

6장. 클래스와 객체



Class & Object



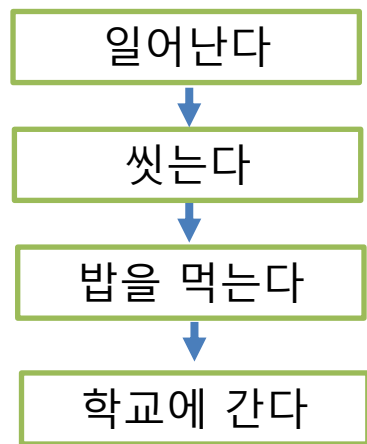
객체 지향 프로그래밍

■ 객체(Object)란?

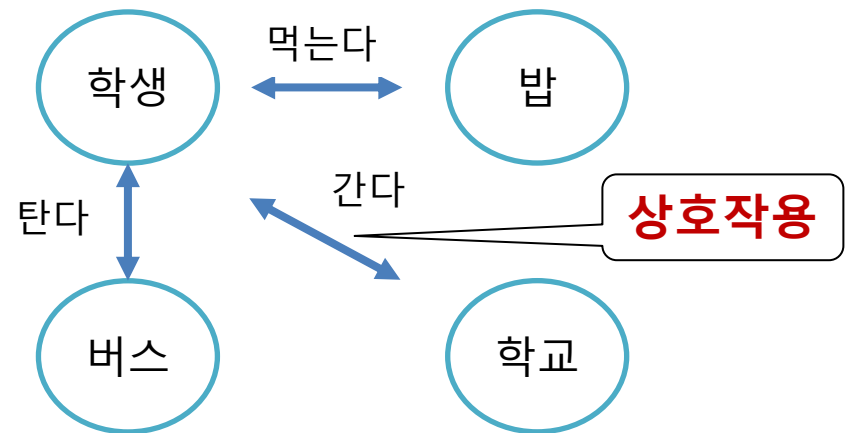
- 의사나 행위가 미치는 대상 -> 사전적 의미
- 구체적, 추상적 데이터 단위 (구체적-책상, 추상적-회사)

- 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 **동일**해야 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

Student student = **new** Student();

클래스

인스턴스

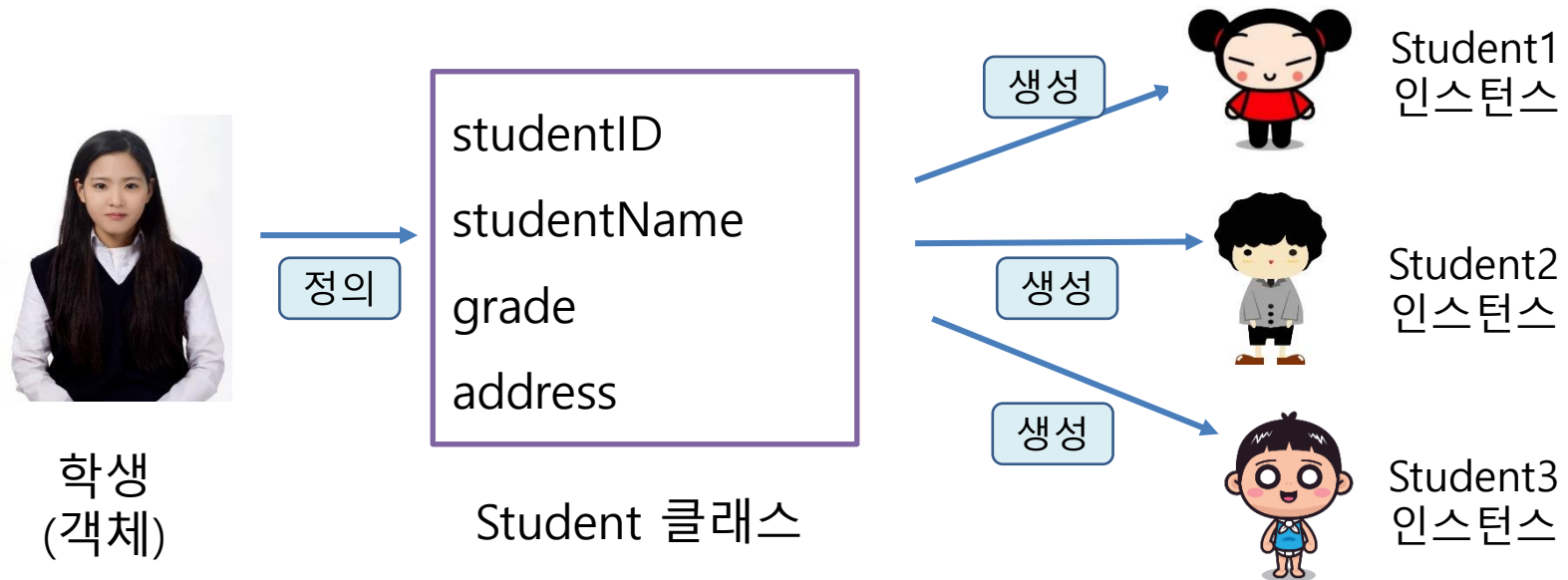
```
public class StudentTest {  
  
    public static void main(String[] args) {  
        // Student 클래스 사용하기  
        Student student = new Student(); //student 객체 생성  
  
        student.studentID = 1001;  
        student.studentName = "홍길동";  
        student.grade = 3;  
        student.address = "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

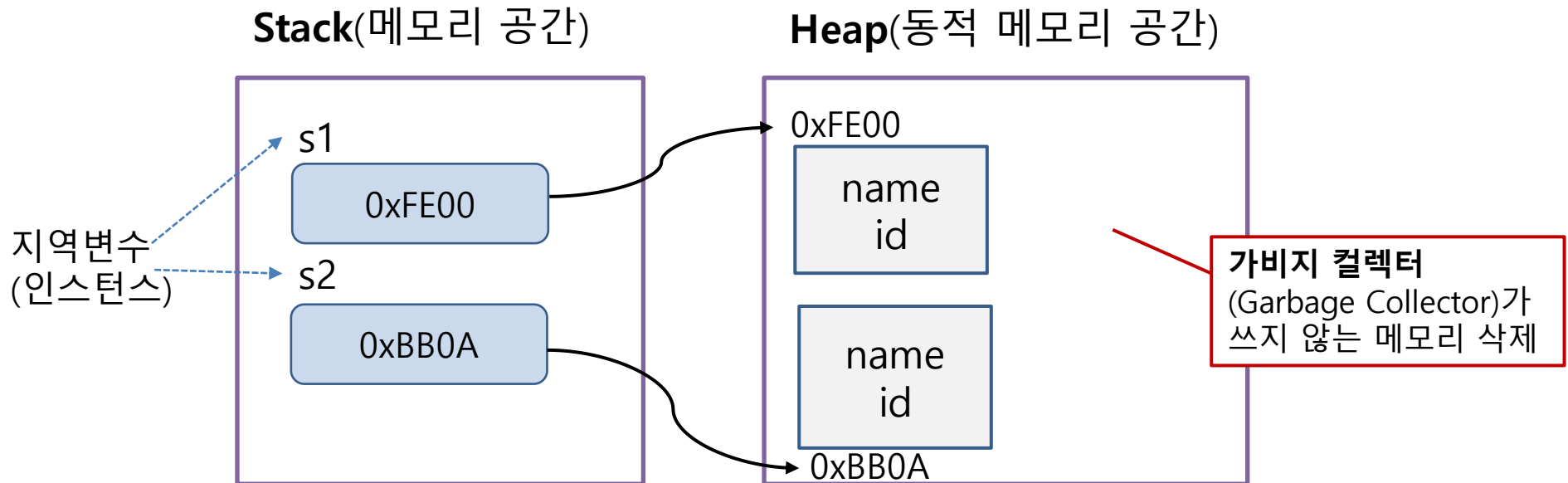
학번 : 1002
이름 : 이순신
학년 : 2
주소 : 경기도 평택시



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.)



패키지(package)

■ 패키지란?

- 클래스 파일의 묶음이다.
- 패키지를 만들면 프로젝트 하위에 물리적으로 디렉터리가 생성된다.
- 클래스의 실제 이름은 패키지이름.클래스이름 이다. (예:classpart.Student)

```
package classpart; ← 패키지 이름

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
    }
}
```

학번 : 1001
이름 : 홍길동
학년 : 3
주소 : 서울시 구로구

classpart.Student@7d6f77cc



메서드 정의

▪ Employee 클래스에 메서드 정의하기

- 메인 메소드(함수)가 있는 클래스에서 실행 사용할 수 있음

```
class Employee{  
    int companyId; //사번  
    String name;    //이름  
    int age;        //나이  
    String gender;  //성별  
  
    public void showInfo() {  
        System.out.println("사번 : " + companyId);  
        System.out.println("이름 : " + name);  
        System.out.println("나이 : " + age);  
        System.out.println("성별 : " + gender);  
    }  
}
```

showInfo() 정의



메서드 사용

▪ EmployeeTest 클래스에서 메서드 사용하기

```
public class EmployeeTest {  
    public static void main(String[] args) {  
        Employee employeeJang = new Employee();  
        Employee employeeAhn = new Employee();  
  
        employeeJang.companyId = 10001;  
        employeeJang.name = "장그래";  
        employeeJang.age = 27;  
        employeeJang.gender = "남";  
  
        employeeJang.showInfo();  
  
        employeeAhn.companyId = 10002;  
        employeeAhn.name = "안영이";  
        employeeAhn.age = 28;  
        employeeAhn.gender = "여";  
  
        employeeAhn.showInfo();  
    }  
}
```

showInfo() 호출

사번 : 10001
이름 : 장그래
나이 : 27
성별 : 남
사번 : 10002
이름 : 안영이
나이 : 28
성별 : 여



인스턴스형 메서드 만들기

- 인스턴스 메서드 – 외부 클래스에서 메서드 정의하기

```
package sayhello;

public class Hello {
    public void sayHello() {
        System.out.println("Hello~");
    }

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

```
package sayhello;

public class UseHello {

    public static void main(String[] args) {
        // Hello 클래스의 say 객체(인스턴스) 생성
        Hello say = new Hello();
        say.sayHello();
        say.sayHello("수영");
    }
}
```



외부 클래스에서 메서드 만들기

- 외부 파일(클래스)에서 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 int div(int n1, int n2) { //나누기  
        return n1 / n2;  
    }  
}
```



외부 클래스에서 메서드 만들기

- CalculatorTest 클래스에서 Calculator의 메서드 사용하기

파일이름 : CalculatorTest.java

```
Calculator calc = new Calculator();  
int num1 = 10, num2 = 2;  
  
int add = calc.add(num1, num2);  
int sub = calc.sub(num1, num2);  
int mul = calc.mul(num1, num2);  
int div = calc.div(num1, num2);  
  
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);
```

두 수의 합	: 12
두 수의 차	: 8
두 수의 곱	: 20
두 수의 나누기	: 5



인스턴스형 메서드 만들기

- 인스턴스형 메서드 만들기

인스턴스 메서드 – new 객체를 생성하여 사용한다.

static 메서드 – new 객체를 생성하지 않고 바로 사용할 수 있다.

```
class Gugudan{  
    public void gugudan(int dan) {  
        for(int i=1; i<10; i++) {  
            System.out.println(dan + "x" + i + "=" + (dan*i));  
        }  
    }  
}  
  
public class GugudanTest {  
    public static void main(String[] args) {  
        Gugudan gugu = new Gugudan(); //gugu 인스턴스변수 생성  
        gugu.gugudan(4);  
    }  
}
```

인스턴스 메서드

4x1=4
4x2=8
4x3=12
4x4=16
4x5=20
4x6=24
4x7=28
4x8=32
4x9=36



생성자(constructor)

기본 생성자

- 매개변수가 없는 생성자를 기본 생성자라 하며, 생략할 수 있다.
- 기본 생성자는 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);  
    }  
}
```

기본생성자



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

```
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);  
    }  
}
```

```
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 만들 수 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



생성자(constructor)

멤버변수가 정수인 생성자

- 생성자는 클래스를 생성할 때만 호출한다.
- 생성자 이름은 클래스 이름과 같고, 생성자는 반환값(return)이 없다.

```
public class Number {  
    int x;  
  
    public Number() { //생성자  
        x = 4;  
    }  
  
    public static void main(String[] args) {  
        Number myObj = new Number();  
        System.out.println(myObj.x);  
    }  
}
```



생성자(constructor)

매개변수가 있는 생성자

```
public class Number2 {  
    int x;  
  
    public Number2(int y) {  
        x = y;  
    }  
  
    public static void main(String[] args) {  
        Number2 myObj = new Number2(4);  
        System.out.println(myObj.x);  
    }  
}
```



❖ 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



▪ 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  
    }  
}
```



- **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;  
}
```



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

```
Account account1 = new Account();  
//account.ano = "100-1000";  
//private 멤버는 접근 불가  
  
//get(), set() 메서드로 접근 가능  
account1.setAno("100-1001");  
account1.setOwner("성춘향");  
account1.setBalance(20000);  
  
System.out.println("계좌번호1: " + account1.getAno());  
System.out.println("계좌주1: " + account1.getOwner());  
System.out.println("잔액1: " + account1.getBalance());
```



- 생성자 멤버를 **this**로 초기화 하기

매개변수 이름과 **this** 멤버 이름이 같아야 한다.

```
public class Account {  
    private String ano;    //계좌 번호  
    private String owner;  //계좌주  
    private int balance;   //잔액  
  
    public Account() {  
  
    }  
  
    public Account(String ano, String owner, int balance) {  
        this.ano = ano;  
        this.owner = owner;  
        this.balance = balance;  
    }  
}
```



■ 생성자 멤버를 this로 초기화 하기

```
Account account1 = new Account();  
//account.ano = "100-1000";  
//private 멤버는 접근 불가  
  
Account account2 = new Account("100-1002", "이몽룡", 50000);  
  
//get(), set() 메서드로 접근 가능  
account1.setAno("100-1001");  
account1.setOwner("성춘향");  
account1.setBalance(20000);  
  
System.out.println("계좌번호1: " + account1.getAno());  
System.out.println("계좌주1: " + account1.getOwner());  
System.out.println("잔액1: " + account1.getBalance());  
  
System.out.println("계좌번호2: " + account2.getAno());  
System.out.println("계좌주2: " + account2.getOwner());  
System.out.println("잔액2: " + account2.getBalance());
```



■ MyDate 클래스 – private 접근 제어자 사용

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

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

public 접근가능

```
public static void main(String[] args) {
    MyDate date = new MyDate();
    date.setYear(2019);
    date.setMonth(2);
    date.setDay(29);
}
```

날짜 오류입니다.



■ 윤년인 경우 코드

```
public void setDay(int day) {  
    if(month==2) {  
        if(day < 1 || day > 28) {  
            if(day==29) {  
                if(year % 4 == 0 && year % 100 != 0 || year % 400 == 0) {  
                    System.out.println("윤년입니다.");  
                }else {  
                    System.out.println("날짜 오류입니다.");  
                }  
            }else {  
                System.out.println("날짜 오류입니다.");  
            }  
        }else {  
            this.day = day;  
        }  
    }  
    this.day = day;  
}
```



▪ MyDateTest 클래스 - 날짜 테스트

```
package hiding;

public class MyDateTest {

    public static void main(String[] args) {
        MyDate date = new MyDate();

        //자료 입력
        date.setYear(2020);
        date.setMonth(2);
        date.setDay(29);
    }
}
```

윤년입니다.



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);
    }
}
```

```
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
    }
}
```

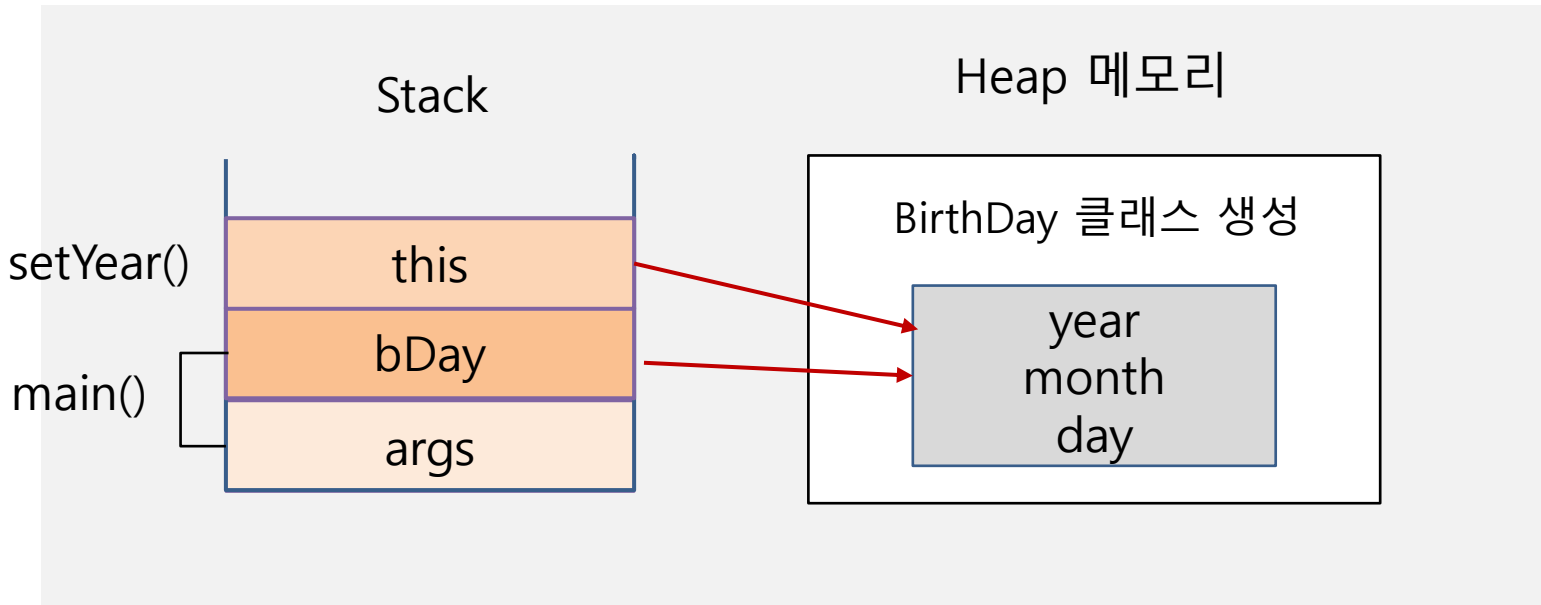
클래스이름@메모리 주소

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

