

클래스

클래스

클래스 표기법

분석 단계의 상세화

프로퍼티

관계(Associations)

가시성

객체 개수

방향성

클래스 간 관계

양방향 연관

단방향 연관

연관 클래스

집합 vs 합성

문제 1

문제 2

정적 필드(Static Field)

정적 상수

정적 메서드

생성자

정의

this

JVM 메모리 모델

상속

문제 3

공부할 거

cf) 우리나라에서는 JPA보다 Mybatis를 더 많이 쓴다.

cf2) 다른 게 다 바뀌어도 도메인과 서비스는 바뀌면 안 된다.

클래스

클래스 표기법

- 네모 모양 표기
- 이름, 속성, 오퍼레이션(메서드) 나타내는 세 개의 구획

분석 단계의 상세화

```
class SearchService {  
    SearchEngine engine;  
    Configuration config;  
  
    private static SearchEngine createEngine() {
```

```

    return null;
}

SearchResult search(SearchRequest query) {
    return null;
}

```

프로퍼티

관계(Associations)

- 참조 객체의 경우
- Role Name: 연관 형태 명확하게 하기 위해 역할 이름 가짐
- Navigability: 화살표 가리키는 방향에 있는 클래스를 탐색하거나 질의할 수 있음
- Multiplicity: 하나의 인스턴스에 참여하는 다른 쪽의 인스턴스 개수 표현

가시성

- +: public
- -: private
- ~: package
- #: protected

객체 개수

- 0..1: 0 혹은 1
- *: 무제한(0 포함)
- 1..*: 적어도 1 이상

방향성



주체 —→ 대상

person —→ car (owns)

```

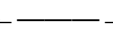
class person {
    Car car;
}

```

클래스 간 관계

관계	UML 표기
Generalization (일반화)	
Realization (실체화)	
Dependency (의존)	
Association (연관)	
Directed Association (직접연관)	
Aggregation (집합, 집합연관)	
	
Composition (합성, 복합연관)	
	

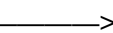
양방향 연관

A  B

```
class A {
    B b;
}

class B {
    A a;
}
```

단방향 연관

A  B (+b, 3)

```
class A {
    B[] b = new B[3];
}

class B {
}
```

연관 클래스

A  B
List

- 관계 자체에서 다른 엔티티가 나올 수 있다

```
class A {
    List<B> b;
}

class B {
}
```

- 의존 관계: 점선
 - 특정 클래스의 메서드 내에서만 사용 가능

A ----->B

```
class A {
    void method (B b) {
    }
}

class B {
}
```

- 연관 관계: 실선
 - 특정 클래스의 필드로서 존재

A —————> B

```
class A {
    B b;
}
```

집합 vs 합성

- 전체-부분의 관계, 얼마나 더 끈끈한가
 - 집합(Aggregation): 느슨한 관계 (흰색)
 - 합성(Composition): 강한 관계 (검은색)

문제 1

과제>Video 관리 프로그램을 만들어 보자.

클래스 : 모든 멤버변수는 private 선언

1. Video --> 비디오 정보를 담고 있다. (번호, 제목, 배우)
2. GeneralMember -> 일반회원에 대한 정보 (아이디, 이름, 주소, 빌린비디오)
-> 일반회원내역 출력 (아이디, 이름, 주소, 빌린비디오 정보)
3. SpecialMember -> 특별회원에 대한 정보 (보너스포인트)
-> 보너스정보 출력

4. main() 갖고 있는 클래스 결과>

회원의 아이디 : aaa

회원의 이름 : 홍길동

회원의 주소 : 동탄

회원이 대여한 비디오 번호 : 1

회원이 대여한 비디오 제목 : 트랜스포머3

회원이 대여한 비디오 주인공 : 서봉수

회원의 아이디 : bbb

회원의 이름 : 김철수

회원의 주소 : 서울

회원이 대여한 비디오 번호 : 2

회원이 대여한 비디오 제목 : 쿵다펀더2

회원이 대여한 비디오 주인공 : 지성민

회원의 보너스 포인트 적립 : 10

kosta.video 패키지를 만든다

프로그램을 만들려면 어떤 객체가 필요할까 생각을 해봐야겠죠?

비디오, 제너럴멤버= (아이디, 이름, 주소, (객체 비디오) 비디오 정보 (이회사는 한개만 쓴다 배열안쓰고)

CLASS A{ 빌린 비디오 정보는 연관관계이다. 그래서 video객체를 써야함

B b;

}

스페셜멤버= 당골고객. 메리트 줘야함. 보너스정보 출력

스페셜멤버가 아들이고 제너럴이 부모클래스구만

둘다 화면에 출력하는 메소드 있어야함.

클래스 몇개 있어야겠어?

4개 있어야한다 메인까지 포함해서.

- Video.java

```
package kosa.video;

public class Video {
    private String sno;
    private String title;
    private String actor;

    public Video() {
    }

    public void show() {
        System.out.println("회원이 대여한 비디오 번호: " + sno);
        System.out.println("회원이 대여한 비디오 제목: " + title);
        System.out.println("회원이 대여한 비디오 주인공: " + actor);
    }

    public Video(String sno, String title, String actor) {
        super();
        this.sno = sno;
        this.title = title;
        this.actor = actor;
    }
}
```

```

    public String getSno() {
        return sno;
    }

    public void setSno(String sno) {
        this.sno = sno;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getActor() {
        return actor;
    }

    public void setActor(String actor) {
        this.actor = actor;
    }
}

```

- GeneralMember.java

```

package kosa.video;

public class GeneralMember {
    private String id;
    private String name;
    private String address;
    // 멤버 변수로 비디오 객체 가지고 있기
    private Video rentalVideo;

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public Video getRentalVideo() {
        return rentalVideo;
    }
}

```

```

public void setRentalVideo(Video rentalVideo) {
    this.rentalVideo = rentalVideo;
}

public GeneralMember() {
}

public GeneralMember(String id, String name, String address) {
    this.id = id;
    this.name = name;
    this.address = address;
}

// 비디오를 여러 개 빌릴 것도 가정
// 자료 구조가 list 등으로 바뀔 경우 set 메서드로 사용 불가
public void rental(Video video) {
    rentalVideo = video;
}

public void show() {
    System.out.println("회원의 아이디: " + id);
    System.out.println("회원의 이름: " + name);
    System.out.println("회원의 주소: " + address);
    rentalVideo.show();
}
}

```

- VideoMain.java

```

package kosa.video;

public class VideoMain {

    public static void main(String[] args) {
        Video v1 = new Video("1", "탐건 2", "툼 크루즈");
        Video v2 = new Video("2", "헤어질 결심", "탕웨이");

        GeneralMember gm = new GeneralMember("abcd1234", "김김김", "서울");
        gm.rental(v1);
        gm.show();
    }
}

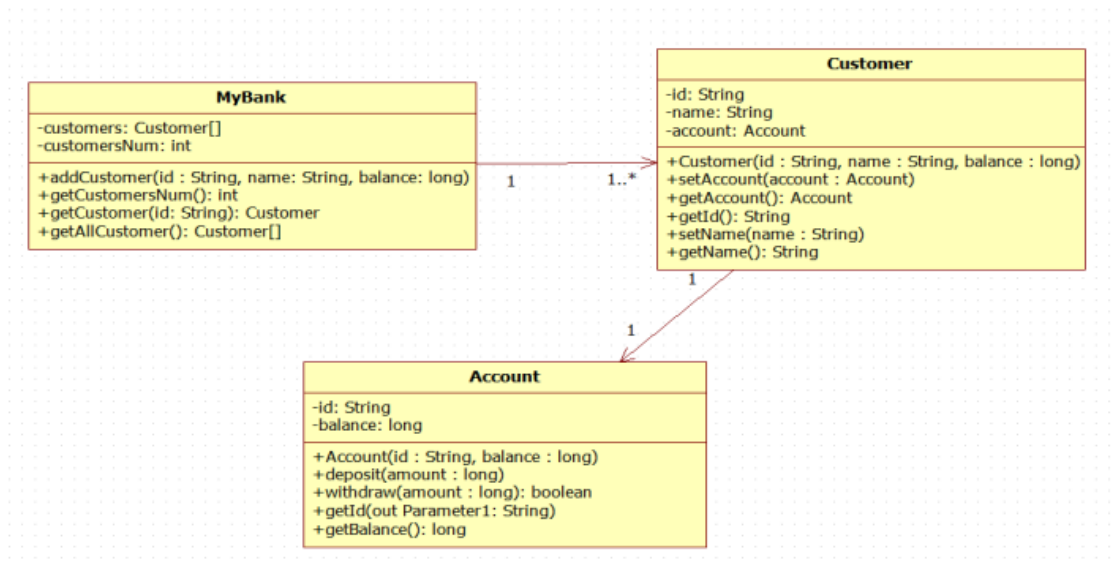
```

문제 2

- 오늘 푼 거 복습

1. Bank 과제

✓ Bank 클래스 다이어그램



- Account.java

```
package kosa.bank;

public class Account {
    private String id;
    private long balance;

    public Account() {}

    public Account(String id, long balance) {
        super();
        this.id = id;
        this.balance = balance;
    }

    public void deposit(long amount) {
        balance += amount;
    }

    public boolean withdraw(long amount) {
        if(balance < amount) {
            return false;
        }
        balance -= amount;

        return true;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public long getBalance() {
```



```

        return balance;
    }

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

```

- Customer.java

```

package kosa.bank;

public class Customer {
    private String id;
    private String name;
    private Account account;

    public Customer() {}

    public Customer(String id, String name, long balance) {
        this.id = id;
        this.name = name;
        this.account = new Account(id, balance);
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Account getAccount() {
        return account;
    }

    public void setAccount(Account account) {
        this.account = account;
    }
}

```

- MyBank.java

```

package kosa.bank;

public class MyBank {
    private Customer customers[];
    private int customersNum;

    public MyBank() {

```

```

        customers = new Customer[10];
    }

    public void addCustomer(String id, String name, long balance) {
        customers[customersNum++] = new Customer(id, name, balance);
    }

    public Customer getCustomer(String id) {
        Customer cust = null;
        for(int i=0;i<customersNum;i++) {
            if(customers[i].getId().equals(id)) {
                cust = customers[i];
                break;
            }
        }
        return cust;
    }

    public Customer[] getAllCustomers() {
        Customer newCustomers[] = new Customer[customersNum];

        // for(int i=0;i<customersNum;i++) {
        //     newCustomers[i] = customers[i];
        // }

        // 복사해 올 배열, 시작점, 복사되는 배열, 시작점, 끝점
        System.arraycopy(customers, 0, newCustomers, 0, customersNum);

        return newCustomers;
    }
}

```

- BankSystem.java

```

package bank1.me;

import java.io.BufferedReader;
import java.io.InputStreamReader;

public class BankSystem {
    private MyBank myBank; // MyBank

    public BankSystem() {
        myBank = new MyBank();
        showMenu();
    }

    public void showMenu() { // show method 만들기
        String menu = null;
        String id = null;
        String name = null;
        long balance = 0;

        do {
            System.out.println("****메뉴를 입력하세요****");
            System.out.println("1. 고객등록");
            System.out.println("2. 고객보기(1명)");
            System.out.println("3. 고객전체보기");
            System.out.println("4. 고객예금출금");
            System.out.println("5. 고객예금입금");
            System.out.println("***끝내기***");
            System.out.println("*****");
            System.out.print(">");

```

```

menu = readFromKeyboard();

if (menu.equals("1")) { // 고객등록

    System.out.print("ID를 입력하세요: ");
    id = readFromKeyboard();

    System.out.print("이름을 입력하세요: ");
    name = readFromKeyboard();

    System.out.print("잔고를 입력하세요: ");
    balance = Long.parseLong(readFromKeyboard());

    myBank.addCustomer(id, name, balance);

} else if (menu.equals("2")) {
    System.out.print("고객ID를 입력하세요: ");
    id = readFromKeyboard();
    Customer cust = myBank.getCustomer(id);
    System.out.println(cust.getId() + ":" + cust.getName() + ": " + cust.getAccount().getBalance());

} else if (menu.equals("3")) {

    Customer[] allCust = myBank.getAllCustomers();

    for (int i = 0; i < allCust.length; i++) {
        System.out.println(allCust[i].getId() + ": " + allCust[i].getName() + " : "
            + allCust[i].getAccount().getBalance());
    }

} else if (menu.equals("4")) {

    System.out.print("고객의 ID를 입력하세요.: ");
    id = readFromKeyboard();

    Customer cust = myBank.getCustomer(id);

    if (cust == null) {
        System.out.println("등록된 고객이 아닙니다.");
    } else {
        System.out.print("출금액을 입력하세요: ");
        balance = Long.parseLong(readFromKeyboard());

        if (cust.getAccount().withdraw(balance)) {
            System.out.println("정상적으로 출금되었습니다.");
            System.out.println("출금후 잔고는 : " + cust.getAccount().getBalance());
        } else {
            System.out.println("잔고가 부족합니다.");
        }
    }
}

} else if (menu.equals("5")) {
    System.out.println("고객 ID를 입력하세요.");
    id = readFromKeyboard();

    Customer cust = myBank.getCustomer(id);

    if (cust == null) {
        System.out.println("등록된 고객이 아닙니다.");
    } else {
        System.out.print("입금액을 입력하세요: ");
        balance = Long.parseLong(readFromKeyboard());
        cust.getAccount().deposit(balance);
        System.out.println("입금 후 잔고는 : " + cust.getAccount().getBalance());
    }
}
}

```

```

    } while (!menu.equals("q"));
}

public String readFromKeyboard() {
    String input = null;
    try {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        input = br.readLine();
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
    return input;
}

public static void main(String[] args) {
    BankSystem bank = new BankSystem();
}
}

```

정적 필드(Static Field)

- 해당 클래스의 모든 인스턴스들이 공유하는 클래스 변수
 - 클래스 이름으로 접근

```
Employee.id;
```

- 처음 클래스가 로딩될 때 딱 한 번만 생성
- static + 인스턴스 변수

정적 상수

```
static final;
```

정적 메서드

```

class Exam {
    // 객체를 생성해야 사용 가능 (메모리에 아직 올라오지 않았음)
    private int a;
    // class 생성될 때 메모리에 자동 생성 (static 객체)
    private static int b;

    public static void method() {
        // static 변수만 사용 가능
        a += 1;
        b += 10;
    }
}

```

생성자

정의

- 클래스와 이름이 같은 메소드
- 클래스 초기화 관련 작업
- 생성자 이름 앞에 new 키워드 붙여서 호출

this

- 객체 자신을 나타내는 키워드
- 객체 자신의 필드 참조, 해당 클래스의 다른 생성자 호출할 때 사용
- 정적(Static) 메서드에서는 this를 사용할 수 없다

JVM 메모리 모델

- 메소드 영역
- 스택 영역: 지역변수, 매개변수 할당. 자동으로 초기화되지 않는다.
- 힙 영역: 배열 공간이나 클래스에 포함된 멤버를 할당하는 영역 (G.C 대상)

상속

- Account
 - 멤버 변수: 계좌 번호, 계좌 주인 이름, 잔액
 - 기능: 입금하기, 출금하기, 계좌 정보 출력하기
- Checking Account
 - 멤버 변수: 카드 번호
 - 기능: 결제하기(pay 메서드: int)
 - 카드 번호가 다르거나 결제 금액 초과하면 결제 불가
- CheckingAccount가 Account 상속

```
public class CheckingAccount extends Account {
    private String cardNo;

    public CheckingAccount() {
    }

    // 자식이 생성자면 메모리에 부모도 올라간다
    // 그냥 두면 default가 올라가기 때문에, super 함수 사용
    public CheckingAccount(String accountNo, String ownerName, int balance, String cardNo) {
        // 자식에서 부모의 생성자 호출 => 자식에서 데이터 값을 부모로 올리기 위해 사용
        super(accountNo, ownerName, balance);
        this.cardNo = cardNo;
    }

    // pay 메서드 만들기
    public int pay(String cardNo, int amount) throws Exception {
```

```

        // balance는 private 변수라서 가져올 수 없다
        // getBalance() 메서드 사용해서 가져오기
        if(!cardNo.equals(this.cardNo) || getBalance() < amount) {
            throw new Exception("잔액 부족으로 결제할 수 없습니다.");
        }
        // 부모 메서드 사용
        return withdraw(amount);
    }
}

```

```

public class AccountMain {

    public static void main(String[] args) {
        CheckingAccount ca = new CheckingAccount("444-444", "김민주", 10000, "444-444");
        // try-catch로 pay 감싸 주기
        try {
            ca.pay("444-444", 3000);
            // 예외가 발생하면 여기 라인은 실행 안 됨
        } catch (Exception e) {
            e.printStackTrace();
        } // finally: 예외가 발생하든 발생하지 않은 무조건 실행
        finally {

        }

        ca.printAccount();
    }
}

```

문제 3

- 특별 회원이 일반 회원 상속
- 보너스 += 10점씩

```

package kosa.video;

public class GeneralMember {
    private String id;
    private String name;
    private String address;
    private Video rentalVideo;

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getAddress() {
        return address;
    }
}

```

```

    }

    public void setAddress(String address) {
        this.address = address;
    }

    public Video getRentalVideo() {
        return rentalVideo;
    }

    public void setRentalVideo(Video rentalVideo) {
        this.rentalVideo = rentalVideo;
    }

    public GeneralMember() {
    }

    public GeneralMember(String id, String name, String address) {
        this.id = id;
        this.name = name;
        this.address = address;
    }

    // 비디오를 여러 개 빌릴 것도 가정
    // 자료 구조가 list 등으로 바뀔 경우 set 메서드로 사용 불가
    public void rental(Video video) {
        rentalVideo = video;
    }

    public void show() {
        System.out.println("회원의 아이디: " + id);
        System.out.println("회원의 이름: " + name);
        System.out.println("회원의 주소: " + address);
    }
}

```

```

package kosa.video;

public class SpecialMember extends GeneralMember {
    int bonusPoint;

    public SpecialMember(String id, String name, String address, int bonusPoint) {
        super(id, name, address);
        this.bonusPoint = bonusPoint;
    }

    public SpecialMember() {
    }

    public int getBonusPoint() {
        return bonusPoint;
    }

    // public void rentalSpecialVideo(Video video) {
    //     super.rental(video);
    //     bonusPoint += 10;
    // }

    // public void showMemberInfo() {
    //     super.show();
    //     System.out.println("회원의 보너스 포인트 적립: " + bonusPoint);
    // }
}

```

```

// 부모 메서드 오버라이딩 (다형성 구현)
@Override
public void rental(Video video) {
    super.rental(video);
    bonusPoint += 10;
}

@Override
public void show() {
    super.show();
    System.out.println("회원의 보너스 포인트 적립: " + bonusPoint);
}
}

```

```

package kosa.video;

public class VideoMain {

    public static void main(String[] args) {
        Video v1 = new Video("1", "탐견 2", "툼 크루즈");
        Video v2 = new Video("2", "헤어질 결심", "탕웨이");

        // 회원을 한 군데에서 관리하기 위하여 형 변환 사용
        // 데이터 형태가 다르더라도 한 배열에서 전부 관리 가능 (부모이기 때문)
        GeneralMember[] arr = {
            new GeneralMember("abcd1234", "김민주", "서울"),
            new SpecialMember("defg5678", "김채원", "경기", 50)
        };

        // 부모로 형 변환 해서 자식 메서드는 사용 불가
        // overriding 통해 다형성 구현
        for(int i = 0; i < arr.length; i++) {
            arr[i].show();
            arr[i].rental(v1);
        }

        // GeneralMember gm = new GeneralMember("abcd1234", "김민주", "서울");
        // SpecialMember sp = new SpecialMember("defg5678", "김채원", "경기", 50);
        // gm.rental(v1);
        // gm.show();
        // sp.rentalSpecialVideo(v2);
        // sp.showMemberInfo();
    }
}

```

공부할 거

- 배열, String
- 객체지향(도메인, 관계)
- Collection(List)
- 정규 표현식