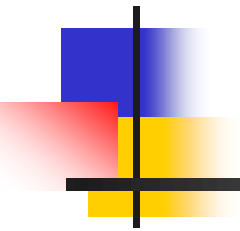


학생 성적처리
(클래스와 객체)

경복대학교
소프트웨어융합과
배희호 교수

학생 성적처리

- 홍길동 학생의 학번(hakbun), 이름(name), 국어(kor), 영어(eng), 수학(math) 성적을 입력 받아 학생의 총점(sum), 평균(avg)을 구하는 Program을 객체 지향 프로그램 기법을 이용하여 구하여라.

학생 성적처리

- 객체 지향 Modeling 절차
  - 문제 이해
    - 문제 영역의 대상들을 파악(객체 별 정보와 Data 파악)
  - 문제 영역에서 Class 식별해 내기
    - 명사들을 파악, 일반화 및 추상화
  - Class의 행위(Method), 속성 지정
    - 동사들을 파악, 메소드 파악
    - 클래스 추상화
  - 클래스간의 관계를 식별
    - 상속 관계(Super, Sub), 포함 관계
  - 클래스 구현
    - Member Field 생성
    - 생성자 구현
    - Setter()와 Getter() 구현
    - 메소드 구현

학생 성적처리 : 문제 파악

- 객체를 이용하여 학생의 성적을 관리를 위한 Class 생성
- 입력내용 – 학번, 이름, 성별, 국어, 영어, 수학, 선택 (Data)
  - 선택 과목은 학생의 성별에 따라 남자는 '기술'을, 여자는 '가정'을 선택함

학생 성적처리 : 문제 파악

- 출력결과 – 학번, 이름, 성별, 국어, 영어, 수학, 선택
  합계, 평균 (정보)

타 입	변수명	설 명
String	name	이름
String	hakbun	번호
char	gender	성별 (‘M’ = 남, ‘F’ = 여)
int	kor	국어점수
int	eng	영어점수
int	math	수학점수
int	option	선택점수

학생 성적처리 :클래스 구현

- Member Field 선언 (속성)
  - 클래스의 Field란 클래스에 포함된 변수(variable)를 의미
  - Field의 구분(선언된 위치에 따라)
    - 클래스 변수(static variable)
    - 인스턴스 변수(instance variable)
    - 지역 변수(local variable)
- 메소드 정의 (기능)
  - getter(), setter() 메소드
    - 클래스의 특성 중 정보 은닉(Information Hiding)을 가장 잘 보여주는 특별한 메소드
    - 보통 클래스의 Member 변수는 private로 접근 제한자를 설정한 후 getter()/setter()를 통해 Member 변수의 값을 변경, 호출하게 됨

학생 성적처리 :클래스 구현

- 접근 제어자 고려
  - Object Oriented Programming의 특징 중 하나인 정보 은닉(data hiding)을 위한 Keyword
  - ‘public’ Member는 모든 객체에서 접근할 수 있지만, ‘private’ Member는 해당 객체 내의 Member 변수나 메소드만이 접근할 수 있음

- 생성자 선언
  - 객체가 생성될 때 자동으로 실행되는 특수한 메소드
  - 반환형을 명시하지 않음
  - Class와 이름이 동일
  - Overloading 또한 가능

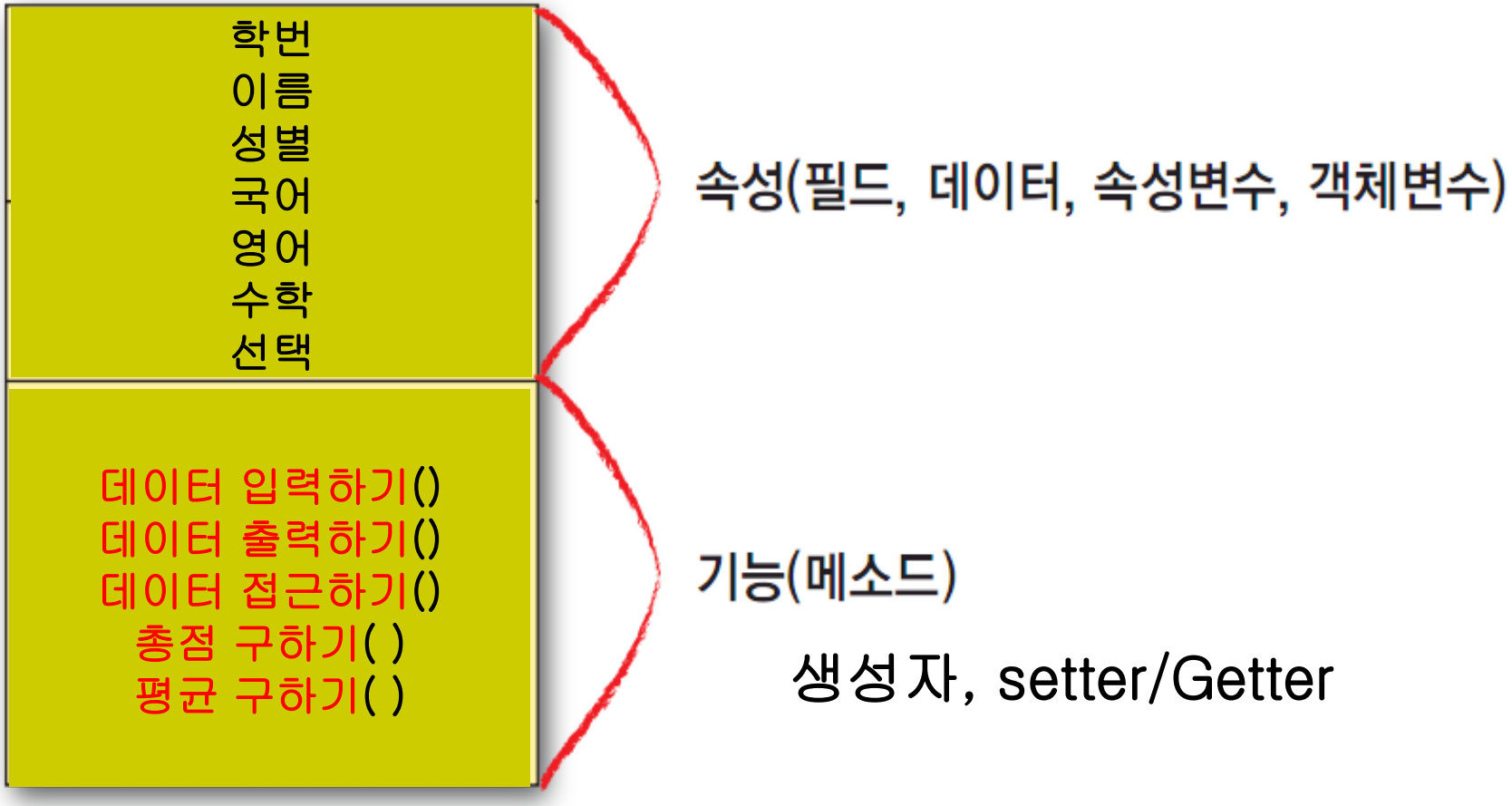
학생 성적처리 :클래스 구현

- Class Diagram

Student
-hakbun : String -name : String -gender : char -kor : int -eng : int -math : int -option : int
+생성자 +getKor() : int +getEng() : int +getMath() : int +getOption() : int +sum() : int +avg() : float +toString : String

학생 성적처리(I)

- Student 클래스 생성

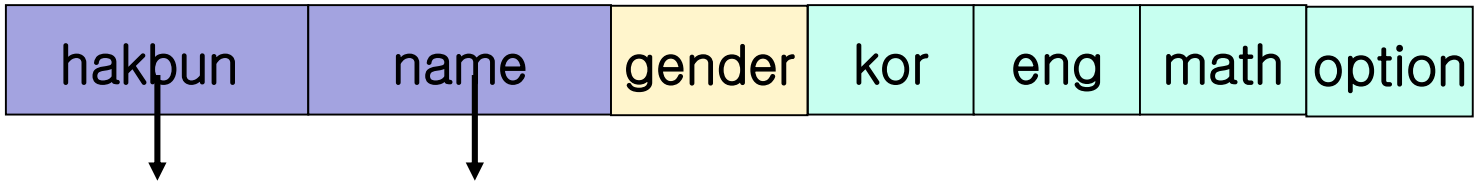


학생 성적처리(I)

- Student 클래스 생성

```
public class Student {
        private String hakbun;
        private String name;
        private char gender;
        private int kor;
        private int eng;
        private int math;
        private int option;
}
```

상태 (속성, 명사)는
Member Field로 구현



학생 성적처리(I)

```
public class Student {
	String hakbun;
	String name;
	char gender;
	int kor;
	int math;
	int eng;
	int option;

	public Student() {	// 기본 생성자

	}
```

학생 성적처리(I)

```
    public Student(String hakbun, String name, char gender, int kor,
                                        int math, int eng, int option) {
        this.hakbun = hakbun;
        this.name = name;
        this.gender = gender;
        this.kor = kor;
        this.math = math;
        this.eng = eng;
        this.option = option;
}
```

생성자

학생 성적처리(I)

- 정의한 Student 클래스에 다음과 같이 정의된 2개의 사용자 정의 메소드 sum()과 ave()를 추가

✓ 메소드명 : sum()
✓ 기 능 : 국어(kor), 영어(eng), 수학(math)의 점수를 모두 더해서 반환
✓ 반환 타입 : int
✓ 매개변수 : 없음

✓ 메소드명 : ave()
✓ 기 능 : 총점(국어점수 + 영어점수 + 수학점수)을 과목수로 나눈 평균을 구함 (소수점 둘째자리에서 반올림)
✓ 반환 타입 : float
✓ 매개변수 : 없음

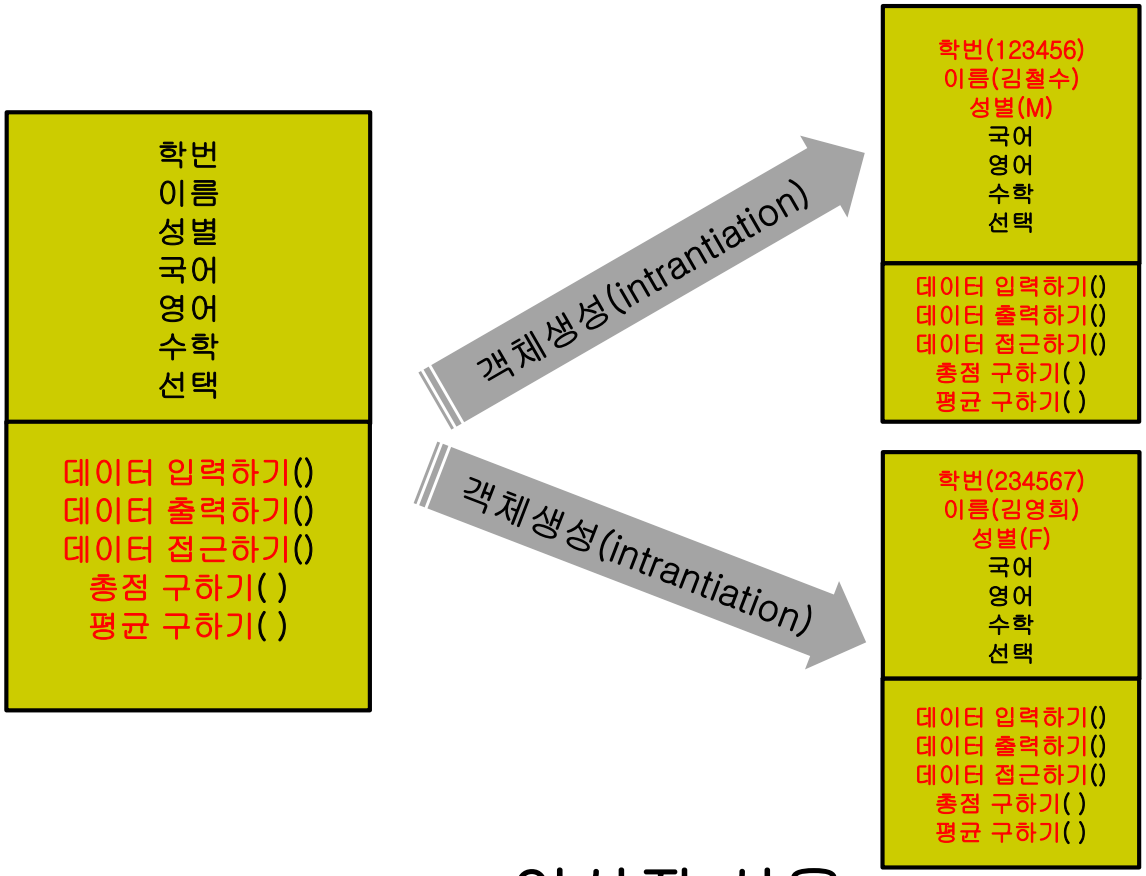
학생 성적처리(I)

```
    private float avg() {
        return  sum() / 4.0f;
    }

    private int sum( ) {
        return kor + math + eng + option;
    }

    @Override
    public String toString() {
        return String.format("이름 : %s₩n", name) +
                String.format("학번 : %s₩n", hakbun) +
                String.format("성별 : %s₩n", (gender == 'M' ? "남자" : "여자")) +
                String.format("국어 : %d, 수학 : %d, 영어 : %d, %s : %d₩n",
                        kor, math, eng, (gender == 'M' ? "기술" : "가정"), option) +
                String.format("총점 : %d, 평균 : %5.2f₩n", sum(), avg());
    }
}
```

동작(기능)은 Method로 구현

학생 성적처리(I)

- Student 클래스로부터 객체(instance)를 생성하는 예



new 연산자 사용

학생 성적처리(I)

- Main.JAVA

```
public static void main(String[] args) {
        Student hong = new Student();

        hong.hakbun = “2201234”;
        hong.name = "홍길동";
        hong.gender = ‘M’;
        hong.kor = 65;
        hong.math = 82;
        hong.eng = 71;
        hong.option = 100;

        System.out.print(hong);
}
```

객체지향프로그래밍(OOP)에서 권장하는 방법은 아님 -> Setter/getter (접근 제어자)

학생 성적처리(II)

- 생성자를 사용

```
public class Main {
    public static void main(String[] args) {
        Student man1 = new Student("2201234", "김철수", 'M', 70, 80,
                                                        80, 100);
        Student man2 = new Student("2201235", "김영희", 'F', 80, 90,
                                                        100, 100);

        man1.kor = 100;
        man2.kor = 90;

        System.out.println(man1);
        System.out.println(man2);
    }
}
```

man1 객체 생성

man2 객체 생성

생성된 객체의 속성에
서로 다른 값을 저장

학생 성적처리(III)

```
class Student {
        private String hakbun;
        private String name;
        private char gender;
        private int kor;
        private int math;
        private int eng;
        private int option;

        public Student() {      // 기본 생성자 , overloading

        }
```

상태 (속성)

Information hiding
Encapsulation

학생 성적처리(III)

```
    public Student(String hakbun, String name, char gender, int kor,
                                        int math, int eng, int option) {
        this.hakbun = hakbun;
        this.name = name;
        this.gender = gender;
        this.kor = kor;
        this.math = math;
        this.eng = eng;
        this.option = option;
}
```

생성자

학생 성적처리(III)

```
public void setHakbun(String hakbun) {
    this.hakbun = hakbun;
}

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

public void setKor(int kor) {
    this.kor = kor;
}

public void setMath(int math) {
    this.math = math;
}

public void setEng(int eng) {
    this.eng = eng;
}
```

Information hiding을
구현할 수 있게
해 주는 setter/getter

학생 성적처리(III)

```
    private float avg() {
        return sum() / 4.0f;
    }

    private int sum() {
        return kor + math + eng + option;
    }

    @Override
    public String toString() {
        return String.format("이름 : %s₩n", name) +
            String.format("학번 : %s₩n", hakbun) +
            String.format("성별 : %s₩n", (gender == 'M' ? "남자" : "여자")) +
            String.format("국어 : %d, 수학 : %d, 영어 : %d, %s : %d₩n",
                kor, math, eng, (gender == 'M' ? "기술" : "가정"), option) +
            String.format("총점 : %d, 평균 : %5.2f₩n", sum(), avg());
    }
}
```

학생 성적처리(III)

- Main.JAVA

```
public static void main(String[] args) {
    Student hong = new Student();
    hong.hakbun = "2201234";
    hong.name = "홍길동";
    hong.gender = 'M';
    hong.kor = 65;
    hong.math = 82;
    hong.eng = 71;
    hong.option = 100;

    System.out.print(hong);
}
```

학생 성적처리(IV)

- Main.JAVA

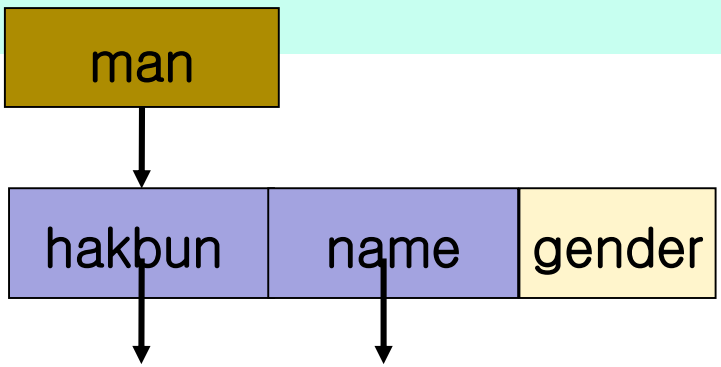
```
public static void main(String[] args) {
        Student hong = new Student("2201234", "홍길동", 'M', 65, 82,
                                                        71, 100);

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

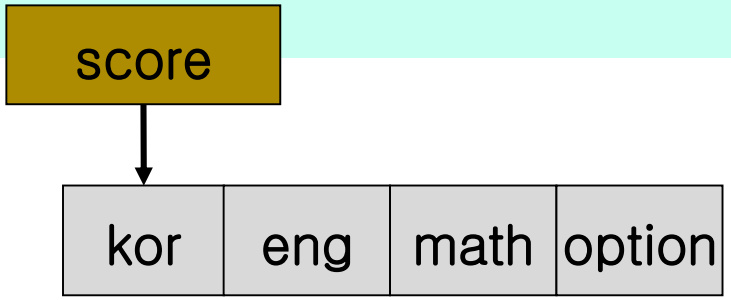
객체지향프로그래밍(OOP)에서 권장하는
방법 (생성자 이용)

학생 성적처리(V)

Student 클래스

```
public class Student {
    final String[] subject = {"국어", "영어", "수학", "선택"};
    private String[] man;
    private int[] score;

    public Student(String hakbun, String name, String gender) {
        man = new String[3];
        man[0] = hakbun;
        man[1] = name;
        man[2] = gender;
        score = new int[subject.length];
    }
```





학생 성적처리(V)

- this Reference
  - 클래스 내에서 자기 자신을 가리키는 Reference
  - Compiler에 의해 자동 관리, 개발자는 사용하기만 하면 됨
  - static으로 선언된 메소드에서는 사용할 수 없음
  - this.Member 형태로 Member를 접근할 때 사용

  - 2가지 목적으로 사용
    - 자기 자신의 Member Field나 메소드를 명확히 지시하기 위해 사용
    - 객체 전체를 함수의 매개 변수로 전달해야 하는 경우 사용

학생 성적처리(V)

- Student 클래스

```
public void input() throws IOException {
    Scanner keyboard = new Scanner(System.in);
    while (true) {
        for (int i = 0; i < subject.length; i++) {
            if (i == 3) {
                System.out.printf("%s 학생 %s 성적 입력 : ", man[1],
                                        (man[2].equals("M") ? "기술" : "가정"));
            } else {
                System.out.printf("%s 학생 %s 성적 입력 : ", man[1], subject[i]);
            }
            score[i] = keyboard.nextInt();
        }
```

학생 성적처리(V)

- Student 클래스

```
            boolean flag = false;
            for (int i = 0; i < subject.length; i++) {
                if (score[i] >= 0 && score[i] <= 100)
                    break;
                else {
                    flag = true;
                }
            }
            if (flag) {
                System.err.print("입력 오류");
                System.in.read();
            } else
                break;
        }
    }
```

학생 성적처리(V)

- Student 클래스

```
private float avg() {
    return sum() / 4.0f;
}

private int sum() {
    int total = 0;
    for (int i = 0; i < subject.length; i++)
        total += score[i];
    return total;
}
```

학생 성적처리(V)

- Student 클래스

```
    @Override
    public String toString() {
        return String.format("이름 : %s₩n", man[1]) +
                String.format("학번 : %s₩n", man[0]) +
                String.format("성별 : %s₩n", (man[2].equals("M") ? "남자" : "여자")) +
                String.format("국어 : %d, 수학 : %d, 영어 : %d, %s : %d₩n",
                        score[0], score[1], score[2],
                                (man[2].equals("M") ? "기술" : "가정"), score[3]) +
                String.format("총점 : %d, 평균 : %5.2f₩n", sum(), avg());
    }
}
```

학생 성적처리(V)

- Main 클래스

```
public static void main(String[] args) throws IOException {
    Student hong = new Student("123456", "홍길동", "M");

    hong.input();
    System.out.println(hong);
}
```

학생 성적[실습]

- 학생의 성적을 100점 만점의 Quiz를 본다.
- 단, Quiz는 여러 번 볼 수가 있음
- 이 Quiz 성적을 가지고 평균(avg)을 구하여 보자.

학생 성적[실습]

- Student Class
  - Member 변수
    - 학생의 이름(name)
    - 학번(hakbun)
    - 점수(quiz)는 ArrayList에 저장 되어야 함(0 ~ 100점)
  - 메소드
    - getName()
    - addQuiz(int score)
    - getTotalScore()
    - getNumQuizzes()
    - getAverageScore()

학생 성적[실습]

- Student 클래스

```
public class Student {
    private String name;
    private String hakbun;
    private ArrayList<Integer> quizs;

    public Student(){
        name = "";
        hakbun = "";
        quizs = new ArrayList<>();
    }

    public Student(String name, String hakbun) {
        this.name = name;
        this.hakbun = hakbun;
        quizs = new ArrayList<>();
    }
```

학생 성적[실습]

- Student 클래스

```
public void addQuiz(int jumsu)    {
    quizs.add(jumsu);
}

public int getQuiz(int index) {
    return quizs.get(index);
}

public int getTotalScore()    {
    int data = 0;
    for (int i = 0; i < quizs.size(); i++)
        data += quizs.get(i);
    return data;
}

public float getAverageScore()    {
    float avg = (float) getTotalScore() / quizs.size();
    return avg;
}
```

학생 성적[실습]

- Student 클래스

```
    public String toString() {
        String result = "";
        for (int i = 0; i < quizs.size(); i++)
            result += String.format("%d 회 : %d 점₩n", i+1, getQuiz(i));

        return String.format("이름 : %s, 학번 : %s₩n", name, hakbun) +
                String.format("총 %d 회 퀴즈 성적₩n", quizs.size()) +
                result +
                String.format("총점 : %d 점 ₩n", getTotalScore()) +
                String.format("평균 : %.2f 점 ₩n", getAverageScore());
    }
}
```

학생 성적[실습]

- Main 클래스

```
public class Main {
    public static void main(String[] args) throws IOException {
        Scanner keyboard = new Scanner(System.in);
        Student hong = new Student("홍길동", "1820978");
        int count;
        while (true) {
            System.out.print("퀴즈를 몇번 보았나요 ? ");
            count = keyboard.nextInt();
            if (count > 0)
                break;
            else {
                System.err.print("입력 오류");
                System.in.read();
            }
        }
```

학생 성적[실습]

- Main 클래스

```
        for (int i = 0; i < count; i++) {
            int jumsu;
            while (true) {
                System.out.printf("%d회 퀴즈 성적은 ", (i + 1));
                jumsu = keyboard.nextInt();
                if (jumsu >= 0 && jumsu <= 100)
                    break;
                else {
                    System.err.print("입력 오류");
                    System.in.read();
                }
            }
            hong.addQuiz(jumsu);
        }
        System.out.println(hong);
    }
}
```