Java 5/31

Day13

지난주

5/24~5/28

Day9

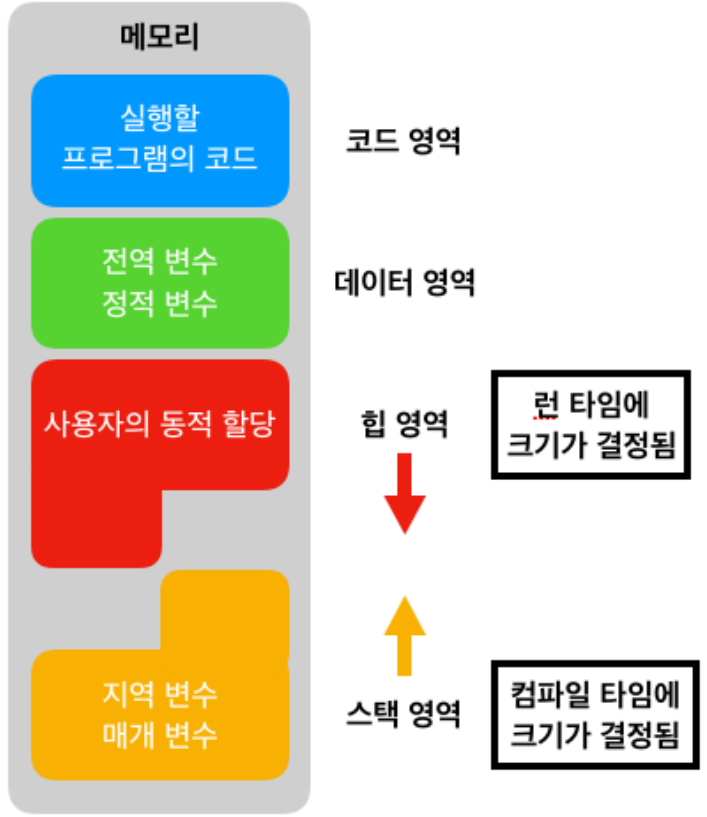
- 메서드 오버라이딩
  - 가상메서드
  - 어노테이션
- 다형성
- 업/다운캐스팅
- instanceOf
- 추상클래스 abstract
- 인터페이스 interface

클래스 선언과 사용

- 자바 클래스 생성
- Main class 와 일반 class
  - Main.java, Student.java
- 패키지가 다른 경우
  - import
  - full package name

객체지향 프로그래밍

- 현실 세계의 요구사항 -> 객체로 분석
  - 객체: 속성, 기능(메서드)
- 객체 -> 코드화
  - 클래스로 정의
    - 멤버 필드
    - 멤버 메서드
- 클래스를 인스턴스로 생성해 사용
  - new 연산자
  - 식별자 -> 객체
- 참조형
  - 기본형 제외한 나머자 (스트링 리터럴, 배열, 클래스 객체)



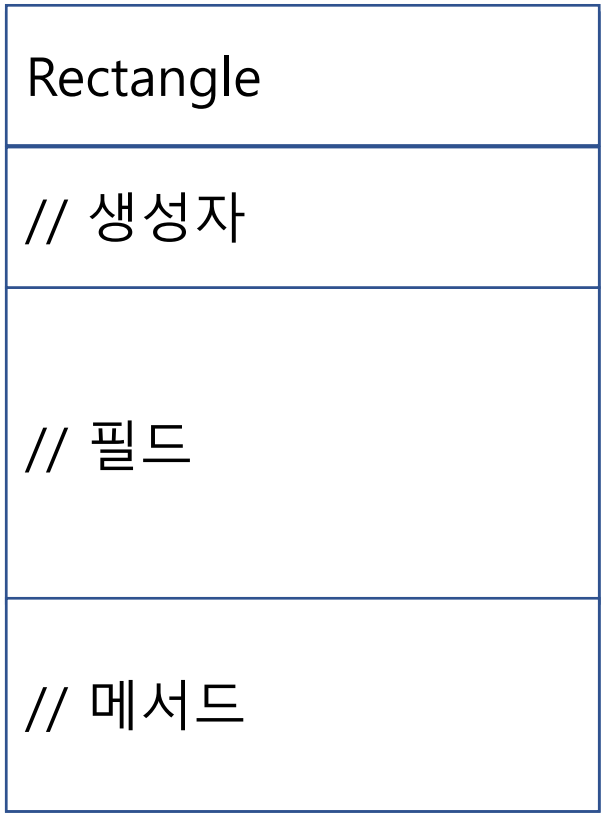
https://jinshine.github.io/2018/05/17컴퓨터%20기초/메모리구조/

생성자

- 특별한 메소드
  - class 이름과 같다.
- return 타입 없다
- 매개변수 갖는다 : 생성자 오버로딩

생성자

- 사각형은 이름 그리고 크기 값인 폭, 너비를 가진다.
- 사각형 클래스를
  - 기본 생성자
  - 기본 이름
  - 기본 폭, 너비



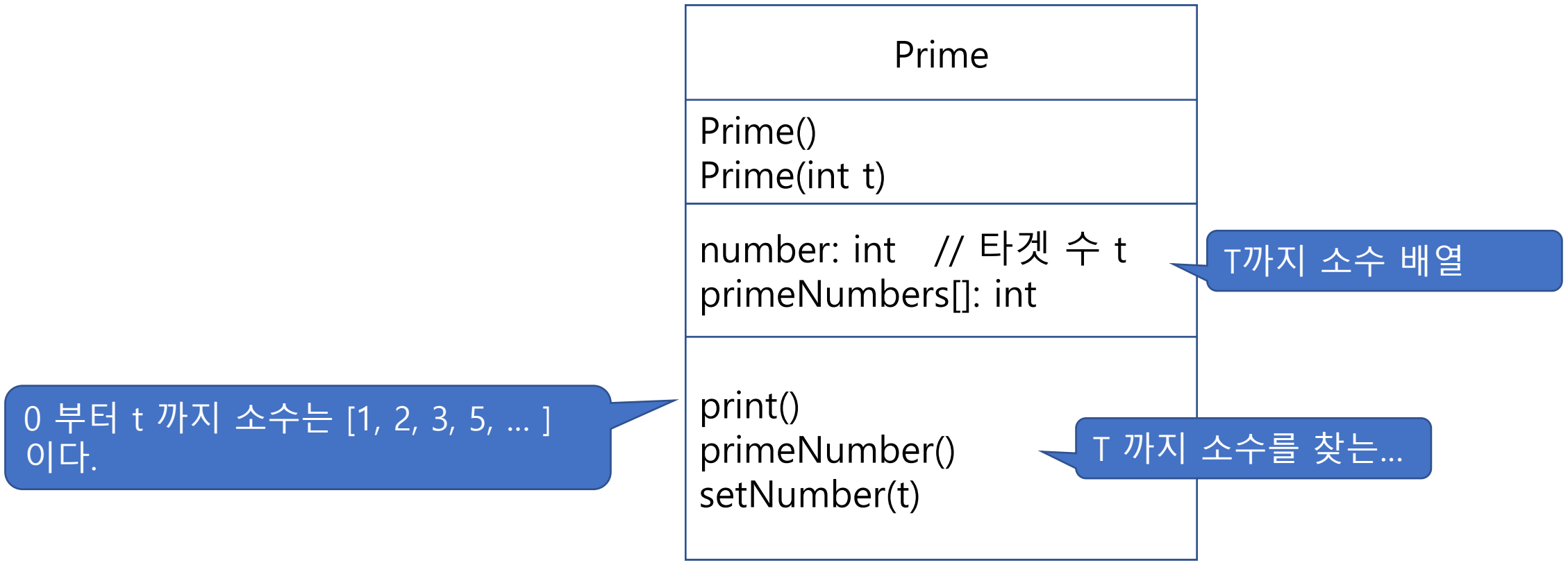
사각형 클래스 (Rectangle)
- 생성자
  - Rectangle()
  - Rectangle(name)
  - Rectangle(w, h)
  - Rectangle(name, w, h)

- 필드
  - name
  - width
  - height

실습: 소수 계산

Day6 실습

- **소수 를 계산하는 클래스를 선언하고 대상 수까지의 소수를 계산 해 보자.**
  - **생성자 입력된 t 값까지 소수 계산**



Day10

- 리뷰 실습: Prime number class
- This p170
- 유효범위 p191

- 정보은닉 p162
  - 접근 제한자 (한정자, Modifier)
  - https://wikidocs.net/232
- 상속
  - https://opentutorials.org/course/1223/6060
  - https://wikidocs.net/280
- 객체지향 언어 특성
  - 추상화 https://codevang.tistory.com/78
  - 캡슐화 https://codevang.tistory.com/79
  - 상속 https://codevang.tistory.com/80
  - 다형성 https://codevang.tistory.com/81

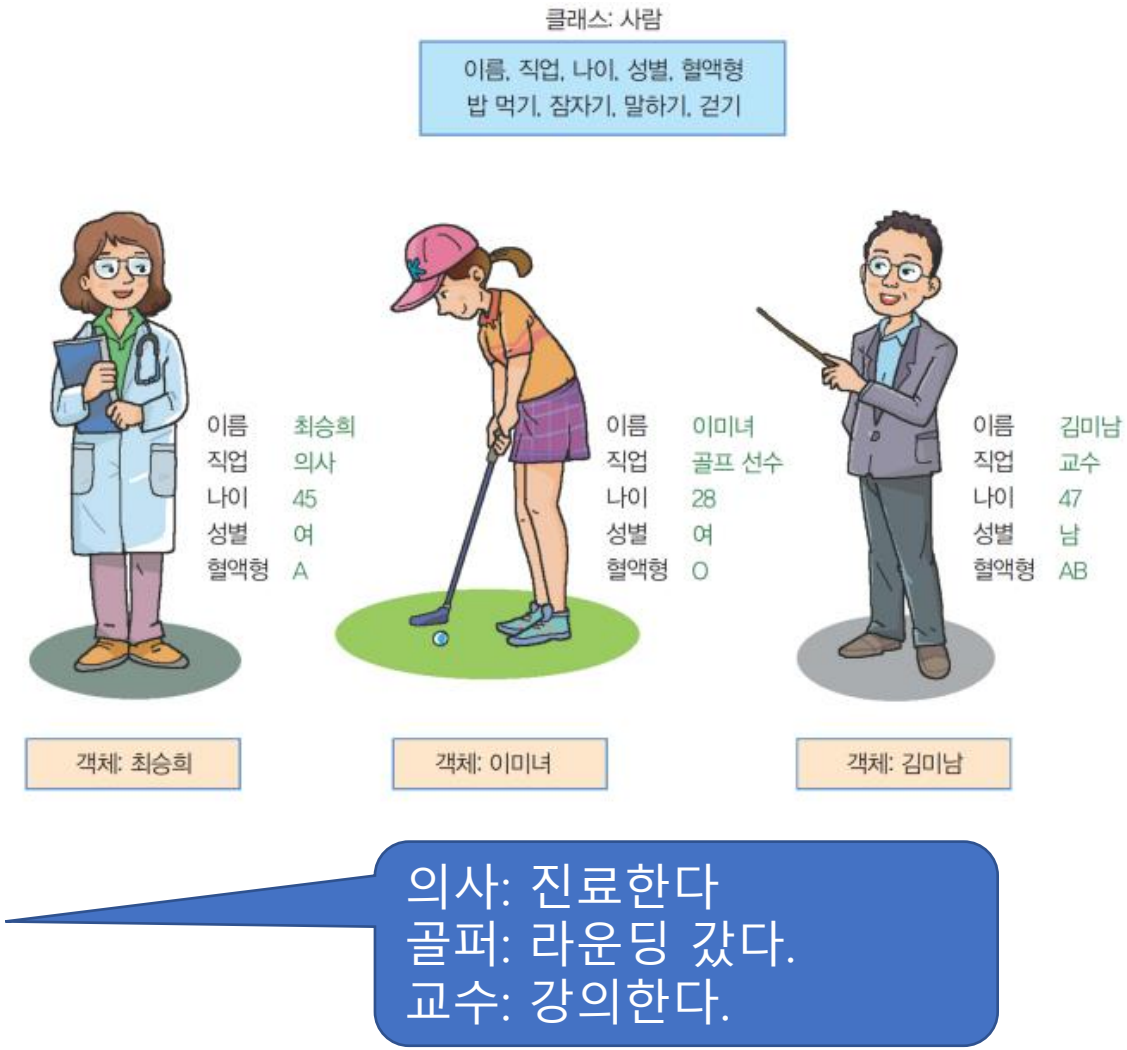
실습: 클래스 생성자, this, this() 이용

- 사람을 표현하는 Person 클래스
  - this, this() 생성자 사용.

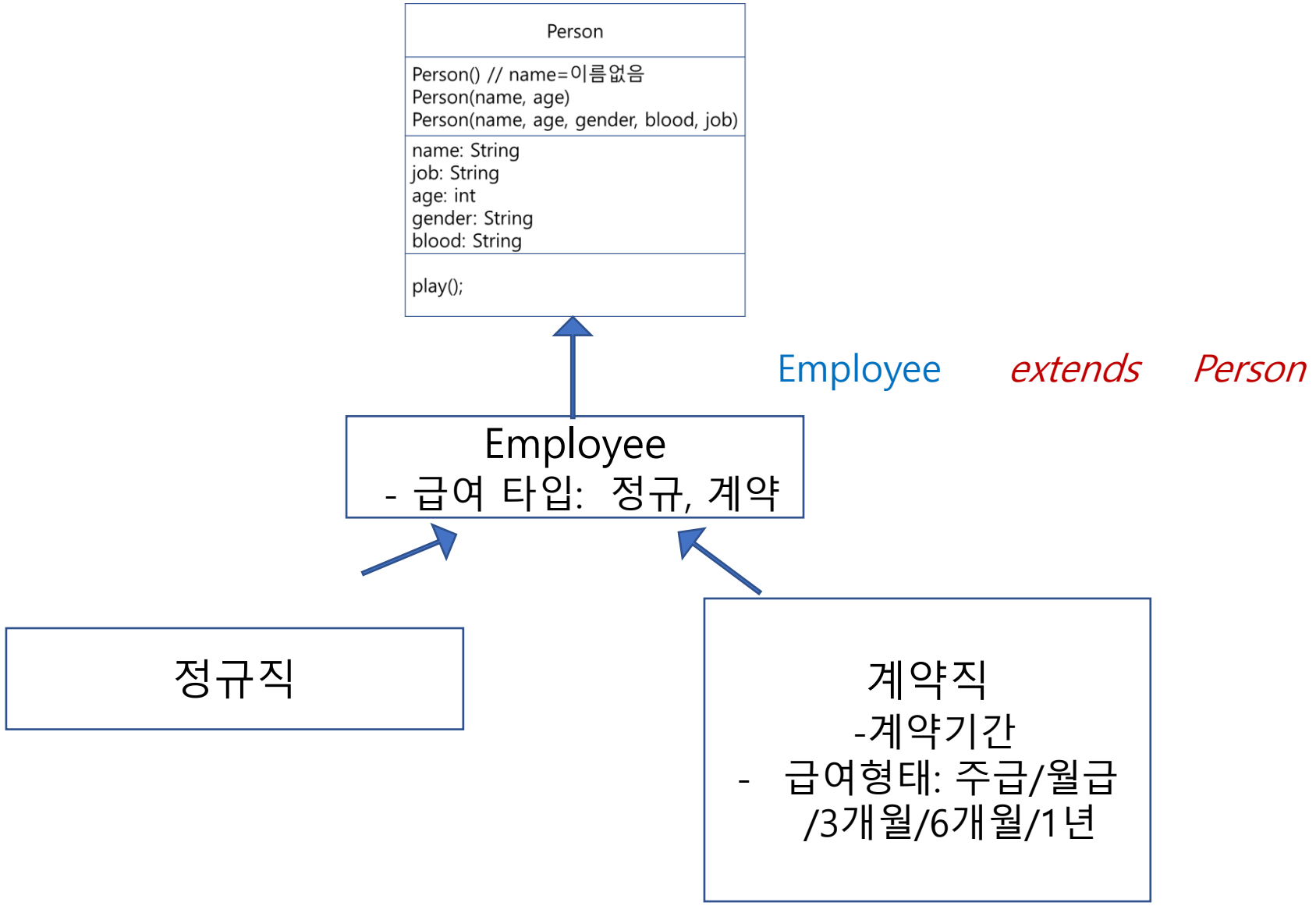
Person

Person() // name=이름없음
Person(name, age)
Person(name, age, gender, blood, job)

name: String job: String age: int gender: String blood: String
play();



상속


Person
Person() // name=이름없음
Person(name, age)
Person(name, age, gender, blood, job)
name: String
job: String
age: int
gender: String
blood: String
play();
Employee extends Person
Employee
- 급여 타입: 정규, 계약
정규직
계약직
-계약기간
- 급여형태: 주급/월급
/3개월/6개월/1년

Day11

- 메서드 오버라이딩
  - 가상메서드
  - 어노테이션
- 다형성
- 업/다운캐스팅
- instanceOf
- 추상클래스 abstract
- 인터페이스 interface

메소드 오버로딩

- 메소드 오버로딩(Overloading)
  - 이름이 같은 메소드 작성
    - 매개변수의 개수나 타입이 서로 다르고
    - 이름이 동일한 메소드들
  - 리턴 타입은 오버로딩과 관련 없음

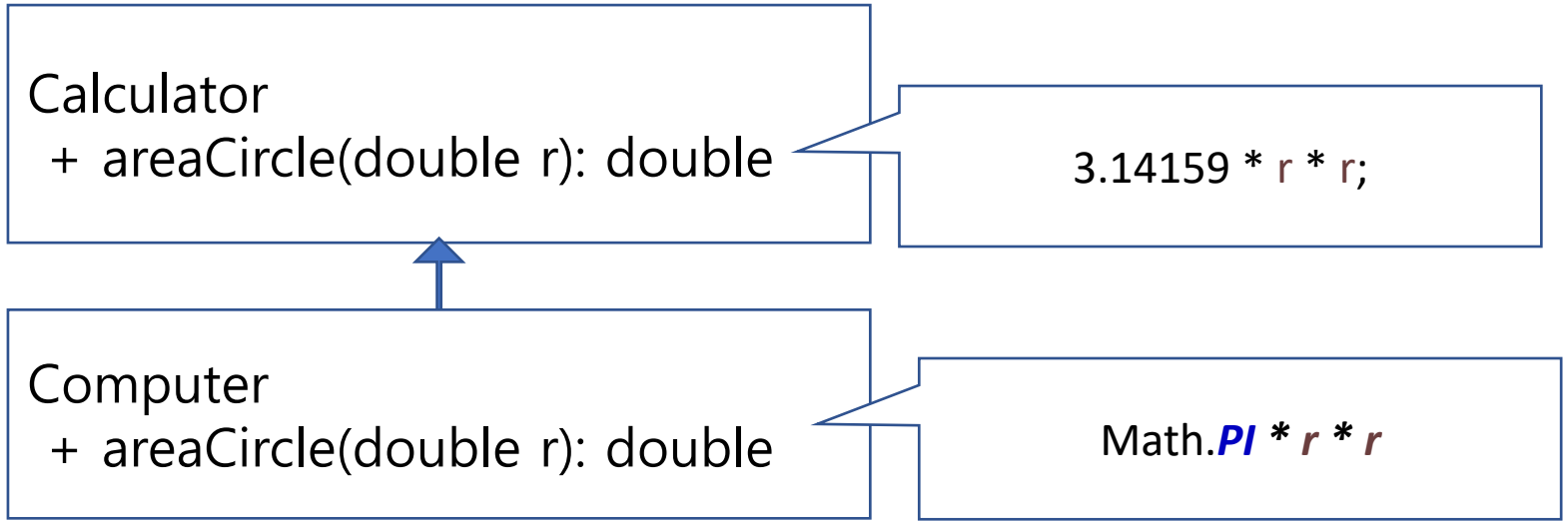
```
// 메소드 오버로딩이 성공한 사례

class MethodOverloading {
   public int getSum(int i, int j) {
      return i + j;
   }
   public int getSum(int i, int j, int k) {
      return i + j + k;
   }
}
```

```
// 메소드 오버로딩이 실패한 사례

class MethodOverloadingFail {
   public int getSum(int i, int j) {
      return i + j;
   }
   public double getSum(int i, int j) {
      return (double)(i + j);
   }
}
```

두 개의 getSum() 메소드는 매개변수의 개수, 타입이 모두 같기 때문에 메소드 오버로딩 실패

실습: 메서드 오러라이딩

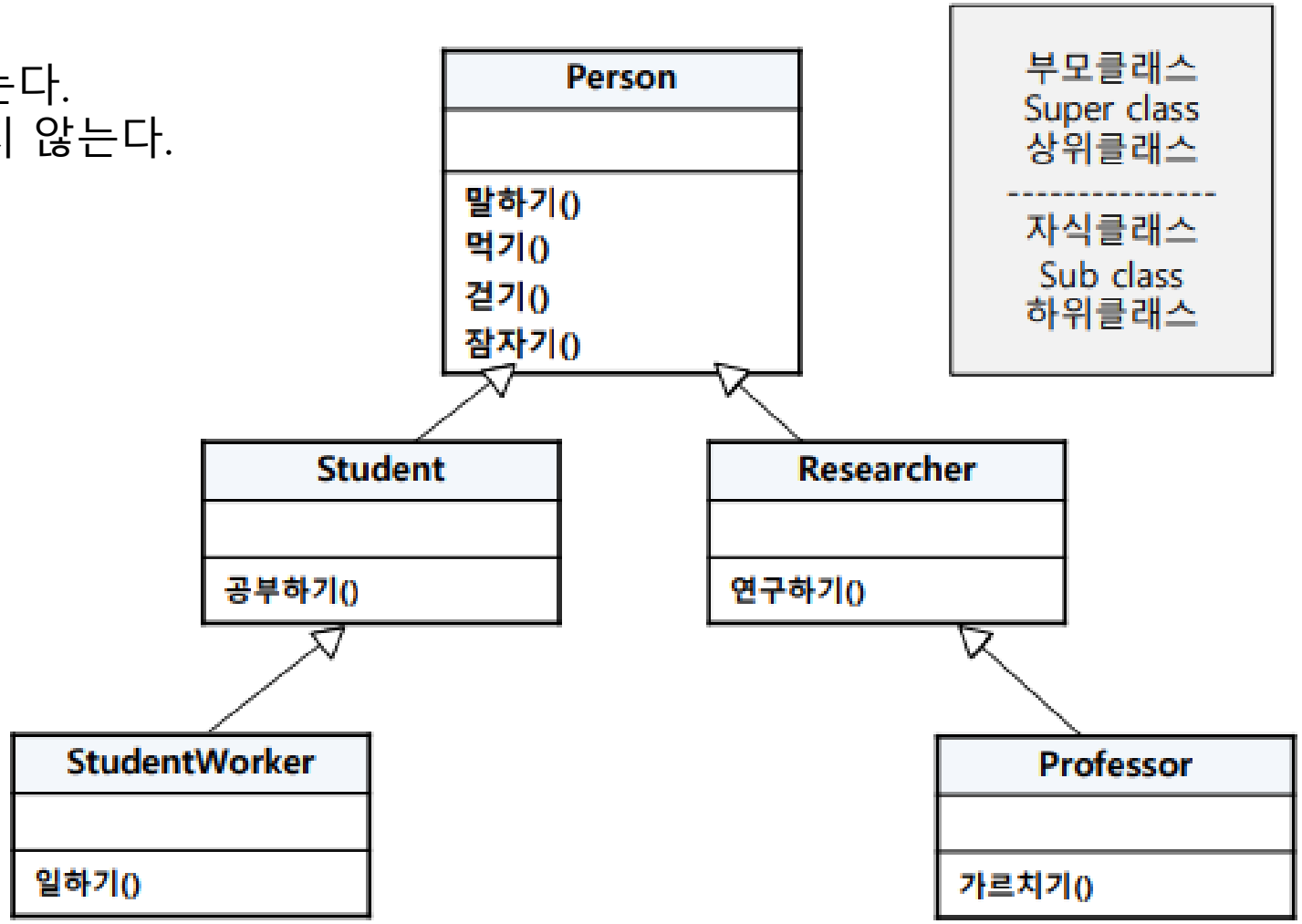
- Calcurator 를 상속받는 Computer




```
Main:
int r = 10;
Calculator calculator = new Calculator();
System.out.println("원면적 : " + calculator.areaCircle(r));
System.out.println();
Computer computer = new Computer();
System.out.println("원면적 : " + computer.areaCircle(r));
```

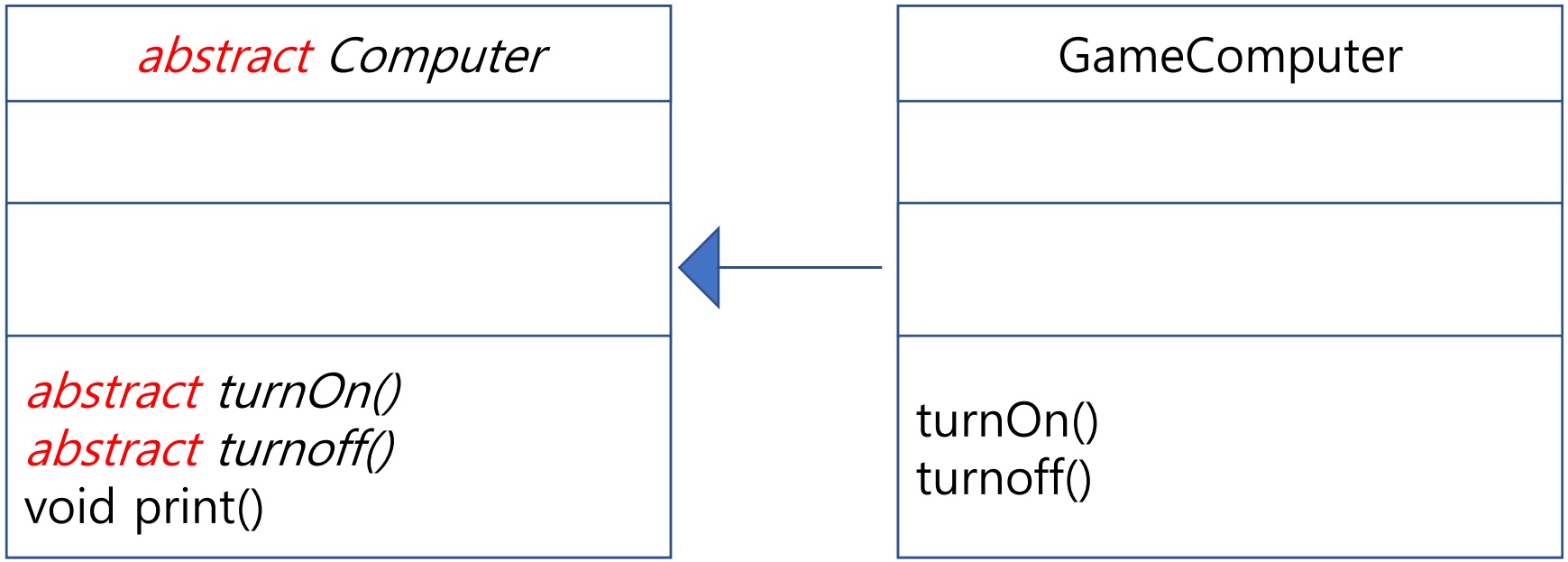
다형성

자바상속의 특징
– 자바에서는 다중 상속을 지원하지 않는다.
– 자바에서는 상속의 횟수에 제한을 두지 않는다.
– 자바에서 계측구조의 최상위에 있는
클래스는 java.lang.Object 이다.



실습: 추상 클래스/메서드

- P280



실습: 추상 클래스/메서드

- P280

```
public abstract class Computer {

    abstract void display();
    abstract void typing();

    public void print() {
        System.out.println("");
    }

}
```

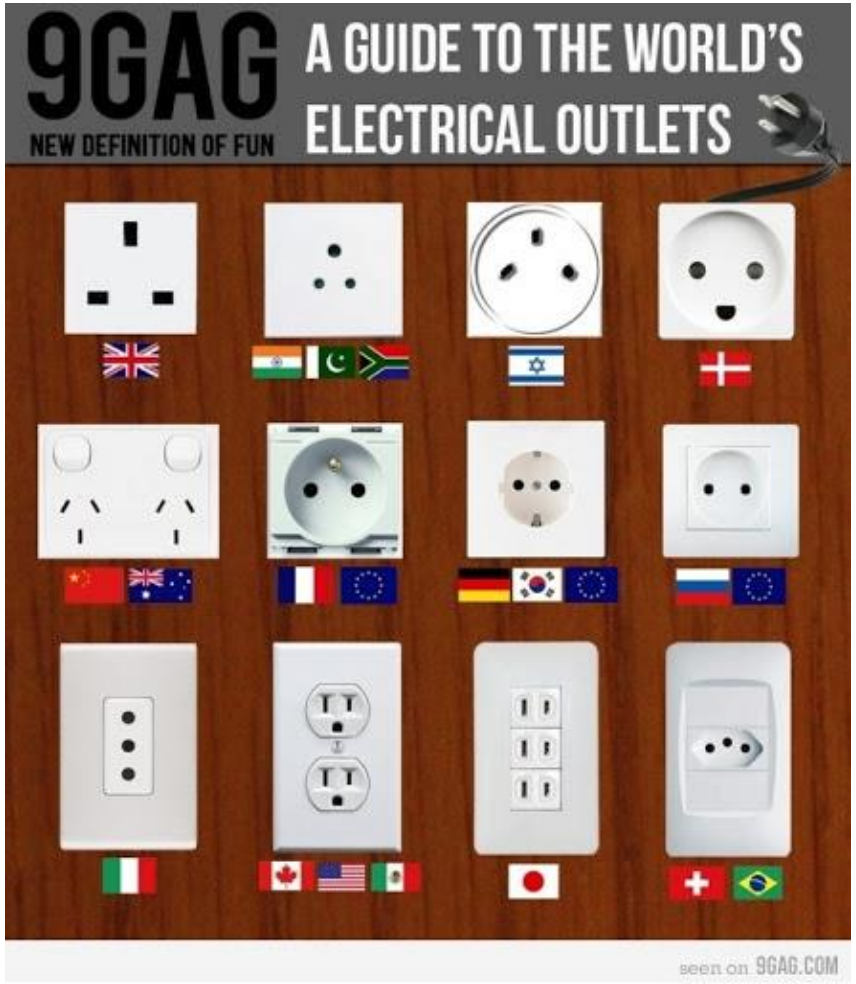
```
public class ComputerMain {

    public static void main(String[] args) {

        Computer c1 = new Computer();
        Computer c2 = new
        c2.display();
        c2.typing();
        //
        c2.print();
    }

}
```

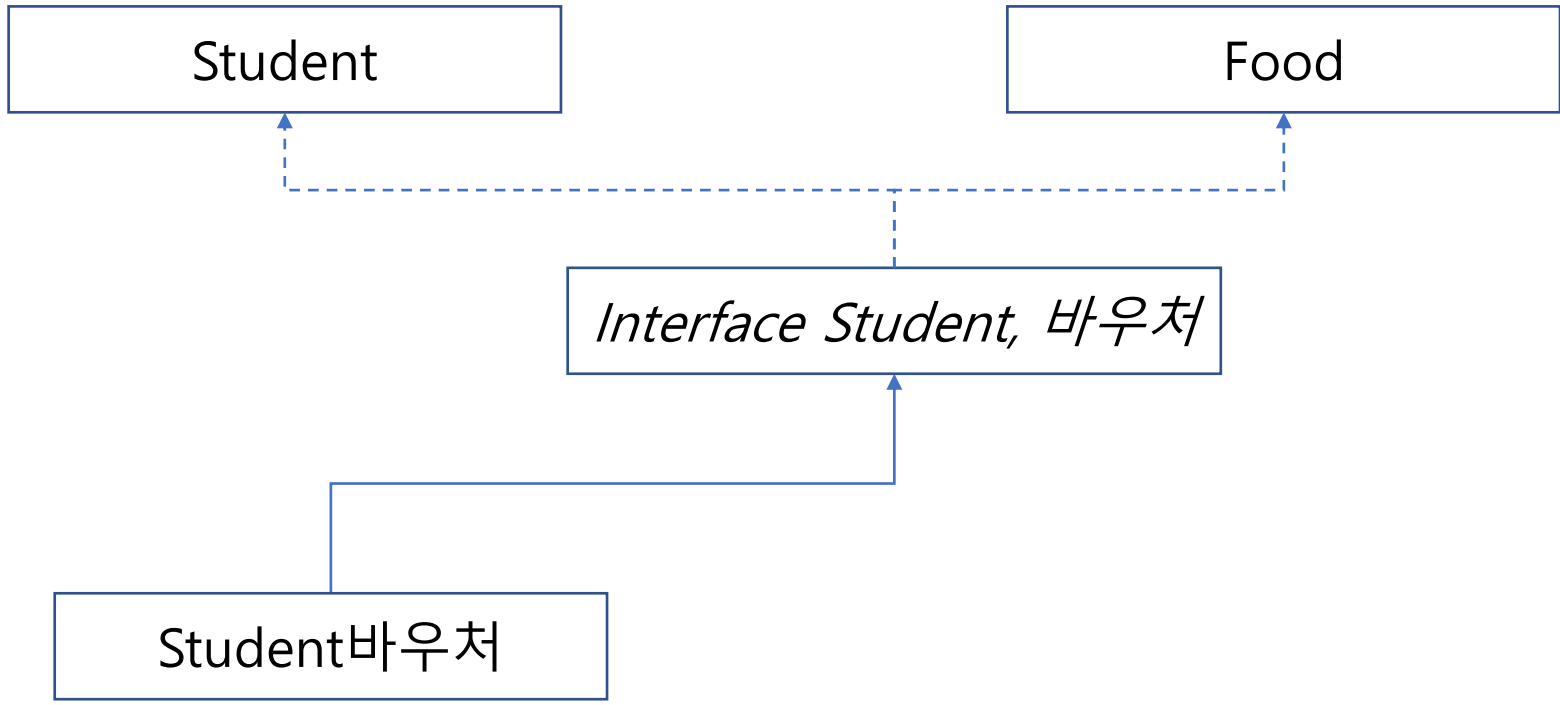
Cannot instantiate the type Computer

```
public class ComputerMain {

    public static void main(String[] args) {

        // Computer c1 = new Computer();
        Computer c2 = new Desktop();
        c2.display();
        c2.typing();
        //
        c2.print();
    }

}
```

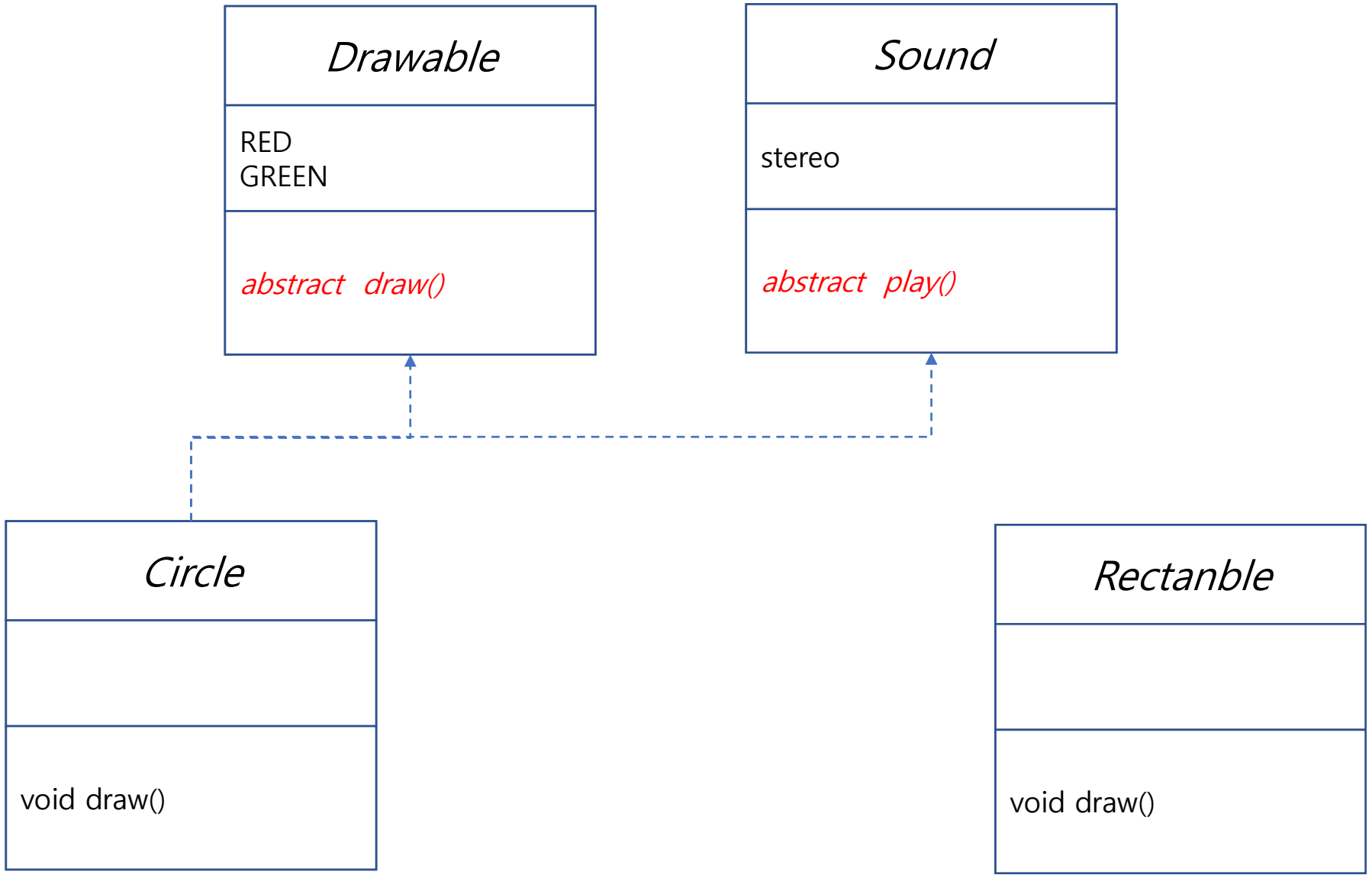
인터페이스

- P314


9GAG
NEW DEFINITION OF FUN
A GUIDE TO THE WORLD'S
ELECTRICAL OUTLETS
seen on 9GAG.COM


Student
Food
Interface Student, 바우처
Student바우처

실습: 인터페이스


Drawable
RED
GREEN
abstract draw()
Sound
stereo
abstract play()
Circle
void draw()
Rectanble
void draw()

Day 12, 5/28

5/28일 Day12

- Ch11 P351 기본 클래스
  - 컬렉션
  - 내부 클래스
- Ch7 P199 배열/객체복사
- 실습?

Object class

- https://docs.oracle.com/javase/10/docs/api/java/lang/Object.html

- toString() 재정의

Object class

- https://docs.oracle.com/javase/10/docs/api/java/lang/Object.html

: equals()

- 두 객체의 비교시 == 와 Object 클래스의equals() 메서드를 사용한다
- == 와 equals()의 차이 : 확실히 구분하여 사용
  - == 참조변수값 비교
  - equals() : 정의한 값 비교

  - 참조 변수값을 먼저 비교한다
  - 참조변수값이 같으면 두 객체는 같은 것으로 한다
  - 참조변수값이 다르면 두 객체의 속성값을 비교한다

Object class

- https://docs.oracle.com/javase/10/docs/api/java/lang/Object.html
- == 와 equals(), hashCode()

```java
@Override
public boolean equals(Object obj) {
    if (obj instanceof Student) {
        Student std = (Student) obj;
        if (studentId == std.studentId)
            return true;
        else
            return false;
    }
    return false;
}
@Override
public int hashCode() {
    return x;
}
```

Object class

- https://docs.oracle.com/javase/10/docs/api/java/lang/Object.html
- clone(), 얕은복사/깊은복사 P213~P217

- 객체 복제 : 원본 객체의 값과 동일한 값을 가진 새로운 객체를 생성하는 것
- 얕은 복제 : 단순히 필드 값을 복사하는 방식으로 객체를 복제하는 것
- Cloneable 인터페이스를 구현하여 clone 메서드를 사용 가능하게 해야 한다

```
public class Point implements Cloneable {
    private int x;
    private int y;

    // ...
    public Point getClone() {
        Point clone = null;
        try {
            clone = (Point)clone();
        } catch (CloneNotSupportedException e) {}
        return clone;
    }
}
```

System class

- https://docs.oracle.com/javase/10/docs/api/java/lang/System.html

System 클래스 용도

- 운영체제의 기능 일부 이용 가능
  - 프로그램 종료, 키보드로부터 입력, 모니터 출력, 메모리 정리, 현재 시간 읽기
  - 시스템 프로퍼티 읽기, 환경 변수 읽기

static void	gc()	Runs the garbage collector.
static Map<String,String>	getenv()	Returns an unmodifiable string map view of the current system environment.
static String	getenv(String name)	Gets the value of the specified environment variable.
static String	getProperty (String key, String def)	Gets the system property indicated by the specified key.

Garbage Collection

Garbage Collection

- 객체 소멸을 다루는 JVM 메커니즘

객체의 소멸과 가비지 컬렉션

- 객체 소멸
  - new에 의해 할당된 객체 메모리를 자바 가상 기계의 가용 메모리로 되돌려 주는 행위
- 자바 응용프로그램에서 임의로 객체 소멸할 수 없음
  - 객체 소멸은 자바 가상 기계의 고유한 역할
  - 자바 개발자에게는 매우 다행스러운 기능
    - C/C++에서는 할당받은 객체를 개발자가 되돌려 주어야 함
      - C/C++ 프로그램 작성을 어렵게 만드는 요인
- 가비지
  - 가리키는 레퍼런스가 하나도 없는 객체
    - 누구도 사용할 수 없게 된 메모리
- 가비지 켈렉션
  - 자바 가상 기계의 가비지 컬렉터가 자동으로 가비지 수집 반환

Garbage Collection 정리

- 가비지 컬렉션
  - 자바에서 가비지를 자동 회수하는 과정
    - 가용 메모리로 반환
  - 가비지 컬렉션 스레드에 의해 수행
- 개발자에 의한 강제 가비지 컬렉션
  - System 또는 Runtime 객체의 gc() 메소드 호출

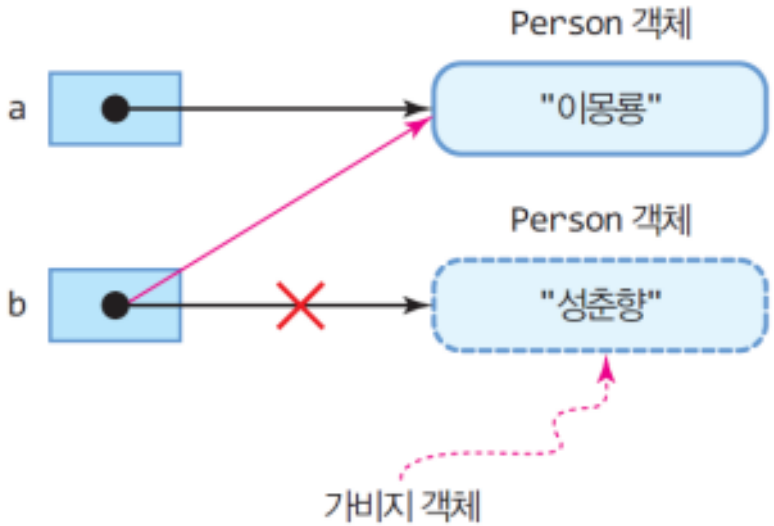
```
System.gc(); // 가비지 컬렉션 작동 요청
```

    - 이 코드는 자바 가상 기계에 강력한 가비지 컬렉션 요청
      - 그러나 자바 가상 기계가 가비지 컬렉션 시점을 전적으로 판단

Garbage Collection

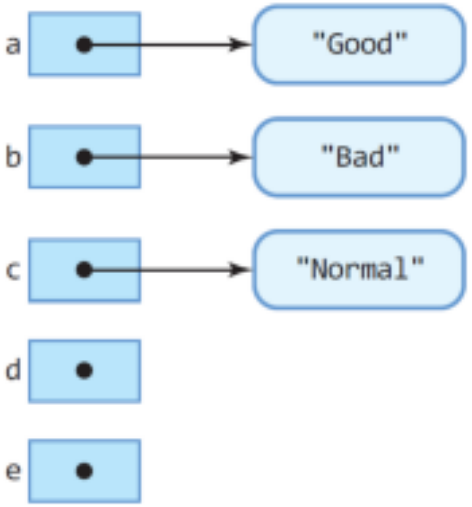
- 소멸 대상

```
Person a, b;
a = new Person("이몽룡");
b = new Person("성춘향");
b = a; // b가 가리키던 객체는 가비지가 됨
```

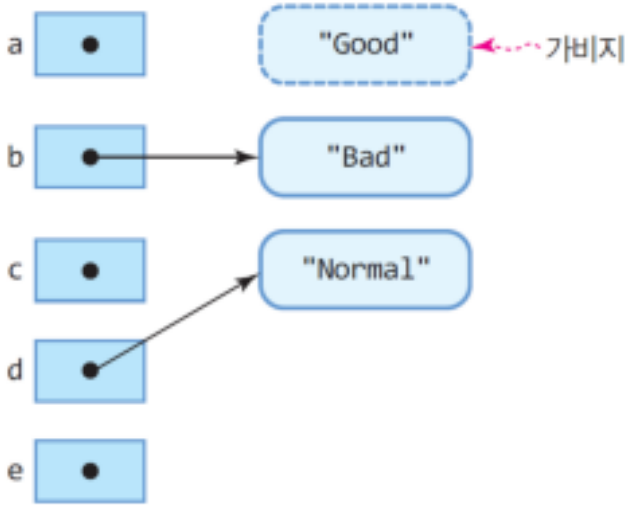


실습: Garbage Collection

```
public class GarbageEx {
    public static void main(String[] args) {
        String a = new String("Good");
        String b = new String("Bad");
        String c = new String("Normal");
        String d, e;
        a = null;
        d = c;
        c = null;
    }
}
```



(a) 초기 객체 생성 시(라인 6까지)



(b) 코드 전체 실행 후

실습: Garbage Collection

- 소멸자 De-Constructor.

```java
class Employee {
    int eno;

    public Employee(int eno) {
        this.eno = eno;
        System.out.println("Emplyee" + eno + " 생성");
    }

    @Override
    protected void finalize() throws Throwable {
        System.out.println("Emplyee" + eno + " 삭제");
    }
}
```

```java
public class SystemGcExample {
    public static void main(String[] args) {
        Employee emp = new Employee(1);
        emp = null;
        emp = new Employee(2);
        emp = new Employee(3);
        System.out.println("Emplyee " + emp.eno );
        System.gc();
    }
}
```

System class

- getProperty()
- getEnv()

키 (Key)	설명	값 (value)
java.version	자바의 버전	1.8.0_20
java.home	사용하는 JRE의 파일 경로	<jdk 설치경로>₩jre
os.name	Operating System name	Window 10
file.separator	File separator("₩" on window)	₩
user.name	사용자의 이름	사용자 계정
user.home	사용자의 홈 디렉토리	C:₩Users₩사용자 계정
user.dir	사용자가 현재 작업 중인 디렉토리 경로	다양

```
// getProperty()
String osName = System.getProperty("os.name");
String userName = System.getProperty("user.name");
```

Format string

- https://interconnection.tistory.com/116

```
// %[argument_index$][flags][width]conversion

/*
 * 필수 값 : conversion
    %d : 10진수(정수)
    %x : 16진수
    %o : 8진수
    %f : 실수
    %s : 문자열
 */
```

홍길동 100 (%), 수학 100, 과학 300

..

..

Format string

- https://interconnection.tistory.com/116

```
System.out.printf("%s [%d] %s [%d]", "홍길동", 100, "고길동", 200);
System.out.println();

String str = String.format("%s [%d] %s [%d]", "홍길동", 100, "고길동", 200);
System.out.println(str);

System.out.printf("%2$s %1$s", "홍길동", "고길동");
System.out.println();

System.out.println(String.format("%10d%10d", 12345, 67890));
```

Format string

- 날짜 포매팅

```
String form = "yyyy년 MM월 dd일 hh 시 mm분 ss초";
SimpleDateFormat sdf = new SimpleDateFormat(form);

// 날짜
Date dt = new Date();
String now = dt.toString();
System.out.println(now);
System.out.println(sdf.format(dt));
```

```
DecimalFormat df = new DecimalFormat("#,###0.0");
String result = df.format(1234459.99);
System.out.println(result);
```

제네릭

- 제네릭
  - 교재 P388-403
  - 생활코딩 https://opentutorials.org/course/1223/6237

제네릭 타입이란?

- 타입을 파라미터로 가지는 클래스와 인터페이스
- 선언 시 클래스 또는 인터페이스 이름 뒤에 “<>” 부호 붙임
- “<>” 사이에는 타입 파라미터 위치

- 타입 파라미터
  - 일반적으로 대문자 알파벳 한 문자로 표현
  - 개발 코드에서는 타입 파라미터 자리에 구체적인 타입을 지정해야

제네릭

제네릭 타입 사용 여부에 따른 비교

- 제네릭 타입을 사용하지 않은 경우
  - Object 타입 사용 → 빈번한 타입 변환 발생 → 프로그램 성능 저하

```
public class Box {
   private Object object;
   public void set(Object object) { this.object = object; }
   public Object get() { return object; }
}
```

```
Box box = new Box();
box.set("hello");                 //String 타입을 Object 타입으로 자동 타입 변환해서 저장
String str = (String) box.get();  //Object 타입을 String 타입으로 강제 타입 변환해서 얻음
```

제네릭

❖ 제네릭 타입 사용 여부에 따른 비교

- 제네릭 타입 사용한 경우
  - 클래스 선언할 때 타입 파라미터 사용
  - 컴파일 시 타입 파라미터가 구체적인 클래스로 변경

```
public class Box<T> {
   private T t;
   public T get() { return t; }
   public void set(T t) { this.t = t; }
}
```

```
Box<String> box = new Box<String>();
```

```
public class Box<String> {
   private String t;
   public void set(String t) { this.t = t; }
   public String get() { return t; }
}
```

```
Box<String> box = new Box<String>();
box.set("hello");
String str = box.get();
```

```
Box<Integer> box = new Box<Integer>();
```

```
public class Box<Integer> {
   private Integer t;
   public void set(Integer t) { this.t = t; }
   public Integer get() { return t; }
}
```

```
Box<Integer> box = new Box<Integer>();
box.set(6);
int value = box.get();
```

5/31

Day13

5/31 진행

- 배열 교재 P199
- 예외처리
- 컬렉션 프레임워크 P404
  - 제네릭 P388
- Static keyword : P181
- Inner class

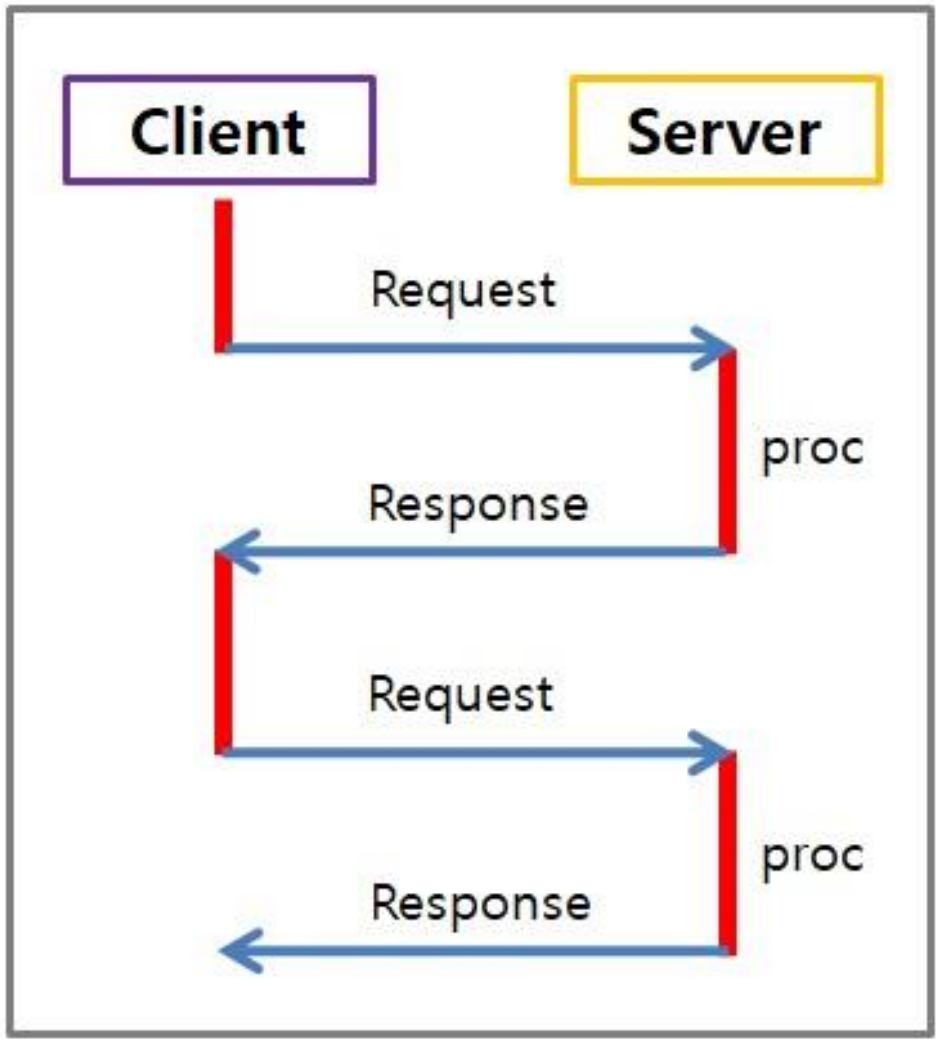
배열

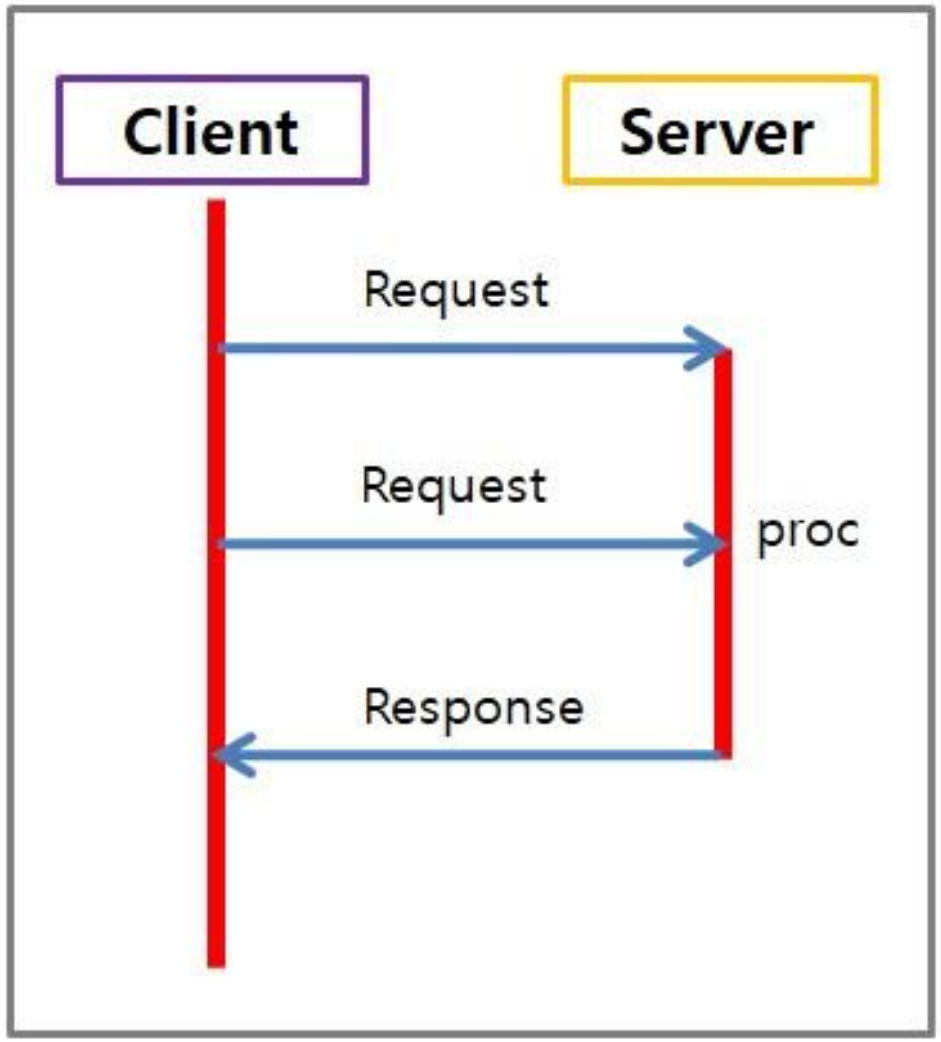
- 교재 P199 부터
- 01.자바_기본프로그래밍.pdf 배열 부분

컬렉션 프레임워크

- 교재 P404
  - 교재 배열 P222 ArrayList
  - 참고: 04.자바_컬렉션프레임워크(2p).pdf
  - 참고: 강의교안_15장-컬렉션프레임워크(2p).pdf
    - 참고:
- Mapping :

동기/비동기





https://jieun0113.tistory.com/73

Java.util.Stack, Queue 클래스

```
Stack<Coin> coinBox = new Stack<Coin>();

coinBox.push(new Coin(100));
coinBox.push(new Coin(50));
coinBox.push(new Coin(500));
coinBox.push(new Coin(10));

while(!coinBox.isEmpty()) {
    Coin coin = coinBox.pop();
    System.out.println("꺼내온 동전 : " + coin.getValue() + "원");
}


Queue<Message> messageQueue = new LinkedList<Message>();

messageQueue.offer(new Message("sendMail", "홍길동"));
messageQueue.offer(new Message("sendSMS", "신용권"));
messageQueue.offer(new Message("sendKakaotalk", "홍두께"));

while(!messageQueue.isEmpty()) {
    Message message = messageQueue.poll();
    switch(message.command) {
        case "sendMail":
            System.out.println(message.to + "님에게 메일을 보냅니다.");
            break;
```

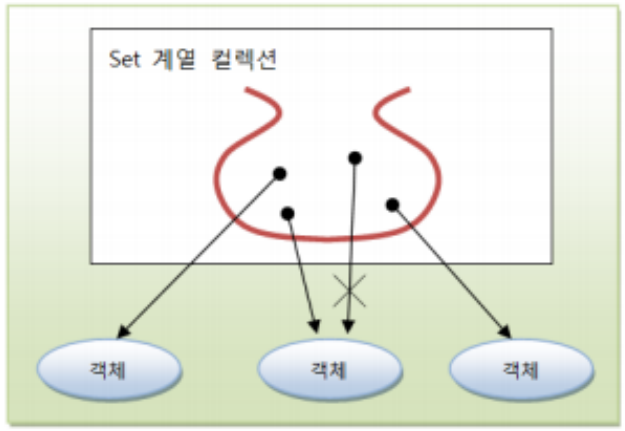
Arrays.asList()

- Java.util.Arrays
  - 나열형 데이터를 List 객체로 반환

```
List<String> list1 = Arrays.asList("홍길동", "신용권", "감자바");
for(String name: list1) {
    System.out.println(name);
}

List<Integer> list2 = Arrays.asList(1, 2, 3);
for(int value : list2) {
    System.out.println(value);
}
```

Set collection

- **Set 컬렉션의 특징 및 주요 메소드**
  - **특징**
    - 수학의 집합에 비유
    - 저장 순서가 유지되지 않음
    - 객체를 중복 저장 불가
    - 하나의 null만 저장 가능
  - **구현 클래스**
    - HashSet, LinkedHashSet, TreeSet



Set

■ **셋(set)>HashSet**

- 순서없이 저장된다.
- 중복값은 저장되지 않는다. 따라서 **중복의 재정의**가 필요하다.
- 수학의 집합에 비유

Set	List
순서없음	순서유지
중복 저장 안됨	중복 저장 가능

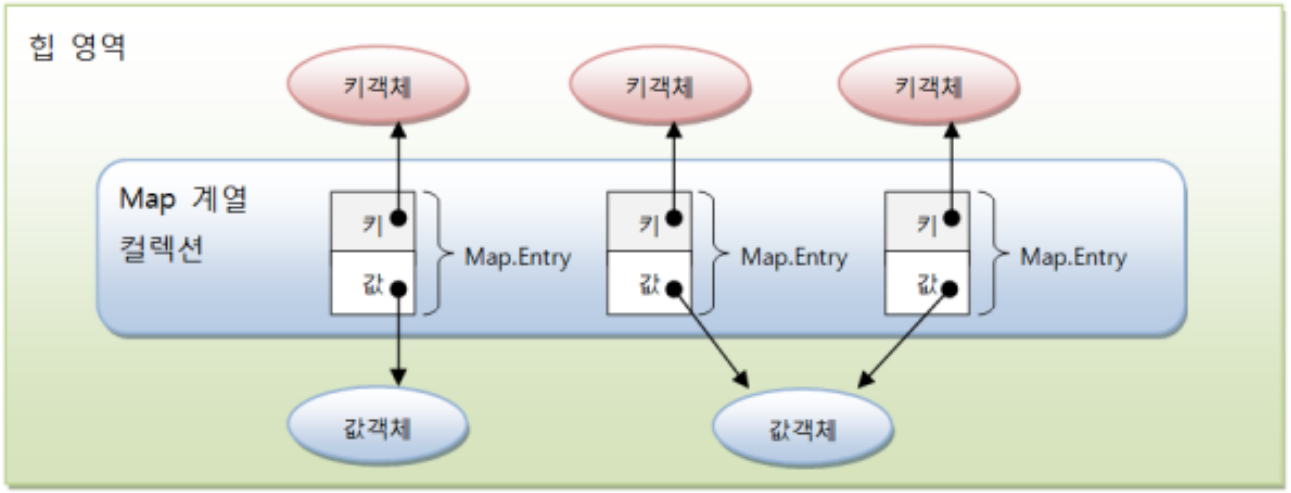
중복정의가 중요
→클래스의 hashCode() 와 equals() 메소드 재정의 필요

Map Collection

Map 컬렉션의 특징 및 주요 메소드

- 특징
  - 키(key)와 값(value)으로 구성된 Map.Entry 객체를 저장하는 구조
  - 키와 값은 모두 객체
  - 키는 중복될 수 없지만 값은 중복 저장 가능

- 구현 클래스
  - HashMap, Hashtable, LinkedHashMap, Properties, TreeMap



static

- Static keyword : P181

- 멤버변수
- 지역변수
- 정적변수