JAVA 프로그램 실습

프로그램 따라하기

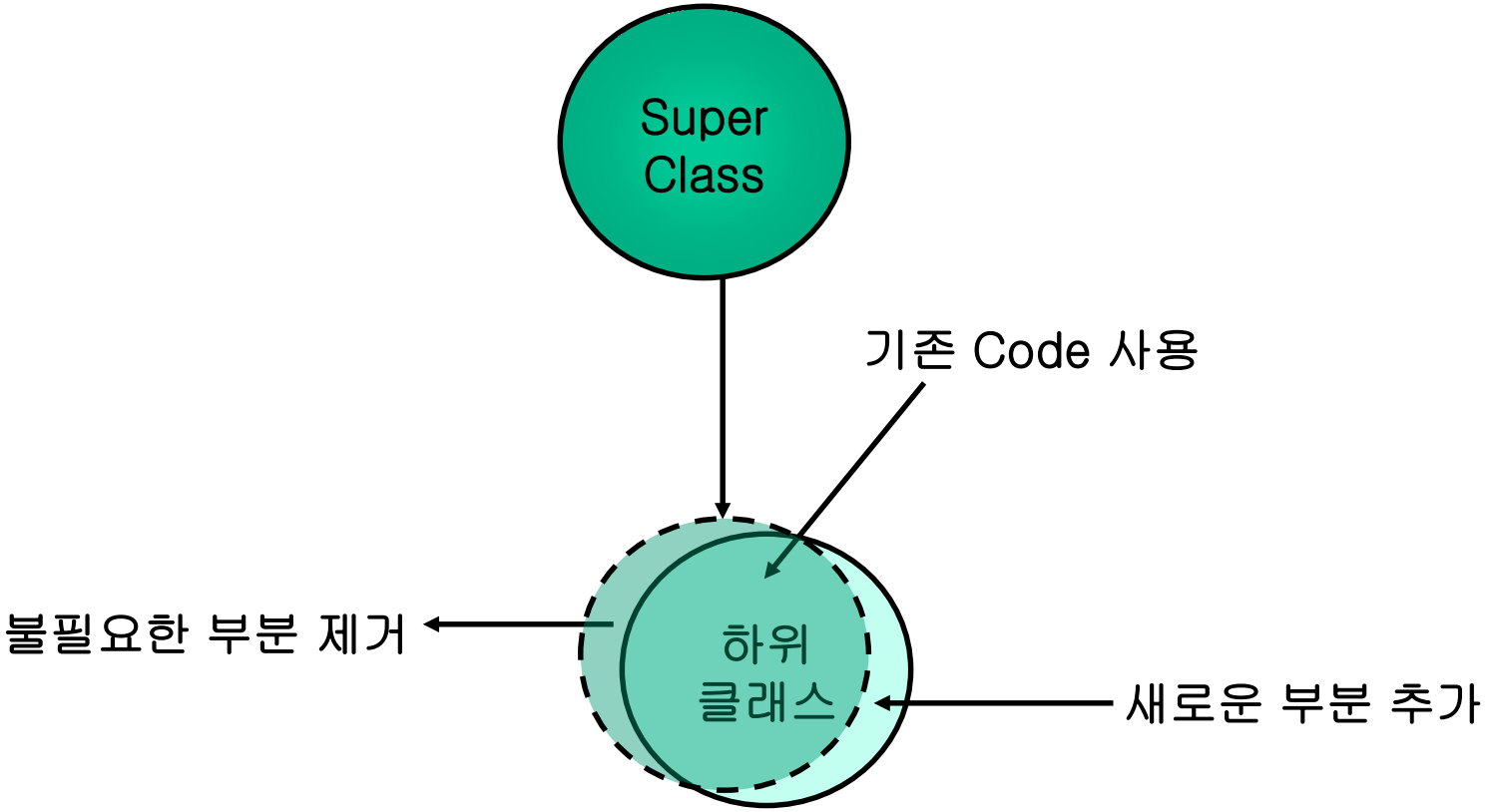
경복대학교
소프트웨어융합과
배희호 교수

Object Oriented Language의 특성

- Inheritance
  - 하나의 Class Object가 정의되었을 때, 차후 정의되는 어떠한 Sub Class라도 일반 Class들의 정의 중에서 하나 또는 그 이상의 정의를 물려받을 수 있다는 개념
  - Programmer에게 있어서 Inheritance이란 개념은, 만약 그것이 Sub Class가 속한 Class의 포괄적인 속성이라면, Sub Class 내의 하나의 Object는 자기 자신만의 Data 또는 Method에 관한 정의를 가지고 다닐 필요가 없다는 것을 의미
  - 이것은 Program 개발 속도를 높여줄 뿐 아니라, 정의된 Sub Class Object들에 대한 유효성이 본래부터 확실하다는 것을 보장

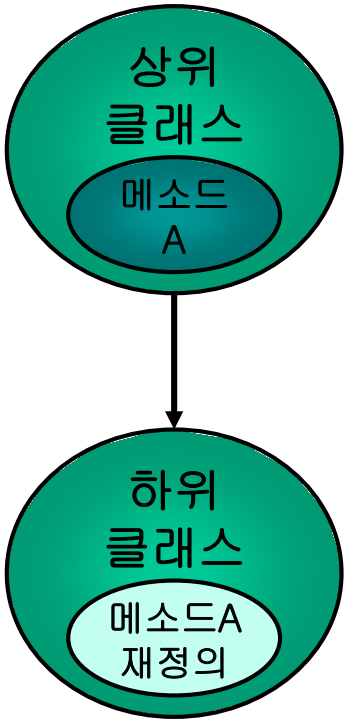
Object Oriented Language의 특성

- Inheritance



Object Oriented Language의 특성

- Overriding(재 정의)
  - Overriding은 Super Class로부터 물려 받은 Method를 동일한 형식으로 Sub Class에서 정의함으로써 Sub Class의 상황에 맞추어 다시 구현하는 것을 말함



Object Oriented Language의 특성

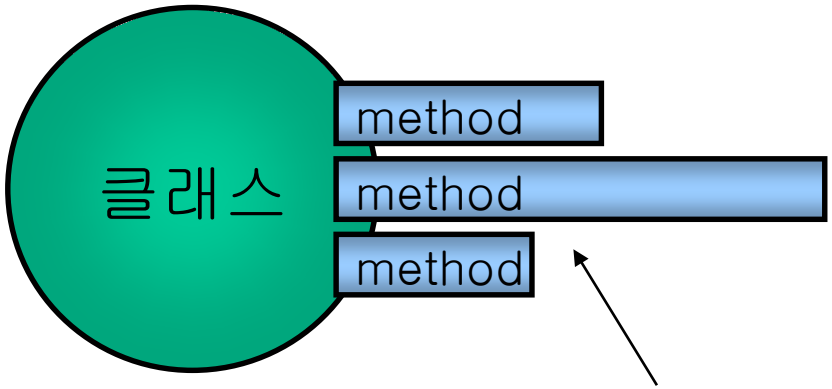
- Polymorphism(다형성)
  - 동일한 이름을 갖는 Method나 연산자가 다양한 문맥(context)에서 겉 모습은 그대로 유지하되 속 모습은 달리하면서 사용될 수 있도록 하는 강력한 기법
  - 예) JAVA에서 1+1이라는 명령문 안에 있는 '+'는 2개의 정수를 더하는 덧셈 연산자이지만 "abc"+"def"라는 명령문 안에 있는 '+'는 "abc"와 "def"라는 2개의 문자열을 서로 연결(concatenation)하라는 문자열 연산자 임
  - '+'라는 기호의 겉모습은 유지되고 있지만, 그 연산자가 실제로 수행하는 연산의 내용은 완전히 다름

Object Oriented Language의 특성

- Polymorphism(다형성)
  - 다형성이 지원되지 않는 조건에서 '+' 기호를 앞에서와 마찬가지로 여러 가지 목적에 사용한다고 가정하자. 이러한 경우에는 함수에 전달된 인수(parameter)의 Data Type을 확인해서 그것이 '정수'면 '더하기'를 수행하는 함수를 호출하고, '문자열'이면 '문자열 연결'을 수행하는 함수를 호출하는 조건문을 Programmer 스스로 작성해야 함
  - JAVA를 사용해서 Programming을 할 때 Method의 이름은 동일하게 유지하면서 입력되는 인수의 수나 Data Type만 살짝 바꿔주는(오버로드(overload)라고 불림) 경우가 있는데, 이러한 방법이 가능한 이유는 바로 JAVA가 다형성을 지원해 주는 '객체지향' 언어이기 때문임

Object Oriented Language의 특성

- Polymorphism(다형성)



이름은 같으나 인자가 다른 메소드

Object Oriented Modeling 절차

- 문제 이해
  - 문제 영역의 대상들을 파악(Object별 정보와 Data 파악)
- 문제 영역에서 Class 식별해 내기
  - 명사들을 파악, 일반화 및 추상화
- Class의 행위(Method), 속성 지정
  - 동사들을 파악, Method 파악
  - Class 추상화
- Class간의 관계를 식별
  - 상속 관계(Super, Sub), 포함 관계
- Class 구현
  - Field 생성
  - 생성자 구현
  - Setter()와 Getter() 구현
  - Method 구현

전기 요금

- 입력으로 사용자 번호, 이름, 전기 사용량을 kw단위로 입력 받은 후, 아래의 처리조건에 의하여 전기요금을 계산하는 프로그램을 작성하시오

전기 요금

- 처리 조건
  - 데이터는 최소 10개
  - 사용자 번호는 5자리 숫자코드로 구성
  - 사용량은 정수형임
  - 기본 요금 : 1,660원
  - 사용 요금 = 기본 요금 +(사용량 * kw당 사용 요금)
  - 세금 : 사용 요금의 7%
  - 납부 요금 = 사용 요금 + 세금
  - 지원 가구는 매달 100Kw까지는 무료로 사용
  - 지원 가구는 사용자 번호가 '9'로 시작함
  - 출력 시 비고란에 지원 가구는 '지원'이라고 출력
  - 출력 시 전기 요금이 많은 순서로 출력 (보너스 점수)

전기 요금

- Kw당 사용 요금 조견표

전력 사용량	사용량 요금
100kw이하	184.1원
100kw초과 200kw이하	223.8원
200kw초과 300kw이하	278.3원
300kw초과 400kw이하	353.6원
400kw초과 500kw이하	466.4원
500kw초과	643.9원

전기 요금

- 일반 가구에서 120kW의 전력을 사용하면
  - 기본요금 : 1,660원
  - 100kW 이상을 사용하였으므로 100kW까지는 184.1원
    = 100 × 184.1 = 18,410원
  - 나머지 20kW까지의 요금은 223.8원
    = 20 × 223.8 = 4,476원

  - 전체 요금 = 기본요금 + 사용요금
    = 1,660 + (18,410 + 4,476) = 24,546 원

  - 세금 (7%)
    = 24,546 * 7 / 100 = 1,718.22원
  - 최종 사용 요금 = 24.546원 + 1,718원 = 26,264원

전기 요금

- 지원 가구에서 120kW의 전력을 사용하면
  - 기본요금 : 1660원
  - 100kW 이상을 사용하였으므로 100kW까지는 무료
    = 100 × 0 = 0원
  - 나머지 20kW까지의 요금은 181.4원
    = 20 × 184.1원 = 3,682원

  - 전체 요금 = 기본요금 + 사용요금
    = 1,660원 + (0 + 3,682원) = 5,342원

  - 세금 (7%)
    = 5,342 * 7 / 100 = 373.94 원
  - 최종 사용 요금 = 5,342 + 373 = 5,715원

전기요금

- if문 사용
- switch ~ case 사용
- charat() 메소드

전기요금

- Class Diagram

Man
속성 : 사용자 번호, 이름 행동 : 출력하기

추상화

Customer
속성 : 사용자 번호, 이름, 사용량 행동 : 요금계산, 세금계산, 납부금액, 출력하기

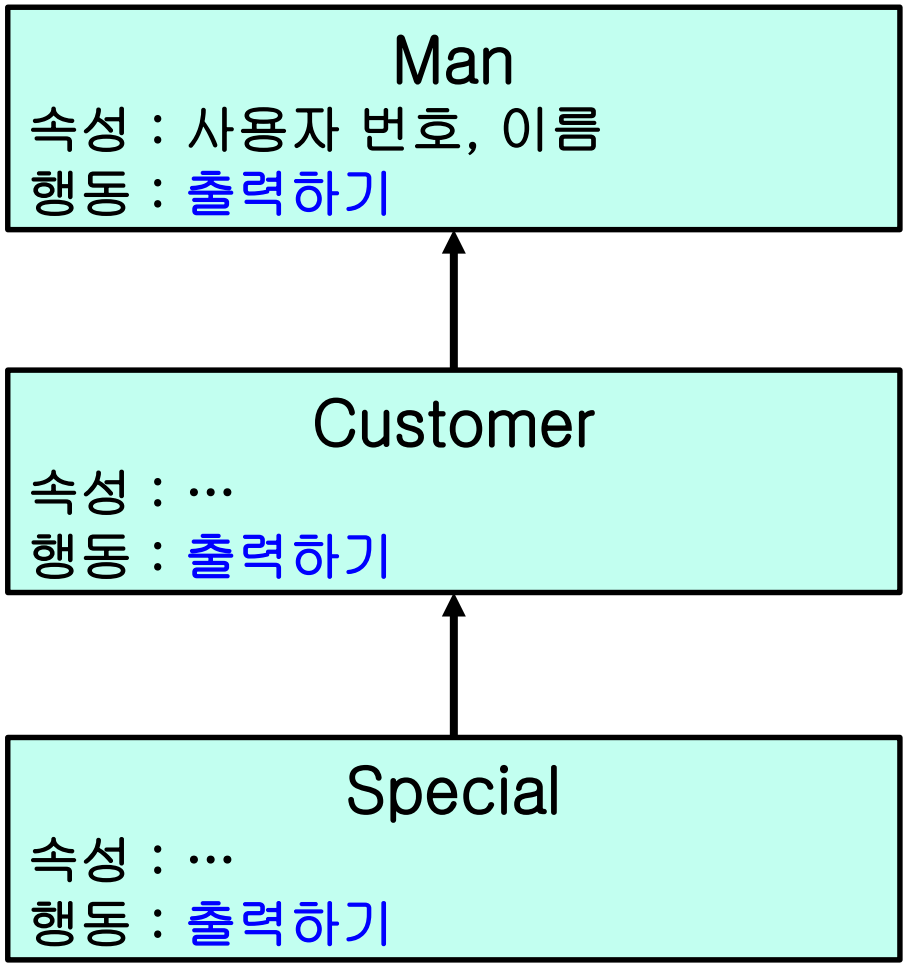
추상화

Special
속성 : 사용자 번호, 이름, 사용량 행동 : 요금계산, 세금계산, 납부금액, 출력하기

추상화

전기요금

- Class Diagram



추상화

전기요금

실행 결과

```
일본인 고객의 사용량 입력 : 567
이대한 고객의 사용량 입력 : 456
*************************************************************
  번호     이름   사용량    사용요금     세금       납부금액     기타
*************************************************************
  12345   이니나 890Kw   403,400원   28,238원   431,638원
  12345   윤상열 785Kw   335,790원   23,505원   359,295원
  12345   정상진 678Kw   266,893원   18,682원   285,575원
  12345   한국인 678Kw   266,893원   18,682원   285,575원
  92345   이기동 678Kw   202,503원   14,175원   216,678원   지원가구
  12345   고길동 567Kw   195,420원   13,679원   209,099원
  92345   일본인 567Kw   136,887원    9,582원   146,469원   지원가구
  12345   이대한 456Kw   131,757원    9,222원   140,979원
  12345   홍길동 345Kw    86,191원    6,033원    92,224원
  92345   박종호 345Kw    54,973원    3,848원    58,821원   지원가구
*************************************************************
```

전기요금

- Man 클래스

```
public class Man {
    final private String bunho;
    final private String name;   // 데이터 (생성자 초기화)

    public Man(String bunho, String name) {
        this.bunho = bunho;
        this.name = name;
    }

    public String getName() {
        return name;
    }

    @Override
    public String toString() {
        return String.format(" %6s %4s", bunho, name);
    }
}
```

전기요금

- Man을 상속받은 Customer 클래스

```
public class Customer extends Man {
    protected int used;      // 입력 메소드

    public Customer(String bunho, String name) {
        super(bunho, name);
    }
```

전기요금

- Man을 상속받은 Customer 클래스

```
public void input() {
    Scanner keyboard = new Scanner(System.in);

    while (true) {
        System.out.printf("%s 고객의 사용량 입력 : ", getName());
        used = keyboard.nextInt();
        if (used < 0 || used > 999) {
            System.err.println("사용량 오류");
            try {
                System.in.read();
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }else
            break;
    }
}
```

전기요금

- Man을 상속받은 Customer 클래스

```
public int fee() { // 사용요금 계산하기 메소드
    int fee;
    final int basic = 1660;
    final float[] table = {184.1f, 223.8f, 278.3f, 353.6f, 466.4f, 643.9f};

    if (used <= 100) {
        fee = (int) (used * table[0]) + basic;
    } else if (used <= 200) {
        fee = (int) (100 * table[0]) + basic;
        fee += (int) ((used - 100) * table[1]);
    } else if (used <= 300) {
        fee = (int) (100 * table[0]) + basic;
        fee += (int) (100 * table[1]);
        fee += (int) ((used - 200) * table[2]);
    } else if (used <= 400) {
        fee = (int) (100 * table[0]) + basic;
        fee += (int) (100 * table[1]);
        fee += (int) (100 * table[2]);
        fee += (int) ((used - 300) * table[3]);
```

전기요금

- Man을 상속받은 Customer 클래스

```
        } else if (used <= 500) {
            fee = (int) (100 * table[0]) + basic;
            fee += (int) (100 * table[1]);
            fee += (int) (100 * table[2]);
            fee += (int) (100 * table[3]);
            fee += (int) ((used - 400) * table[4]);
        } else {
            fee = (int) (100 * table[0]) + basic;
            fee += (int) (100 * table[1]);
            fee += (int) (100 * table[2]);
            fee += (int) (100 * table[3]);
            fee += (int) (100 * table[4]);
            fee += (int) ((used - 500) * table[5]);
        }
        return fee;
    }
```

전기요금

- Man을 상속받은 Customer 클래스

```
    public int tax() {
        return  (int)(fee() * 7 / 100.0f);
    }

    @Override
    public String toString() {
        return super.toString() + String.format(" %3dKw %,8d %,7d %,9d",
                used, fee(), tax(), fee() + tax() );
    }
}
```

전기요금

- Customer를 상속받은 Special 클래스

```
public class Special extends Customer {

    public Special(String bunho, String name) {
        super(bunho, name);
    }
```

전기요금

- Customer를 상속받은 Special 클래스

```
@Override
public int fee() {
    int fee;
    final int basic = 1660;
    final float[] table = {184.1f, 223.8f, 278.3f, 353.6f, 466.4f, 643.9f};
    int temp = used - 100;
    if (temp <= 100) {
        fee = (int) (temp * table[0]) + basic;
    } else if (temp <= 200) {
        fee = (int) (100 * table[0]) + basic;
        fee += (int) ((temp - 100) * table[1]);
    } else if (temp <= 300) {
        fee = (int) (100 * table[0]) + basic;
        fee += (int) (100 * table[1]);
        fee += (int) ((temp - 200) * table[2]);
    } else if (temp <= 400) {
        fee = (int) (100 * table[0]) + basic;
        fee += (int) (100 * table[1]);
```

전기요금

- Customer를 상속받은 Special 클래스

```
            fee += (int) (100 * table[2]);
            fee += (int) ((temp - 300) * table[3]);
        } else if (temp <= 500) {
            fee = (int) (100 * table[0]) + basic;
            fee += (int) (100 * table[1]);
            fee += (int) (100 * table[2]);
            fee += (int) (100 * table[3]);
            fee += (int) ((temp - 400) * table[4]);
        } else {
            fee = (int) (100 * table[0]) + basic;
            fee += (int) (100 * table[1]);
            fee += (int) (100 * table[2]);
            fee += (int) (100 * table[3]);
            fee += (int) (100 * table[4]);
            fee += (int) ((temp - 500) * table[5]);
        }
        return fee;
    }
```

전기요금

- Customer를 상속받은 Special 클래스

```
    @Override
    public String toString() {
        return super.toString() + "  지원가구";
    }
}
```

전기요금

- PowerOffice 클래스

```
public class PowerOffice {
    private Customer[] customers;

    public PowerOffice(Customer[] customers) {
        this.customers = customers;
    }
    public void sort() {
        Customer temp;
        for (int i = 0; i < customers.length - 1; i++) {
            for (int j = i + 1; j < customers.length; j++) {
                if ((customers[i].tax() + customers[i].fee()) <
                        (customers[j].tax() + customers[j].fee())) {
                    temp = customers[i];
                    customers[i] = customers[j];
                    customers[j] = temp;
                }
            }
        }
    }
```

전기요금

- PowerOffice 클래스

```
public void display() {
    sort();
    line();
    System.out.println("  번호    이름  사용량   사용요금    세금     납부금액   기타 ");
    line();
    for (int i = 0; i < customers.length; i++)
        System.out.println(customers[i]);
    line();
}

private static void line() {
    System.out.println("-----------------------------------------------------------");
}
}
```

전기요금

- Main 클래스

```
public static void main(String[] args) throws IOException {
    Customer[] customers = new Customer[]{
            new Customer("12345","홍길동"),
            new Special("92345", "이기동"),
            new Customer("12345", "정상진"),
            new Customer("12345", "윤상열"),
            new Customer("12345", "고길동"),
            new Special("92345", "박종호"),
            new Customer("12345", "이니나"),
            new Customer("12345", "한국인"),
            new Special("92345", "일본인"),
            new Customer("12345", "이대한")};
```

전기요금

- Main 클래스

```
        for (int i = 0; i < customers.length; i++)
            customers[i].input();

        PowerOffice powerOffice = new PowerOffice(customers);
        powerOffice.display();
    }
```

전기요금(ArrayList)

- PowerOffice 클래스

```
public class PowerOffice {
    private ArrayList<Customer> customers;

    public PowerOffice(ArrayList<Customer> customers) {
        this.customers = customers;
    }

    public void sort() {
        Collections.sort(customers, new SumComparator().reversed());
    }

    private class SumComparator implements Comparator<Customer> {
        @Override
        public int compare(Customer o1, Customer o2) {
            return (o1.fee() + o1.tax()) - (o2.fee() + o2.tax());
        }
    }
```

전기요금(ArrayList)

- PowerOffice 클래스

```
public void display() {
    sort();
    line();
    System.out.println("  번호    이름  사용량   사용요금    세금     납부금액   기타 ");
    line();
    for (int i = 0; i < customers.size(); i++)
        System.out.println(customers.get(i));
    line();
}

private static void line() {
    System.out.println("**************************************************");
}
}
```

전기요금(ArrayList)

- Main 클래스

```
public static void main(String[] args) throws IOException {
    ArrayList<Customer> customers = new ArrayList<>();
    customers.add(new Customer("12345", "홍길동"));
    customers.add(new Special("92345", "이기동"));
    customers.add(new Customer("12345", "정상진"));
    customers.add(new Customer("12345", "윤상열"));
    customers.add(new Customer("12345", "고길동"));
    customers.add(new Special("92345", "박종호"));
    customers.add(new Customer("12345", "이니나"));
    customers.add(new Customer("12345", "한국인"));
    customers.add(new Special("92345", "일본인"));
    customers.add(new Customer("12345", "이대한"));
```

전기요금(ArrayList)

- Main 클래스

```
        for (int i = 0; i < customers.size(); i++)
            customers.get(i).input();

        PowerOffice powerOffice = new PowerOffice(customers);
        powerOffice.display();
    }
```