

객체지향프로그래밍 – 4차 실습 활동지

이름 : 김태석


학번 : 201721083

학과 : 미디어학과

가)

가) Employee 클래스를 테스트 하는 프로그램이다. 아래 코드를 수행한 결과가 무엇이고 어떻게 나왔는지 설명하시오. 입력데이터로 각각 2020, 4, 13을 제공한다. [수행결과 및 설명 포함] (1점)

```
public class EmployeeTest
{
    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);
        int yy = in.nextInt();
        int mm = in.nextInt();
        int dd = in.nextInt();
        Employee e = new Employee("Hong", 100000.0, yy, mm, dd); // (A)
        printEmployee(e);
    }
    public static void printEmployee(Employee e)
    {
        System.out.println("Name: "+e.getName());
        System.out.println("Salary: "+e.getSalary());
        System.out.println("Hireday: "+e.getHireDay().toString());
    }
}
```

수행결과 : 

스캐너를 통해 년도를 입력 받으면 Employee 객체가 생성되면서 그 안에 날짜 값을 받게 된다. 그 후 printEmployee 메소드를 실행해주어서 이름, 연봉, 고용일을 출력하게 된다.

나)

- 나) (가) 코드의 (A) 부분을 아래와 같이 변경하고자 한다. 이러한 방식으로 Employee객체를 생성하기 위해 Employee 클래스에 알맞은 constructor를 추가하시오. 수행 결과는? [constructor 코드 및 수행 결과 포함] (3점)

```
Date d = new Date(yy, mm, dd);  
Employee e = new Employee("Hong", 100000.0, d);
```

코드 추가 : public Employee(String n, double s, Date d){

name = n;

salary = s;

hireDay=d;

}

수행 결과 :

```
<terminated> w4 (1) [Java Application] C:\Program Files\Java\jdk-11.0.6\bin\javaw.  
START  
2020  
4  
13  
Name: Hong  
Salary: 100000.0  
Hireday: 2020:4:13
```

다)

- 다) (나) 코드에서 yy, mm, dd를 잘못된 날짜 가령, 2020, 4, 38을 입력하여 수행해보자. 아무런 문제를 발생시키지 않고 정상적으로 수행한다. 입력 데이터(날짜)의 유효성을 체크하기 위해 Date 클래스의 constructor를 수정하시오. 이 과정에서 어떤 문제가 발생하는지 설명하시오. (윤년을 검사하기 위해 java.util.GregorianCalendar class의 isLeapYear() method를 이용해도 좋다.) [수행결과 및 수정한 코드부분, 설명 포함] (3점)

수행 결과 :

```
START
2020
4
38
Error!
2020
4
13
Name: Hong
Salary: 100000.0
Hireday: 2020:4:13
```

수정한 코드부분 :

```
Scanner in = new Scanner(System.in);
GregorianCalendar gregori = new GregorianCalendar();
int End = 0;
while(End < 1){
    if(gregori.isLeapYear(yy)) {
        if((mm==1) || (mm==3) || (mm==5) || (mm==7) || (mm==8) || (mm==10) || (mm==12)){
            if(dd > 31) {
                System.out.println("Error!");
                yy = in.nextInt();
                mm = in.nextInt();
                dd = in.nextInt();
            }
            else {
                year = yy;
                month = mm;
                day = dd;
                End = 1;
            }
        }
        else if((mm==4) || (mm==6) || (mm==9) || (mm==11)) {
            if(dd > 30) {
                System.out.println("Error!");
                yy = in.nextInt();
                mm = in.nextInt();
                dd = in.nextInt();
            }
            else {
                year = yy;
                month = mm;
                day = dd;
                End = 1;
            }
        }
        else if(mm==2) {
            if(dd > 29) {
                System.out.println("Error!");
                yy = in.nextInt();
                mm = in.nextInt();
                dd = in.nextInt();
            }
            else {
                year = yy;
                month = mm;
            }
        }
    }
}
```

```

        day =dd;
        End = 1;
    }
}
}
else {
    if((mm==1)|| (mm==3)|| (mm==5)|| (mm==7)|| (mm==9)|| (mm==11)) {
        if(dd>31) {
            System.out.println("Error!");
            yy = in.nextInt();
            mm = in.nextInt();
            dd = in. nextInt();
        }
        else {
            year = yy;
            month =mm;
            day =dd;
            End = 1;
        }
    }
    else if ((mm==4)|| (mm==6)|| (mm==8)|| (mm==10)|| (mm==12)) {
        if(dd>30) {
            System.out.println("Error!");
            yy = in.nextInt();
            mm = in.nextInt();
            dd = in. nextInt();
        }
        else {
            year = yy;
            month =mm;
            day =dd;
            End = 1;
        }
    }
}
else if(mm==2) {
    if(dd>28) {
        System.out.println("Error!");
        yy = in.nextInt();
        mm = in.nextInt();
        dd = in. nextInt();
    }
    else {
        year = yy;
        month =mm;
        day =dd;
        End = 1;
    }
}
else {
    System.out.println("Error!");
    yy = in.nextInt();
    mm = in.nextInt();
    dd = in. nextInt();
}
}
}
}

```

설명 : 윤년의 규칙성을 이용하여 날짜 유효성 검사를 해 보았다.(윤년일 경우 2월은 29일까지 있으며, 1 3 5 7 8 10 12월이 31일 까지 존재한다. 윤년이 아닐 경우 2월은 28일까지 있으며, 1 3 5 7 9 11 월이 31일 까지 존재한다.)

java.util.GregorianCalendar의 isLeapYear 메소드를 사용해서 윤년인지 아닌지를 판별하고, 입력받은 날짜가 유효한지를 판별하였다. 유효한 날짜인 경우 객체를 생성할 수 있도록 넘어갈 수 있게 했지만, 유효하지 않은 날짜인 경우 Error 메시지와 함께 값을 다시 입력받도록 하였다. (유효한 값을 입력할 때 까지 계속해서 반복문이 돌게 하였다.)

유효성 검사를 진행하는데는 문제가 없었지만, Date 클래스의 constructor이 너무 길어지고, 생성자가 갖고있는 변수들을 한눈에 알아보기가 어려워졌다.

라)

라) (다)의 문제를 해결하기 위해 유효성 체크를 하는 static method(가령, isValid())를 Date 클래스에 추가하고, Date 객체를 생성하기 전에 입력데이터의 유효성을 체크한 후 유효한 데이터라면 Date 객체를 생성하고, 유효하지 않은 데이터라면 유효한 데이터가 입력될 때까지 다시 입력 받아 Date 객체를 생성하도록 (B) 부분을 적절하게 채우시오. 정상적인 데이터와 유효하지 않은 데이터 두가지에 대해 수행하시오. ((가)의 데이터 입력부분을 포함하여 수정함)

[수정한 코드부분, 수행결과 포함] (3점)

(B)

```
Date d = new Date(yy, mm, dd);
Employee e = new Employee("Hong", 100000.0, d);
```

```
// in Date class
public static boolean isValid(int yy, int mm, int dd) { ... }
```

수행 결과 :

```
START
2020
4
38
Date Validation Start!
Not Valid
Re-Enter!
2020
4
13
Date Validation Start!
Valid.
Name: Hong
Salary: 100000.0
Hireday: 2020:4:13
```

추가한 static method :

```
public static boolean DateCheck(int a, int b, int c) {
    System.out.println("Date Validation Start!");
    GregorianCalendar gregori = new GregorianCalendar();
    if(gregori.isLeapYear(a)) {
        if((b==1)|| (b==3)|| (b==5)|| (b==7)|| (b==8)|| (b==10)|| (b==12)){
            if(c>31) {
                return false;
            }
            else {
                return true;
            }
        }
        else if((b==4)|| (b==6)|| (b==9)|| (b==11)) {
            if(c>30) {
                return false;
            }
            else {
                return true;
            }
        }
        else if(b==2) {
            if(c>29) {
                return false;
            }
            else {
                return true;
            }
        }
    }
    else {
        if((b==1)|| (b==3)|| (b==5)|| (b==7)|| (b==9)|| (b==11)) {
            if(c>31) {
                return false;
            }
            else {
                return true;
            }
        }
        else if ((b==4)|| (b==6)|| (b==8)|| (b==10)|| (b==12)) {
            if(c>30) {
                return false;
            }
            else {
                return true;
            }
        }
        else if(b==2) {
            if(c>28) {
                return false;
            }
            else {
                return true;
            }
        }
        else {
            return false;
        }
    }
}
```

```

    }
    return false;
}

```

수정한 코드 부분 :

```

while(End<1) {
    if(Date.DateCheck(yy, mm, dd)) {
        System.out.println("Valid.");
        End =1;
    }
    else {
        System.out.println("Not Valid");
        System.out.println("Re-Enter!");
        yy = in.nextInt();
        mm = in.nextInt();
        dd=in.nextInt();
    }
}
}

```

설명 : DateCheck method를 만들어서 날짜가 유효하다면 true 값을, 유효하지않다면 false값을 return 하도록 하였다. 이를 통해 Employeetest class에 유효한 날짜가 입력 될 때 까지 값을 재입력 받는 반복문을 하나 추가 해 주었다. DateCheck method도 다)에서와 마찬가지로 윤년의 규칙성을 이용하는 방식으로 만들었고, 똑같이 java.util.GregorianCalendar을 import해주었다.

마) Employee클래스의 instance variable 중 name과 hireDay의 선언에서 final은 무슨 의미인지 설명하시오.[설명 포함](1점)

Final을 선언하게 되면, 값이 저장되었을 때 변경이 불가능함을 의미한다. 하지만 final 변수가 객체를 가졌을 때에는 객체 내부가 수정 될 수 있다.

바)

바) (가) 코드 (A) 부분을 다음과 같이 변경하여 수행하여 그 결과를 본 후 (마)의 관점에서 어떤 문제가 발생했는지 또 그 이유가 무엇인지 설명하시오. [수행결과 및 설명 포함] (2점)

```
Date d = new Date(yy, mm, dd); // 2020, 4, 13으로 입력
Employee e = new Employee("Hong", 100000.0, d);
printEmployee(e);
Date d2 = e.getHireday();
d2.setDate(new Date(2020, 3, 15));
```

수행 결과 :

```
START
2020
4
13
Date Validation Start!
Valid.
Name: Hong
Salary: 100000.0
Hireday: 2020:4:13
Name: Hong
Salary: 100000.0
Hireday: 2020:3:15
```

(마)의 관점에서의 문제점 과 그 이유 : 마)에서는 final 선언을 통하여 hireday의 값이 변하지 않도록 하는걸 목표로했다. 하지만. 바)의 실행결과에서 보이듯 Employee e의 객체는 그대로인데 여기의 hireday의 값이 setDate를 통해 바뀐 것을 볼 수 있다. Final 선언을 해 주었는데 값이 바뀐 이유는, hireday가 객체를 갖게 되어서 그 객체의 내부가 수정된 것이기 때문이다.

사) Date class에서 getYear(), getMonth(), getDay() 메소드를 모두 제거한다면 setDate() 메소드를 어떻게 수정할 수 있겠는가? 수정한 후 (바)의 코드를 수행해보시오. [setDate() 코드 및 수행 결과 포함] (2점)

setDate()코드 :

```
public void setDate(Date d)
{
    this.year = d.year;
    this.month = d.month;
    this.day = d.day;
}
```

수행 결과 :

```
START
2020
4
13
Date Validation Start!
Valid.
Name: Hong
Salary: 100000.0
Hireday: 2020:4:13
Name: Hong
Salary: 100000.0
Hireday: 2020:3:15
```