CarTest {

    public static void main(String[] args) {
        //자동차 회사 객체 생성
        CarFactory factory = CarFactory.getInstance();

        //자동차 객체 생성
        Car car1 = factory.createCar();
        Car car2 = factory.createCar();
        Car car3 = factory.createCar();

        System.out.println(car1.getCarNumber());
        System.out.println(car2.getCarNumber());
        System.out.println(car3.getCarNumber());
    }
}
```

```
신차 번호: 1001
신차 번호: 1002
신차 번호: 1003
```

static 응용 : 싱글톤 패턴

- Car 클래스

```
public class Car {

    private static int serialNum = 1000; //정적 변수
    private int carNumber;

    public Car() {
        serialNum++;
        carNumber = serialNum;
    }

    public int getCarNumber() {
        return carNumber;
    }
}
```

static 응용 : 싱글톤 패턴

- **CarFactory 클래스**

```
public class CarFactory {
    private static CarFactory instance = new CarFactory();

    private CarFactory() {}

    public static CarFactory getInstance() {
        if(instance==null) {
            instance = new CarFactory();
        }
        return instance;
    }

    public Car createCar() {  //자동차 생성 메서드
        Car car = new Car();
        return car;
    }
}
```

메서드(멤버 함수)

- static 이 있는 메서드

```java
package methods;

public class VoidMethods {

    //반환 자료형이 없는 함수
    public static void sayHello() {
        System.out.println("Hello~");
    }

    //메서드 오버로딩(이름이 같고 매개변수 형태가 다른 것)
    public static void sayHello(String name) {
        System.out.println("Hello~ " + name);
    }

    public static void main(String[] args) {
        //메서드 호출
        sayHello();

        sayHello("유빈");
        sayHello("정후");
    }
}
```

인스턴스형 메서드 만들기

- 외부 클래스에서 메서드 정의하기

```
class Hello{
    //메서드 오버로딩(중복)
    public void sayHello() {
        System.out.println("Hello~");
    }

    public void sayHello(String name) {
        System.out.println("Hello~ " + name);
    }
}
```

인스턴스형 메서드 만들기

- 외부 클래스에서 메서드 정의하기

```
public class UseHello {

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

        //인스턴스형 메서드 사용
        greeting.sayHello();
        greeting.sayHello("Elsa");
        greeting.sayHello("기용샘");
    }
}
```

```
Hello~
Hello~ Elsa
Hello~ 기용샘
```

외부 클래스에서 메서드 만들기

- 외부 클래스에서 메서드 정의하기

```
public class Calculator {

    public int add(int n1, int n2) { //더하기
        return n1 + n2;
    }

    public int sub(int n1, int n2) { //빼기
        return n1 - n2;
    }

    public int mul(int n1, int n2) { //곱하기
        return n1 * n2;
    }
    public double div(int n1, int n2) { //나누기
        //예외 처리
        if (n2 == 0) {
            System.out.println("0으로 나눌 수 없습니다.");
            return Double.NaN; //NaN(Not a Number) 반환
        }
        return (double)n1 / n2;
    }
}
```

외부 클래스에서 메서드 만들기

- CalculatorTest 클래스에서 Calculator의 메서드 사용하기

```
public class CalculatorTest {

    public static void main(String[] args) {
        // 계산기 객체 생성
        Calculator calc = new Calculator();

        // 인스턴스 메서드 호출
        int add = calc.add(10, 20);
        int sub = calc.sub(10, 20);
        int mul = calc.mul(10, 20);
        double div = calc.div(10, 0);

        System.out.println("두 수의 합: " + add);
        System.out.println("두 수의 차: " + sub);
        System.out.println("두 수의 곱: " + mul);
        System.out.println("두 수의 나누기: " + div);
    }
}
```

```
0으로 나눌 수 없습니다.
두 수의 합: 30
두 수의 차: -10
두 수의 곱: 200
두 수의 나누기: NaN
```

회원 로그인 서비스 클래스

❖ 로그인/로그아웃을 서비스하는 클래스

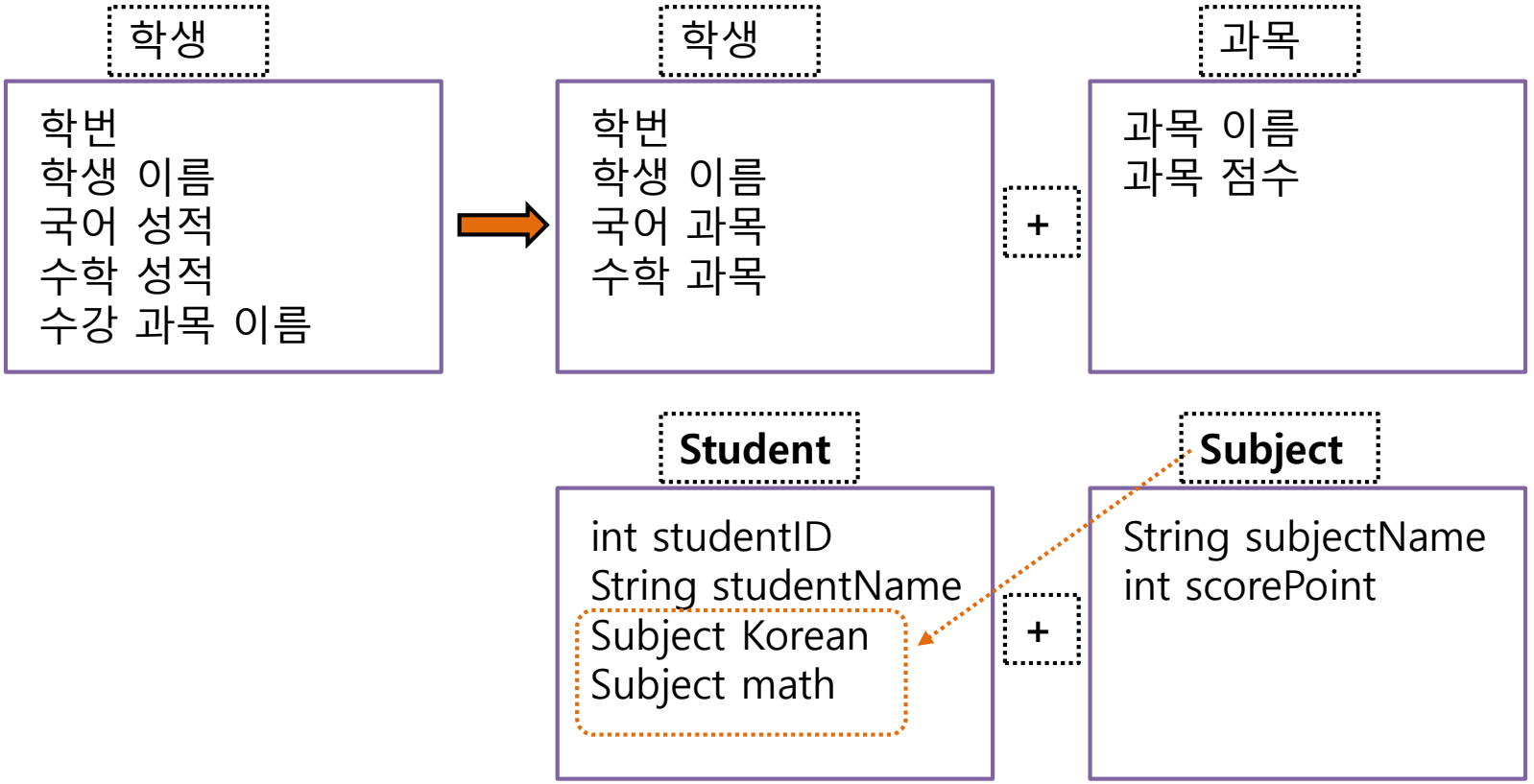
```
public class MemberService {

    //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.");
        }
    }
}
```

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

        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(   //학생 정보 출력
                        "학번: " + studentId +
                        "\n이름: " + studentName +
                        "\n" + subjects[i].getSubjectName() +
                        "점수: " + subjects[i].getScorePoint());
                System.out.println("------------------------");
            }
        }
        //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);
        lee.displayInfo();

        Student shin = new Student(1002, "신유빈");
        //과목 확장
        shin.addSubject("국어", 92);
        shin.addSubject("수학", 80);
        shin.addSubject("과학", 79);
        shin.displayInfo();
    }
}
```

```
학번: 1001
이름: 이정후
국어점수: 90
------------------------
학번: 1001
이름: 이정후
수학점수: 85
------------------------
학번: 1001
이름: 이정후
과학점수: 80
------------------------
평균 점수: 85.0점
========================
```

```
학번: 1002
이름: 신유빈
국어점수: 92
------------------------
학번: 1002
이름: 신유빈
수학점수: 80
------------------------
학번: 1002
이름: 신유빈
과학점수: 79
------------------------
평균 점수: 83.7점
========================
```

실습 문제 1 – 싱글톤 패턴

카드 회사에서 카드를 발급할 때마다 카드 고유 번호를 부여해줍니다.
카드 클래스를 만들고, 카드 회사 클래스 CardCompany를 싱글톤 패턴을 사용하여 구현해 보세요.

☞ 실행 결과

```
카드번호: 1001
카드번호: 1002
카드번호: 1003
```