REPORT

전자공학도의 윤리 강령 (IEEE Code of Ethics)

(출처: http://www.ieee.org)

나는 전자공학도로서, 전자공학이 전 세계 인류의 삶에 끼치는 심대한 영향을 인식하여 우리의 직업, 동료와 사회에 대한 나의 의무를 짐에 있어 최고의 윤리적, 전문적 행위를 수행할 것을 다짐하면서, 다음에 동의한다.

1. **공중의 안전, 건강 복리에 대한 책임**: 공중의 안전, 건강, 복리에 부합하는 결정을 할 책임을 질 것이며, 공중 또는 환경을 위협할 수 있는 요인을 신속히 공개한다.

2. **지위 남용 배제**: 실존하거나 예기되는 이해 상충을 가능한 한 피하며, 실제로 이해가 상충할 때에는 이를 이해 관련 당사자에게 알린다. (이해 상충: conflicts of interest, 공적인 지위를 사적 이익에 남용할 가능성)

3. **정직성**: 청구 또는 견적을 함에 있어 입수 가능한 자료에 근거하여 정직하고 현실적으로 한다.

4. **뇌물 수수 금지**: 어떠한 형태의 뇌물도 거절한다.

5. **기술의 영향력 이해**: 기술과 기술의 적절한 응용 및 잠재적 영향에 대한 이해를 높인다.

6. **자기계발 및 책무성**: 기술적 능력을 유지, 증진하며, 훈련 또는 경험을 통하여 자격이 있는 경우이거나 관련 한계를 전부 밝힌 뒤에만 타인을 위한 기술 업무를 수행한다.

7. **엔지니어로서의 자세**: 기술상의 업무에 대한 솔직한 비평을 구하고, 수용하고, 제공하며, 오류를 인정하고 수정하며, 타인의 기여를 적절히 인정한다.

8. **차별 안하기**: 인종, 종교, 성별, 장애, 연령, 출신국 등의 요인에 관계없이 모든 사람을 공평하게 대한다.

9. **도덕성**: 허위 또는 악의적인 행위로 타인, 타인의 재산, 명예, 또는 취업에 해를 끼치지 않는다.

10. **동료애**: 동료와 협력자가 전문분야에서 발전하도록 도우며, 이 윤리 헌장을 준수하도록 지원한다.

위 IEEE 윤리헌장 정신에 입각하여 report를 작성하였음을 서약합니다.

학 부: 전자공학부
마감일: 2021. 05. 23.
과목명: 전자공학프로그래밍
교수명: 이정원 교수님
학 번: 201620837
성 명: 안민규

1. 함수 기능 및 코드 분석

1) Adult

```
package assignment3;

public class Adult extends Visitor{
	int fee;
	boolean status; //변수 선언

	public Adult(int age, int fee) {
		this.age = age;			//age로 입력되는 값 age에 저장
		this.fee = fee;			//fee로 입력되는 값 fee에 저장
		this.status = false;		//입장 관련 변수 false로 초기화
	}

	public boolean pay() {
		if(this.fee - ADULT_FEE < 0) //입장료가 정해진 ADULT_FEE 값보다 낮다면 pay
			return false;		//에 false -> 입장료 못 냄
		else
			return true;		//아니라면 true
	}

	public boolean isMember() {
		return false;			//회원이 아니므로 false
	}
}
```

Adult로 선언된 class는 main에서 Adult로 선언된 v들을 관리한다. vX에 Adult(나이, 입장료)로 선언하면 vX에 Adult함수를 사용해 값들을 저장함. 후에 조건에 따라 입장하는 것을 관리한다.
boolean 변수로 입장 상태를 선언하고 요금은 int로 선언.

2) Child

```
package assignment3;

public class Child extends Visitor{
	boolean status;			//입장료가 0이므로 입장했는지에 관련한 변수만 선언

	public Child(int age) {		//다른 것들과 다르게 나이만을 이용해 호출
		this.age = age;			//입력된 age값 저장
		this.status = false;		//입장 관련 변수 false로 초기화
	}

	public boolean pay() {
		return true;			//입장료가 0이므로 pay는 항상 true 반환
	}

	public boolean isMember() {	//회원으로 불러오지 않으므로 회원값 false
		return false;
	}
}
```

Child로 선언된 class는 main에서 Child로 선언된 v들을 관리한다. vY에 Child(나이)로 선언하면 vY에 Child 함수를 사용해 값들을 저장함. 후에 저장된 값들을 이용해 입장 관리.
앞과 같이 입장상태를 boolean 변수로 선언하고, 요금관련 변수는 선언하지 않는다.

3) Student

```
package assignment3;

public class Student extends Visitor{
	int fee;
	boolean status;

	public Student(int age, int fee) {
```

```
                this.age = age;
                this.fee = fee;
                this.status = false;
        }

        public boolean pay() {
                if(this.fee - STUDENT_FEE < 0) //정해진 입장료보다 갖고 있는 돈이 낮으므로
                        return false;                //pay에 false -> 입장 못 함
                else
                        return true;                 //아니라면 true
        }

        public boolean isMember() {
                return false;                    //회원이 아니므로 false
        }
}
```

Student 함수로 Main에서 student로 선언될 때 사용된다. 입력되는 나이와 돈을 갖고 이를 저장한 후에 이를 가지고 돈을 내었는지를 확인한다. 또한 Member로 선언되지 않았으므로 회원 상태는 false가 된다.

4) Member

```
package assignment3;

public class Member extends Visitor{
        boolean status;             //회원이라면 입장료가 0이므로 따로 변수 선언 X

        public Member(int age) {
                this.age = age;
                this.status = false;
        }

        public boolean pay() {
                return true;          //회원은 입장료를 내지않으므로 pay 항상 true 반환
        }

        public boolean isMember() {
                return true;          //회원이므로 true 반환
        }
}
```

Member로 Main에서 member로 선언될 때 사용된다. 회원인 사람을 선언할 때 사용하고, 회원의 경우 나이와 상관없이 항상 입장료가 0이므로 입장 상태에 관련한 변수만 boolean으로 선언하였다. 입장료가 0이므로 입장료가 항상 0이므로 돈을 냈는지 확인하는 pay는 항상 true를 반환한다. 또한 isMember도 true를 반환한다.

5) Visitor

```
package assignment3;

interface Common {
        int getAge();
        boolean getStatus();
} //Visitor 함수에서 사용하기 전에 common interface 선언

public abstract class Visitor implements Common{
        int age;          //나이 변수 선언
        boolean status; //상태 변수 선언
        final static int STUDENT_FEE = 2000; //학생의 입장료 2천원으로 고정
        final static int ADULT_FEE = 5000; //어른의 입장료 5천원으로 고정

        public abstract boolean pay(); //돈을 냈는지 안 냈는지
        public abstract boolean isMember(); //회원인지 아닌지
```

```
	public int getAge() {
		return age; //age 반환
	}
	public boolean getStatus() {
		return this.status; //status 반환
	}

}
```

방문하는 자들을 관리하는 함수이다. Museum함수에서 enter 혹은 exit를 작동하기 위한 인자로 사용된다. Main에서 사람을 선언할 때 이들을 Museum함수 내에서 비교한다. Main에서 사람을 선언할 때 예를 들어 v1에 Adult(25, 5000)을 저장하였으므로 Museum함수 내에서 불러오는 v1의 경우 저장된 class와 입장료, 입장상태 등을 확인한다.

6) Museum

```
package assignment3;

public class Museum {
	static int currentNum;    //현재 박물관 관람객 수
	int todayNum;              //일일 박물관 관람객 수
	int sales;                 //입장하는 사람의 class에 따라 입장료값
	int totalSales;            //일일 수익
	String Class;              //class를 출력하고 이를 이용해 나누기 위해 선언

	Museum(){
		currentNum = todayNum = sales = totalSales = 0; //앞에서 선언한 변수들 0으로 초기화
	}

	int getTodayNum() {
		return todayNum;  //일일 관람객 수 값을 반환하는 함수
	}

	int getSales() {
		return sales;      //사람의 입장료 값 반환
	}

	public boolean enter(Visitor v) { //enter 함수
		if(v instanceof Adult) {
			Class="adult";
		}
		else if (v instanceof Student) {
			Class="Student";
		}
		else if (v instanceof Child) {
			Class="Child";
		}
		else if (v instanceof Member) {
			Class="Member";
		} //입장한 사람의 class를 Class에 저장
		if(v instanceof Adult || v instanceof Student) { //어른이거나 학생일 때
			if(v.pay() == false) { //돈을 내지 않았다면 - 입장료가 모자라다면
				sales = (v instanceof Adult ? Visitor.ADULT_FEE : Visitor.STUDENT_FEE); //내야하는 입장료를 저장 어른이라면 어른의 입장료, 아니라면 학생의 입장료
				System.out.println("-> 입장 거부 : "+Class); //돈을 안낸 사람이므로 입장 거부 하며 저장된 Class 출력
				System.out.println("-> 필요한 입장료 : "+sales); //입장하려는 class의 필요한 입장료인 sales 출력
				sales=0; //입장료를 출력해줬으므로 sales값 초기화
				return false; //false를 반환하며 함수 종료
			}
			sales = (v instanceof Adult ? Visitor.ADULT_FEE : Visitor.STUDENT_FEE); //돈을 냈다면 class에 따라 sales에 입장료 저장
		}
```

```
                if(v instanceof Child) { //어린이일 때
                        if(v.age>8) { //나이가 8살보다 많다면 나이 제한에 걸리므로
                                System.out.println("-> 입장 거부 : "+Class);
                                System.out.println("-> 나이 : "+v.age);
                                sales=0; //sales값 0으로 초기화
                                return false; //입장 불가
                        }
                }
                else if(v instanceof Student) { //학생일 때
                        if(v.age>15||v.age<8) { //나이가 8보다 적거나 15살보다 많다면 나이 제한에
걸리므로
                                System.out.println("-> 입장 거부 : "+Class);
                                System.out.println("-> 나이 : "+v.age);
                                sales=0;
                                return false; //입장 불가
                        }
                }
                else if(v instanceof Adult) { //어른일 때
                        if(v.age<16) { //나이가 16살보다 작다면 나이 제한에 걸리므로
                                System.out.println("-> 입장 거부 : "+Class);
                                System.out.println("-> 나이 : "+v.age);
                                sales=0;
                                return false; //입장 불가
                        }
                }
                System.out.println("관람객 입장 : "+Class);
                v.status = true; //입장했으므로 입장 상태를 나타내는 status값 true
                todayNum++; //입장할 때마다 입장한 사람 수 1씩 증가
                currentNum++; //현재 관람객 수 1씩 증가
                totalSales+=sales; //전체 수익에 입장한 사람의 입장료 더한 후 다시 전체 수익에
저장
                System.out.println("현재 관람객 수 : "+currentNum);
                System.out.println("일일 관람객 수 : "+todayNum);
                System.out.println("일일 수익 : "+totalSales);
                sales=0; //sales를 다시 0으로 초기화
                return true; //true 반환
        }

        public void exit(Visitor v) { //exit 함수
                if(v.status) { //현재 입장한 사람이라면 status가 true이므로
                        System.out.println("관람객 퇴장");
                        currentNum--; //현재 관람객 수 1 감소
                        System.out.println("현재 관람객 수 : "+currentNum);
                }
                else { //status가 false라면 이미 퇴장했거나 입장하지 못 한 사람이므로
                        System.out.println("현재 관람객 없음"); //관람객 없다는 메세지 출력
                        System.out.println("현재 관람객 수 : "+currentNum);
                }
        }
}
```

Museum 함수로 박물관에 입장한 사람들을 관리한다. 사람들의 입장과 퇴장과 관련된 함수가 들어있다. Main에서 선언된 visitor vX로 enter와 exit함수를 호출한다.

Main함수에서 enter 혹은 exit 함수를 호출했다면 호출인자 vX에 따라 조건에 따라 행동을 달리한다.

기본적으로 enter함수가 실행하면 입력된 호출인자 v를 선언한 class 값을 Class 변수에 저장한다. 저장한 후 입장료를 냈는지 안 냈는지를 확인한다. 회원과 어린이는 입장료를 내지 않으므로 v가 어른이나 학생일 때만 입장료 검사를 진행한다. 입장료를 내지 않았으면 입장이 불가능하므로 sales 변수에 저장되어있는 어른 혹은 학생의 입장료를 저장하고 Class와 sales를 출력하고 함수를 종료한다. 입장료를 냈다면 sales에 어른 혹은 학생의 입장료를 저장하고 함수를 계속해서 진행한다.

이후 나이 제한에 맞추어 조건문을 진행하고 이에 맞지 않으면 입장 거부 오류문을 출력하고 함수를 종료한다. sales를 초기화해주지 않으면 입장하지 않거나 입장료가 없는 사람에게도 먼저 입장한 사람의 입장료가 출력되므로 sales를 초기화시켜줘야 한다.

잘 입장했다면 입장한 사람의 Class를 출력하고 status를 true로 변경한다. 또한 지금까지 입장한 사람

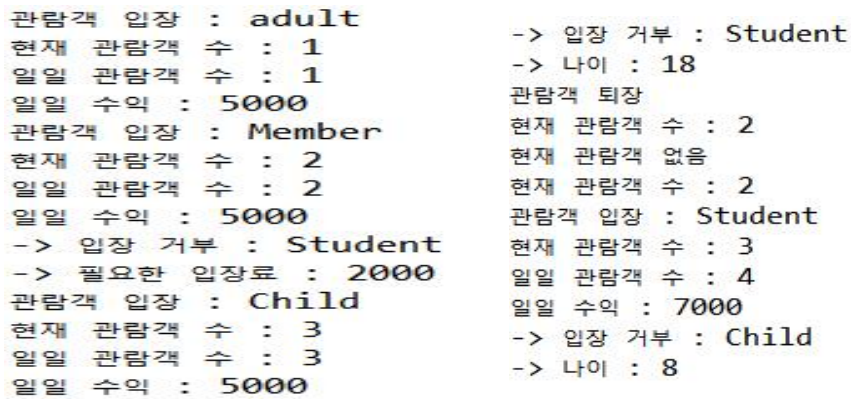
수와 현재 관람객 수를 1씩 증가시키고 입장한 사람의 입장료를 totalSales에 더한다. 이후 sales를 다시 0으로 초기화를 하고 true를 반환한다.
exit함수는 현재 입장한 사람을 퇴장시키는 함수이다. status가 true여야 입장한 상태이기 때문에 status가 true일 때만 정상적으로 퇴장 문구가 나오게 설정하였다. 그 후 현재 관람객 수를 1 감소시키고 현재 관람객 수를 출력한다. status가 false라면 이미 퇴장했거나 입장하지 못한 사람이므로 관람객이 없다는 문구를 출력하고 현재 관람객 수를 출력한다.

7) Main

```
package assignment3;

public class Main {
	public static void main(String[] args) {
		Museum m = new Museum(); //박물관 선언
		Adult v1 = new Adult(25, 5000); //25살에 5천원의 입장료를 가진 어른 선언
		Member v2 = new Member(30);	//30살에 회원 선언
		Student v3 = new Student(14, 1000);  //14살에 1천원의 입장료를 가진 학생 선언
		Child v4 = new Child(6);		//6살 어린이 선언
		Student v5 = new Student(18, 5000);  //18살에 5천원의 입장료를 가진 학생 선언
		Student v6 = new Student(15, 2000);  //15살에 2천원의 입장료를 가진 학생 선언
		Child v7 = new Child(8);		//8살 어린이 선언

		m.enter(v1); //어른 입장
		m.enter(v2); //회원 입장
		m.enter(v3); //학생 입장료 제한 확인
		m.enter(v4); //어린이 입장
		m.enter(v5); //학생 나이 제한 확인

		m.exit(v1); //어른 퇴장
		m.exit(v3); //입장 못한 사람 퇴장 확인
		m.enter(v6); //학생 입장
		m.enter(v7); //어린이 나이 제한 확인
	}

}
```

Main으로 실제 프로그램이 진행되는 함수이다. main 함수에서는 입장하는 사람들을 실제로 선언하고 이들의 입장과 퇴장을 관리한다. 함수를 호출할 때 값을 넣어 함수를 호출하여 사용한다.
여러 수정사항을 확인하기 위해 여러 사람을 호출하여 이를 확인하였다. m.enter의 m의 경우 앞서 선언된 museum이고 enter와 exit을 이용해 입장과 퇴장을 관리한다.

2. 결과화면 분석



1. adult 입장 : class로 adult 출력, 입장했기 때문에 일일 수익 +5000 확인
2. Member(회원)이므로 입장료 없이 입장 가능
3. Student 입장하려 했으나 입장료가 모자라서 입장 불가능 -> 입장 거부 당한 class와 필요한 입장료 출력
4. Child 입장 : 어린이는 입장료가 필요하지 않으므로 바로 입장, 나이 제한에 맞음으로 입장 가능
5. 한 번 더 Student 입장하려 했으나 나이제한에 맞지 않아 입장 불가능
6. 먼저 들어왔던 사람 퇴장하여 현재 관람객 2 출력
7. 입장 못 했던 Student 퇴장시키려 했으나 입장하지 않았기 때문에 관람객 없음 문구 출력됨
8. Student가 입장하려 할 때 나이 제한에 맞고 충분한 입장료가 있으므로 입장 가능
9. 현재 관람객 수 3으로 증가
10. 현재까지 입장한 사람 어른 1명, 회원 1명, 어린이 1명, 학생 1명이므로 일일 관람객 수 4명, 어른 1명 들어와 5천원 + 학생 1명 2천원 -> 일일 수익 7천원
11. 마지막 어린이 나이가 8로 나이 제한에 맞지 않아 입장 불가