Report_Chap08

Date: 2022. 05. 10

Name: 송예지

Student ID: 22100396

(8 장) Page 703. #6

For this Programming Project, start with implementations of the Person, Student, and Undergraduate classes as depicted in Figure 8.4 and the polymorphism demo in Listing 8.6. Define the Employee, Faculty, and Staff classes as depicted in Figure 8.2. The Employee class should have instance variables to store the employee ID as an int and the employee's department as a String. The Faculty class should have an instance variable to store the faculty member's title (e.g. "Professor of Computer Science") as a String. The Staff class should have an instance variable to store the staff member's pay grade (a number from 1 to 20) as an int. Every class should have appropriate constructors, accessors, and mutators, along with a writeOutput method that outputs all of the instance variable values.

Modify the program in Listing 8.6 to include at least one Faculty object and at least one Staff object in addition to the Undergraduate and Student objects. Without modification to the for loop, the report should output the name, employee ID, department, and title for the Faculty objects, and the name, employee ID, department, and pay grade for the Staff objects.

이 프로그래밍 프로젝트의 경우 그림 8.4 에 표시된 사람, 학생 및 학부 수업의 구현과 목록 8.6 의 다형성 데모부터 시작합니다. 그림 8.2 와 같이 직원, 교직원 및 직원 클래스를 정의합니다. Employee 클래스에는 직원 ID 를 int 로 저장하고 직원의 부서를 String 으로 저장할 인스턴스 변수가 있어야 합니다. 교수 클래스에는 교직원의 직함(예: "컴퓨터 과학 교수")을 문자열로 저장할 인스턴스 변수가 있어야 합니다. 직원 클래스에는 직원의 급여 등급(1 에서 20 까지의 숫자)을 int 로 저장할 인스턴스 변수가 있어야 합니다. 모든 클래스에는 적절한 생성자, 액세스자 및 돌연변이가 있어야 합니다.모든 인스턴스 변수 값을 출력하는 출력 방법입니다.
Listing 8.6 의 프로그램을 수정하여 학부 및 학생 객체 외에 하나 이상의 교수진 객체 및 하나 이상의 직원 객체를 포함합니다. 포루프를 수정하지 않고 리포트는 교수진 객체의 이름, 직원 ID, 부서 및 직함을 출력하고 직원 객체의 이름, 직원 ID, 부서 및 급여 등급을 출력해야 합니다.

기본 문제 해결

- ⇨ method 를 작성하기 위한 7 개의 class 를 생성한다.
- ⇨ Driver program 을 작성하기 위해 PolymorphismDeomo class 도 생성해준다.
- ⇨ 문제에서 주어진 대로 Parent class 와 child class 를 extends 시켜준다.
- ⇨ 각각의 class 에서 각각의 method 코드 구현
    - Constructor: object 생성시 instance variable 초기화 (same with class name)
    - Accessor method: instance variable 을 return (get)
    - Mutator method: instance variable 수정 (set)
    - writeOutput method: 결과 출력
    - equals method: 다른 object 와 같은 값이면 true return

실습 코드

PolymorphismDemo Code

```
public class PolymorphismDemo {
	public static void main(String[] args) {
		Person[] people = new Person[7];

		people[0] = new Undergraduate("Cotty, Manny", 4910, 1);
		people[1] = new Undergraduate("Kick, Anita", 9931, 2);
		people[2] = new Student("DeBanque, Robin", 8812);
		people[3] = new Undergraduate("Bugg, June", 9901, 4);

		people[4] = new Employee2("Ken,Bill", "Dept of Computer Science", 3056);
		people[5] = new Faculty("Mock, Kenrick", "Computer Science", 3056, "Prefessor of Computer Science");
		people[6] = new Staff("Jones, James", "Mathematical Sciences", 1551, 15);

		for(Person p : people) {
			p.writeOutput();
			System.out.println();
		}
	}
}
```

Person code

```
public class Person {
	private String name;

	//constructor
	public Person() {
		name = "No name yet";
	}
	public Person(String n) {
		name = n;
	}
	// accessor
	public String getName() {
		return name;
	}
	// mutator
	public void setName(String newName) {
		name = newName;
	}
```

```
	public void writeOutput() {
		System.out.println("Name: "+name);
	}
	public boolean hasSameName(Person otherPerson) {    // equals method
		if( name.equals(otherPerson.name)) return true;
		else return false;
	}
	public String toString() {
		return ("Name: "+name);
	}
}
```

Student code

```
public class Student extends Person{
	private int id;       // student number

	// contructors
	public Student() {
		super();          // parent class 안에 있는 instance variables 초기
		id = 0;
	}
	public Student(String s, int n) {
		super(s);
		id = n;
	}

	// accessor
	public int getStudentNumber() {
		return id;
	}
	// mutator
	public void setStudentNumber(int newStudentNumber) {
		id = newStudentNumber;
	}

	public void reset(String newName, int newStudentNumber) {
		setName(newName);
		id = newStudentNumber;
	}
	public void writeOutput() {
```

```
		System.out.println("Name: " + getName());
		System.out.println("Student Number: "+id);
	}
	public boolean equals(Student otherStudent) {
		if((this.hasSameName(otherStudent))&& (id== otherStudent.id))
			return true;
		else return false;
	}
	public String toString() {
		return("Name: "+getName()
				+ "\nStudent number: "+id);
	}
}
```

Undergraduate code

```
public class Undergraduate extends Student{
	private int level;

	// constructor
	public Undergraduate(String s, int n1, int n2) {
		super(s, n1);
		level = n2;
	}
	// accessor
	public int getLevel() {
		return level;
	}
	// mutator
	public void setLevel(int newLevel) {
		level = newLevel;
	}

	public void reset(String newName, int newStudentNumber, int newlevel) {
		setName(newName);
		setStudentNumber(newStudentNumber);
		level = newlevel;
	}
	public void writeOutput() {
		System.out.println("Name: "+ getName());
		System.out.println("Student Number: "+ getStudentNumber());
		System.out.println("Student Level: "+level);
```

```
	}
	public boolean equals(Undergraduate otherUndergraduate) {
		if(	this.equals(otherUndergraduate)&&
				level== otherUndergraduate.level)
			return true;
		else return false;
	}
	public String toString() {
		return("Name: "+getName()
				+ "\nStudent number: "+getStudentNumber()
				+ "\nStudent Level: "+level);
	}
}
```

Graduate code

```
public class Graduate extends Student{
	private int level;

	public Graduate(String s, int n1, int n2) {
		super(s, n1);
		level = n2;
	}
	// accessor
	public int getLevel() {
		return level;
	}
	// mutator
	public void setLevel(int newLevel) {
		level = newLevel;
	}

	public void reset(String newName, int newStudentNumber, int newlevel) {
		setName(newName);
		setStudentNumber(newStudentNumber);
		level = newlevel;
	}
	public void writeOutput() {
		System.out.println("Name: "+ getName());
		System.out.println("Student Number: "+ getStudentNumber());
		System.out.println("Student Level: "+level);
```

```
	}
	public boolean equals(Graduate otherUndergraduate) {
		if(	this.equals(otherUndergraduate)&&
				level== otherUndergraduate.level)
			return true;
		else return false;
	}
	public String toString() {
		return("Name: "+getName()
				+ "\nStudent number: "+getStudentNumber()
				+ "\nStudent Level: "+level);
	}
}
```

Employee2 code

```
public class Employee2 extends Person{
	private String dept;
	private int id;

	// constructor
	public Employee2(String s1, String s2, int n) {
		super(s1);
		dept = s2;
		id = n;
	}
	public Employee2() {
		super();
		dept = "";
		id = 0;
	}
	// accessor
	public String getDept() {
		return dept;
	}
	public int getEmployeeId() {
		return id;
	}
	// mutator
	public void setDept(String newDept) {
		dept = newDept;
	}
```

```
	public void setEmployeeId(int newId) {
		id = newId;
	}
	public void reset(String newName, String newDept, int newId) {
		setName(newName);
		dept = newDept;
		id = newId;
	}
	public void writeOutput() {
		System.out.println("Name: "+getName());
		System.out.println("Dept: "+dept);
		System.out.println("Employee ID: "+id);
	}
	public boolean equals(Employee2 otherEmployee) {
		if(this.hasSameName(otherEmployee)
				&& dept.equals(otherEmployee.dept)
				&& id == otherEmployee.id) return true;
		else return false;
	}
	public String toString() {
		return("Name: "+getName()
				+"Dept: "+dept
				+ "\nEmployee ID: "+id);
	}

}
```

Faculty code

```
public class Faculty extends Employee2 {
	private String title;

	// constructor
	public Faculty(String s1, String s2, int n, String s3) {
		super(s1, s2, n);
		title = s3;
	}
	public Faculty() {
		super();
		title = "";
	}
	// accessor
```

```
	public String getTitle() {
		return title;
	}
	//mutator
	public void setTitle(String newTitle) {
		title = newTitle;
	}

	public void reset(String newName, String newDept, int newId, String newTitle) {
		setName(newName);
		setDept(newDept);
		setEmployeeId(newId);
		title = newTitle;
	}
	public void writeOutput() {
		System.out.println("Name: "+getName());
		System.out.println("Dept: "+getDept());
		System.out.println("Employee ID: "+getEmployeeId());
		System.out.println("Title: "+title);
	}
	public boolean euqals(Faculty otherFaculty) {
		if(this.equals(otherFaculty)&&
				title.equals(otherFaculty.title)) return true;
		else return false;
	}
	public String toString() {
		return("Name: "+getName()
				+ "\nDept: "+getDept()
				+ "\nEmployee ID: "+getEmployeeId()
				+ "\nTitle: "+title);
	}

}
```

Staff code

```
public class Staff extends Employee2 {
	private int pay;	// pay grade

	// constructor
	public Staff(String s1, String s2, int n1, int n2) {
```

```
        super(s1, s2, n1);
        pay = n2;
}
public Staff() {
        super();
        pay = 0;
}
// accessor
public int getPay() {
        return pay;
}
// mutator
public void setPay(int newPay) {
        pay = newPay;
}


public void reset(String newName, String newDept, int newId, int newPay ) {
        setName(newName);
        setDept(newDept);
        setEmployeeId(newId);
        pay = newPay;
}
public void writeOutput() {
        System.out.println("Name: " + getName());
        System.out.println("Dept: " + getDept());
        System.out.println("Employee ID: " + getEmployeeId());
        System.out.println("Pay Grade: " + pay);
}
public boolean equals(Staff otherStaff) {
        if(this.equals(otherStaff)&&
                        pay == otherStaff.pay)
                return true;
        else return false;
}
public String toString() {
        return("Name: "+getName()
                        + "\nDept: "+getDept()
                        + "\nEmployee ID: "+getEmployeeId()
                        + "\nPay Grade: "+pay);
}
```

}

실습 코드 설명

PolymorphismDemo Code

```
public class PolymorphismDemo {
        public static void main(String[] args) {
                Person[] people = new Person[7];
```

⇨ person type으로 된 일차원 array 생성

```
                people[0] = new Undergraduate("Cotty, Manny", 4910, 1);
```

⇨ array 0번째에 undergraduate class으로 된 object 생성

```
                people[1] = new Undergraduate("Kick, Anita", 9931, 2);
```

⇨ array 1번째에 undergraduate class으로 된 object 생성

```
                people[2] = new Student("DeBanque, Robin", 8812);
```

⇨ array 2번째에 student class으로 된 object 생성

```
                people[3] = new Undergraduate("Bugg, June", 9901, 4);
```

⇨ array 3번째에 Undergraduate class으로 된 object 생성

```
                people[4] = new Employee2("Ken,Bill", "Dept of Computer Science", 3056);
```

⇨ array 4번째에 Employee2 class으로 된 object 생성

```
                people[5] = new Faculty("Mock, Kenrick", "Computer Science", 3056, "Prefessor of Computer Science");
```

⇨ array 5번째에 Faculty class으로 된 object 생성

```
                people[6] = new Staff("Jones, James", "Mathematical Sciences", 1551, 15);
```

⇨ array 6번째에 Staff class으로 된 object 생성

```
                for(Person p : people) {
                        p.writeOutput();
```

⇨ array에 저장된 정보들 하나씩 출력

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

⇨ 줄바꿈

```
                }
        }
}
```

Person code

```
public class Person {
        private String name;
    ⇨   Instance variable 생성
        //constructor
        public Person() {
                name = "No name yet";
        }
    ⇨   default constructor
        public Person(String n) {
                name = n;
        }
    ⇨   Object 생성시 자동으로 instance variable 초기화
        // accessor
        public String getName() {
                return name;
        }
    ⇨   다른 class에서 Person class의 name 값 알게 해줌.
        // mutator
        public void setName(String newName) {
                name = newName;
        }
    ⇨   다른 class에서 Person class의 Name 값 수정하게 해줌.
        public void writeOutput() {
                System.out.println("Name: "+name);
        }
    ⇨   Name 값 출력

        public boolean hasSameName(Person otherPerson) {    // equals method
                if( name.equals(otherPerson.name)) return true;
                else return false;
        }
    ⇨   현재 Object와 다른 object의 name 값 같은지 확인
        public String toString() {
                return ("Name: "+name);
        }
    ⇨   출력값 return
}
```

Student code

```
public class Student extends Person{
        private int id;        // student number
```
⇨ Instance variable 생성

```
        // contructors
        public Student() {
                super();          // parent class 안에 있는 instance variables 초기화
```
⇨ Parent class 안에 있는 instance variable 초기화
```
                id = 0;
```
⇨ 현재 class 안에 있는 Instance variable 초기화
```
        }
        public Student(String s, int n) {
                super(s);
```
⇨ Parent class 안에 있는 instance variable 초기화
```
                id = n;
```
⇨ 현재 class 안에 있는 Instance variable 초기화
```
        }

        // accessor
        public int getStudentNumber() {
                return id;
```
⇨ 다른 class에서 Student class의 id 값 알게 해줌.

```
        }
        // mutator
        public void setStudentNumber(int newStudentNumber) {
                id = newStudentNumber;
        }
```
⇨ 다른 class에서 Student class의 Id 값 수정하게 해줌.

```
        public void reset(String newName, int newStudentNumber) {
                setName(newName);
                id = newStudentNumber;
        }
```
⇨ Instance variable 값 수정

⇨ Parent class의 instance variable일 경우 Mutator method 사용

```
        public void writeOutput() {
                System.out.println("Name: " + getName());
                System.out.println("Student Number: "+id);
```

```
        }
    ⇨   Name, student number의 값 출력
        public boolean equals(Student otherStudent) {
                if((this.hasSameName(otherStudent))&& (id== otherStudent.id))
                        return true;
                else return false;
        }
    ⇨   현재 Object와 다른 object의 name, student number의 값 같은지 확인
        public String toString() {
                return("Name: "+getName()
                                + "\nStudent number: "+id);
        }
    ⇨   출력값 return
}
```

Undergraduate code

```
public class Undergraduate extends Student{
        private int level;
    ⇨   Instance variable 생성

        // constructor
        public Undergraduate(String s, int n1, int n2) {
                super(s, n1);
    ⇨   Parent class 안에 있는 instance variable 초기화
                level = n2;
    ⇨   현재 class 안에 있는 instance variable 초기화
        }
        // accessor
        public int getLevel() {
                return level;
    ⇨   다른 class에서 undergraduate class의 level 값 알게 해줌.
        }
        // mutator
        public void setLevel(int newLevel) {
                level = newLevel;
    ⇨   다른 class에서 undergraduate class의 level 값 수정하게 해줌.
        }

        public void reset(String newName, int newStudentNumber, int newlevel) {
                setName(newName);
                setStudentNumber(newStudentNumber);
                level = newlevel;
```

```
    }
```

⇨ Instance variable 값 수정

⇨ Parent class의 instance variable일 경우 Mutator method 사용

```
    public void writeOutput() {
        System.out.println("Name: "+ getName());
        System.out.println("Student Number: "+ getStudentNumber());
        System.out.println("Student Level: "+level);
    }
```

⇨ Name, student number, level의 값 출력

```
    public boolean equals(Undergraduate otherUndergraduate) {
        if(     this.equals(otherUndergraduate)&&
                    level== otherUndergraduate.level)
            return true;
        else return false;
    }
```

⇨ 현재 Object와 다른 object의 name, student number, level의 값 같은지 확인

```
    public String toString() {
        return("Name: "+getName()
                    + "\nStudent number: "+getStudentNumber()
                    + "\nStudent Level: "+level);
    }
```

⇨ 출력값 return

```
}
```

Graduate code

```
public class Graduate extends Student{
    private int level;
```

⇨ Instance variable 생성

```
    public Graduate(String s, int n1, int n2) {
        super(s, n1);
```

⇨ Parent class 안에 있는 instance variable 초기화

```
        level = n2;
```

⇨ 현재 class 안에 있는 instance variable 초기화

```
    }
    // accessor
    public int getLevel() {
            return level;
    }
```
⇨ 다른 class에서 Graduate class의 level 값 알게 해줌.

```
    // mutator
    public void setLevel(int newLevel) {
            level = newLevel;
    }
```
⇨ 다른 class에서 Graduate class의 level 값 수정하게 해줌.

```
    public void reset(String newName, int newStudentNumber, int newlevel) {
            setName(newName);
            setStudentNumber(newStudentNumber);
            level = newlevel;
    }
```
⇨ Instance variable 값 수정

⇨ Parent class의 instance variable일 경우 Mutator method 사용

```
    public void writeOutput() {
            System.out.println("Name: "+ getName());
            System.out.println("Student Number: "+ getStudentNumber());
            System.out.println("Student Level: "+level);
    }
```
⇨ Name, student number의 값 출력

```
    public boolean equals(Graduate otherUndergraduate) {
            if(     this.equals(otherUndergraduate)&&
                            level== otherUndergraduate.level)
                    return true;
            else return false;
    }
```
⇨ 현재 Object와 다른 object의 name, student number, level의 값 같은지 확인

```
    public String toString() {
            return("Name: "+getName()
                            + "\nStudent number: "+getStudentNumber()
                            + "\nStudent Level: "+level);
    }
```

⇨ 출력값 return

}

Employee2 code

```
public class Employee2 extends Person{
        private String dept;
        private int id;
```

⇨ Instance variable 생성

```
        // constructor
        public Employee2(String s1, String s2, int n) {
                super(s1);
                dept = s2;
                id = n;
        }
```

⇨ Parent class 안에 있는 instance variable 초기화

⇨ 현재 class 안에 있는 instance variable 초기화

```
        public Employee2() {
                super();
                dept = "";
                id = 0;
        }
```

⇨ Constructor default 값

```
        // accessor
        public String getDept() {
                return dept;
        }
```

⇨ 다른 class에서 Employee2 class의 dept값 알게 해줌.

```
        public int getEmployeeId() {
                return id;
        }
```

⇨ 다른 class에서 Employee2 class의 employee id 값 알게 해줌.

```
        // mutator
        public void setDept(String newDept) {
                dept = newDept;
        }
```

⇨ 다른 class에서 Employee2 class의 dept 값 수정하게 해줌.

```
	public void setEmployeeId(int newId) {
		id = newId;
	}
```

⇨ 다른 class에서 Employee2 class의 employee id 값 수정하게 해줌.

```
	public void reset(String newName, String newDept, int newId) {
		setName(newName);
		dept = newDept;
		id = newId;
	}
```

⇨ Instance variable 값 수정

⇨ Parent class의 instance variable일 경우 Mutator method 사용

```
	public void writeOutput() {
		System.out.println("Name: "+getName());
		System.out.println("Dept: "+dept);
		System.out.println("Employee ID: "+id);
	}
```

⇨ Name, dept, employee id 의 값 출력

```
	public boolean equals(Employee2 otherEmployee) {
		if(this.hasSameName(otherEmployee)
				&& dept.equals(otherEmployee.dept)
				&& id == otherEmployee.id) return true;
		else return false;
	}
```

⇨ 현재 Object와 다른 object의 name, dept, employee id의 값 같은지 확인

```
	public String toString() {
		return("Name: "+getName()
				+"Dept: "+dept
				+ "\nEmployee ID: "+id);
	}
```

⇨ 출력값 return

```
}
```

Faculty code

```
public class Faculty extends Employee2 {
	private String title;
```

⇨ Instance variable 생성

```
// constructor
public Faculty(String s1, String s2, int n, String s3) {
        super(s1, s2, n);
```
⇨ Parent class 안에 있는 instance variable 초기화
```
        title = s3;
```
⇨ 현재 class 안에 있는 instance variable 초기화
```
}
public Faculty() {
        super();
        title = "";
}
```
⇨ Default constructor

```
// accessor
public String getTitle() {
        return title;
}
```
⇨ 다른 class에서 Faculty class의 title값 알게 해줌.

```
//mutator
public void setTitle(String newTitle) {
        title = newTitle;
}
```
⇨ 다른 class에서 Faculty class의 title 값 수정하게 해줌.

```
public void reset(String newName, String newDept, int newId, String newTitle) {
        setName(newName);
        setDept(newDept);
        setEmployeeId(newId);
        title = newTitle;
}
```
⇨ Instance variable 값 수정

⇨ Parent class의 instance variable일 경우 Mutator method 사용

```
public void writeOutput() {
        System.out.println("Name: "+getName());
        System.out.println("Dept: "+getDept());
        System.out.println("Employee ID: "+getEmployeeId());
        System.out.println("Title: "+title);
}
```
⇨ Name, dept, employee id, title 의 값 출력

```
        public boolean euqals(Faculty otherFaculty) {
                if(this.equals(otherFaculty)&&
                                title.equals(otherFaculty.title)) return true;
                else return false;
        }
```
⇨ 현재 Object와 다른 object의 name, dept, employee id, title의 값 같은지 확인

```
        public String toString() {
                return("Name: "+getName()
                                + "\nDept: "+getDept()
                                + "\nEmployee ID: "+getEmployeeId()
                                + "\nTitle: "+title);
        }
```
⇨ 출력값 return

```
}
```
Staff code

```
public class Staff extends Employee2 {
        private int pay;        // pay grade
```
⇨ Instance variable 생성

```
        // constructor
        public Staff(String s1, String s2, int n1, int n2) {
                super(s1, s2, n1);
```
⇨ Parent class 안에 있는 instance variable 초기화

```
                pay = n2;
```
⇨ 현재 class 안에 있는 instance variable 초기화

```
        }

        public Staff() {
                super();
                pay = 0;
        }
```
⇨ Default constructor

```
        // accessor
        public int getPay() {
                return pay;
        }
```
⇨ 다른 class에서 staff class의 pay(pay grade)값 알게 해줌.

```
    // mutator
    public void setPay(int newPay) {
            pay = newPay;
    }
```
⇨ 다른 class에서 staff class의 pay(pay grade) 값 수정하게 해줌.

```
    public void reset(String newName, String newDept, int newId, int newPay ) {
            setName(newName);
            setDept(newDept);
            setEmployeeId(newId);
            pay = newPay;
    }
```
⇨ Instance variable 값 수정

⇨ Parent class의 instance variable일 경우 Mutator method 사용

```
    public void writeOutput() {
            System.out.println("Name: " + getName());
            System.out.println("Dept: " + getDept());
            System.out.println("Employee ID: " + getEmployeeId());
            System.out.println("Pay Grade: " + pay);
    }
```
⇨ Name, dept, employee id, pay(pay grade) 의 값 출력

```
    public boolean equals(Staff otherStaff) {
            if(this.equals(otherStaff)&&
                            pay == otherStaff.pay)
                    return true;
            else return false;
    }
```
⇨ 현재 Object와 다른 object의 name, dept, employee id, pay(pay grade)의 값 같은지 확인

```
    public String toString() {
            return("Name: "+getName()
                            + "\nDept: "+getDept()
                            + "\nEmployee ID: "+getEmployeeId()
                            + "\nPay Grade: "+pay);
    }
```
⇨ 출력값 return

```
}
```