dongguk UNIVERSITY

객체지향프로그래밍

실습강의 10주차

실습강의 소개

- **실습 진행 방법**
  - 간단한 이론 복습 및 해당주차 실습과제 설명
  - 실습 후 보고서와 소스코드를 압축하여 수요일 자정(23:59)까지
    꼭!! 이클래스 제출(**이메일 제출 불가, 반드시 이클래스를 통해 제출**)
  - 실습 과제 제출 기한 엄수(제출 기한 이후로는 0점 처리)
  - **보고서는 출력하여 수업 시작 전에 제출(대면으로 실습 수업 진행 시)**
- **Q & A**
  - **이클래스 및 실습조교 이메일을 통해 질의 응답**
  - **이메일 제목 : [객체지향프로그래밍_홍길동] *본인 과목명과 성명 꼭 작성!**
  - **실습조교 메일 주소 : pmj6823@dgu.ac.kr**
  - **수업용 깃허브 조직 : https://github.com/23-2-Object-Oriented-Programming**

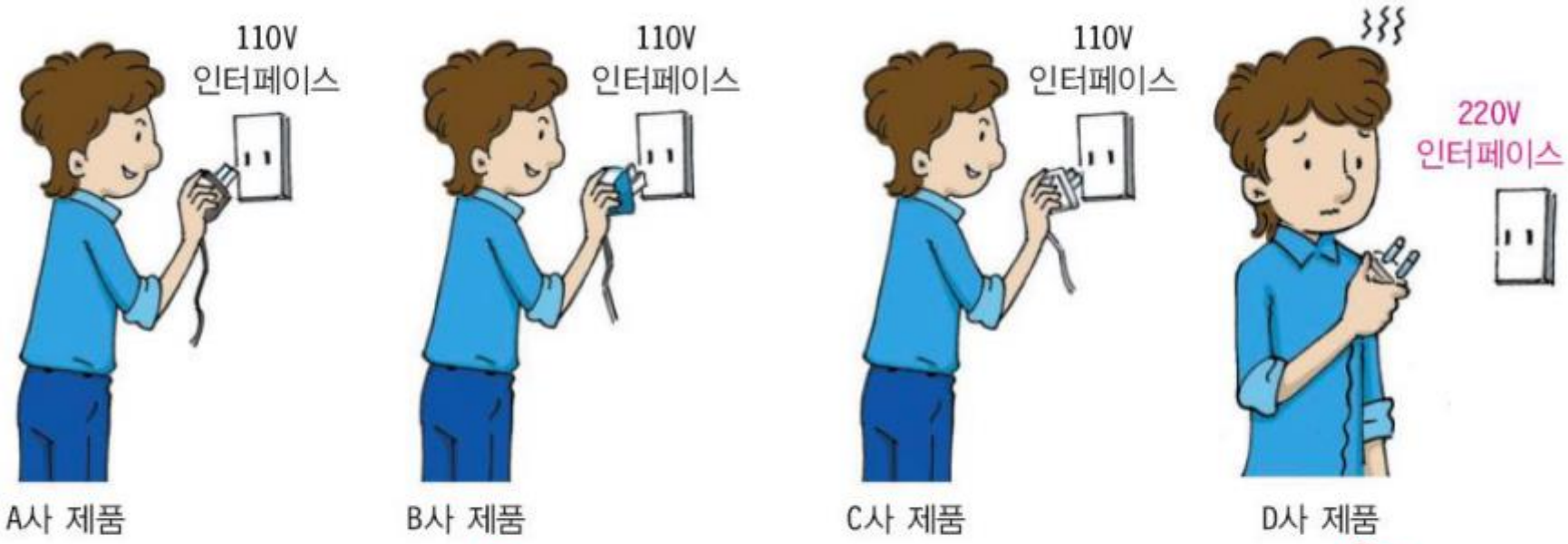
실습강의 소개

- **실습 보고서**
  - 문제 분석, 프로그램 설계 및 알고리즘, 소스코드 및 주석, 결과 및 결과 분석, 소감
- **제출 방법**
  - 보고서, 소스코드, 실행파일을 1개의 파일로 압축하여 e-class "과제" 메뉴를 통해 제출
    - **"이름학번실습주차.zip" 형태로 제출(e.g. :김동국19919876실습7.zip)**
    - 파일명에 공백, 특수 문자 등 사용 금지
  - 대면 수업일 경우 출력한 보고서를 실습 시간에 제출
- **유의 사항**
  - 보고서의 표지에는 학과, 학번, 이름, 담당 교수님, 제출일자 반드시 작성
  - 정해진 기한내 제출
    - 기한 넘기면 0점 처리
    - e-class가 과제 제출 마지막 날 오류로 동작하지 않을 수 있으므로, 최소 1~2일 전에 제출
    - **당일 e-class 오류로 인한 미제출은 불인정**
  - 소스코드, 보고서를 자신이
  - **Eclipse, 동국대학교 Shastra**를 작성하지 않은 경우 **실습 전체 점수 0점 처리**사용하여 실습 진행

보고서 작성 방법

- **보고서 양식**
  - 문제 분석: 실습 문제에 대한 요구 사항 파악, 해결 방법 등 기술
  - 프로그램 설계 및 알고리즘
    - 해결 방법에 따라 프로그램 설계 및 알고리즘 등 기술
    - e.g.) 문제 해결 과정 및 핵심 알고리즘 기술
  - 소스코드 및 주석
    - 소스코드와 그에 해당하는 주석 첨부
    - 각각의 함수가 수행하는 작업, 매개변수, 반환 값 등을 명시
    - **소스코드 전체 첨부(소스코드 화면 캡처X, 소스코드는 복사/붙여넣기로 첨부)**
  - 결과 및 결과 분석
    - 결과 화면을 캡처하여 첨부, 해당 결과가 도출된 이유와 타당성 분석
  - 소감
    - 실습 문제를 통해 습득할 수 있었던 지식, 느낀 점 등을 기술

인터페이스의 필요성



자바 인터페이스

- **인터페이스**
  - 클래스가 구현해야 할 메소드들이 선언되는 추상형
  - 인터페이스 선언
    - interface 키워드로 선언
  - 다형성을 실현하는 도구 중 하나
- **인터페이스 구성 요소**
  - 상수: public 만 허용, public static final 생략 가능
  - 추상 메소드: public abstract 생략 가능
  - default 메소드: 인터페이스에 코드가 작성된 메소드. public만 가능하며 생략 가능
  - private 메소드: 인터페이스 내에 메소드 코드가 작성되어야 함
  - static 메소드: public, private 모두 가능. 생략하면 public

```
interface PhoneInterface {
    public static final int TIMEOUT = 10000; // 변수가 아닌 상수만 선언 가능
    public abstract void sendCall(); // public abstarct는 생략 가능
    void receiveCall();
    private testCallSound(); // 인터페이스에 정의된 메소드로만 호출 가능
    public default void printLogo(){ // default 메소드 public 생략 가능
        System.out.println("** Phone **");
    }
}
```

자바 인터페이스 사례

```
interface PhoneInterface {
    public static final int TIMEOUT = 10000; // 변수가 아닌 상수만 선언 가능
    public abstract void sendCall(); // public abstarct는 생략 가능
    void receiveCall();
    private testCallSound(); // 인터페이스에 정의된 메소드로만 호출 가능
    public default void printLogo(){ // default 메소드 public 생략 가능
        System.out.println("** Phone **");
    };
}
```

자바 인터페이스

- **인터페이스 특징**
  - 인터페이스의 객체 생성 불가
  - 인터페이스 타입의 레퍼런스 변수 선언 가능

```
PhoneInterface iphone15 = new Iphone15();
```

  - 인터페이스 간에 상속 가능

```
interface ApplePhone extends PhoneInterface
```

  - 인터페이스 다중 상속 허용

```
interface AppleTablet extends PhoneInterface, TabletInterface
```

- **인터페이스 구현**
  - 인터페이스의 추상 메소드를 모두 구현한 클래스 작성
  - 여러 개의 인터페이스 동시 구현 가능

```
public class Iphone15 extends ApplePhone {
    // PhoneInterface의 모든 메소드 구현
    public void sendCall() { System.out.println("상대방 컬러링"); }
    public void receiveCall() { System.out.println("아이폰 알람"); }
    // 메소드 추가 작성
    public void flash() { System.out.println("전화기에 불이 켜졌습니다."); }
}
```

[연습 1] 인터페이스 예제

- interface PhoneInterface와 class Calc를 만들고 이를 상속 및 구현하는 class SmartPhone를 만들어라.

```
interface PhoneInterface { // 인터페이스 선언
    final int TIMEOUT = 10000; // 상수 필드 선언
    void sendCall(); // 추상 메소드
    void receiveCall(); // 추상 메소드
    default void printLogo() { // default 메소드
        System.out.println("** Phone **");
    }
}
class Calc { // 클래스 작성
    public int calculate(int x, int y) {
        return x + y;
    }
}
```

```
public class InterfaceEx {
    public static void main(String[] args) {
        SmartPhone phone = new SmartPhone();
        phone.printLogo();
        phone.sendCall();
        System.out.println("3과 5를 더하면 " + ph
one.calculate(3, 5));
        phone.schedule();
    }
}
```

```
// SmartPhone 클래스는 Calc를 상속받고,
// PhoneInterface 인터페이스의 추상 메소드 모두 구현
class SmartPhone extends Calc implements PhoneInterface {
    // PhoneInterface의 추상 메소드 구현
    @Override
    public void sendCall() {
        System.out.println("따르릉따르릉~~");
    }
    @Override
    public void receiveCall() {
        System.out.println("전화 왔어요.");
    }
    // 추가로 작성한 메소드
    public void schedule() {
        System.out.println("일정 관리합니다.");
    }
}
```

자바 Object 클래스

- **특징**
  - 모든 자바 클래스는 반드시 Object 클래스를 상속받도록 자동 컴파일됨
  - 모든 클래스의 부모 클래스이며 공통 메소드가 포함됨
- **주요 메소드**

메소드	설명
boolean equals(Object obj)	obj가 가리키는 객체와 현재 객체를 비교하여 같으면 true 리턴
Class getClass()	현 객체의 런타임 클래스를 리턴
int hashCode()	현 객체에 대한 해시 코드 값 리턴
String toString()	현 객체에 대한 문자열 표현을 리턴
void notify()	현 객체에 대해 대기하고 있는 하나의 스레드를 깨운다.
void notifyAll()	현 객체에 대해 대기하고 있는 모든 스레드를 깨운다.
void wait()	다른 스레드가 깨울 때까지 현재 스레드를 대기하게 한다.

toString 메소드

- **각 클래스는 toString을 오버라이딩하여 자신만의 문자열 리턴 가능**
  - 객체를 문자열로 반환
  - 다형성을 실현하는 도구 중 하나
- **컴파일러에 의해 toString 자동 변환**
  - '객체 + 문자열' -> '객체.toString() + 문자열'로 자동 변환
  - 객체를 단독으로 사용하는 경우 -> 객체.toString()으로 자동 변환

[연습 2] toString() 작성

- class Point에 toString 메소드를 오버라이딩하고 자동 변환을 통해 toString()을 호출하라.

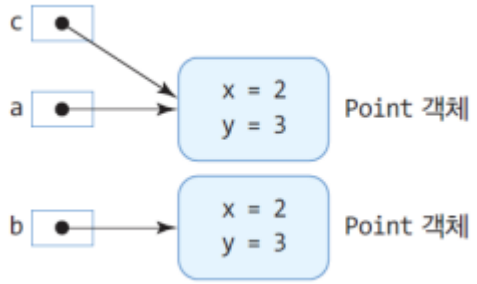
```
class Point {
    private int x, y;
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
    public String toString() {
        return "Point(" + x + "," + y + ")";
    }
}
```

```
public class ToStringEx {
    public static void main(String[] args) {
        Point a = new Point(2, 3);
        System.out.println(a.toString());
        System.out.println(a); // a는 a.toString()으로 자동 변환됨
    }
}
```

equals 메소드

- == 연산자
  - 객체 레퍼런스 비교
  - 객체 내부의 값이 아닌 주소를 비교하여 예상과 다른 결과 도출 가능

```
Point a = new Point(2, 3);
Point b = new Point(2, 3);
Point c = a;
if (a == b) { // false
    System.out.println("a==b");
}
if (a == c) { // true
    System.out.println("a==c");
}
```



- boolean equals(Object obj)
  - 두 객체의 내용물을 비교
  - 객체의 내용물을 비교하기 위해 클래스의 멤버로 작성
  - 호출 시 업캐스팅이 일어남

[연습 2] equals() 작성

- class Point에 equals 메소드를 오버라이딩하고 main 함수의 결과에 대해 설명하라.

```
class Point {
    int x, y;
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
    public boolean equals(Object obj) {
        Point p = (Point) obj; // obj를 Point 타입으로 다운 캐스팅
        if (x == p.x && y == p.y) return true;
        else return false;
    }
}
```

```
public class EqualsEx {
    public static void main(String[] args) {
        Point a = new Point(2, 3);
        Point b = new Point(2, 3);
        Point c = new Point(3, 4);
        if (a == b) System.out.println("a==b");
        if (a.equals(b)) System.out.println("a is equal to b");
        if (a.equals(c)) System.out.println("a is equal to c");
    }
}
```

실습과제 안내 [1/2]

- **실습과제 1**
  - 어느 온라인 쇼핑몰에서 상품마다 고정된 가격을 할인해주는 정책, 상품 가격에 비례해 할인을 해주는 정책이 있다. 이를 구현하기 위해 DiscountPolicy라는 인터페이스를 만들었다. 이를 상속 받은 FixedDiscountPolicy, PercentDiscountPolicy 두 클래스를 구현해야 한다.
  - discountValue 변수는 할인되는 양을 의미하며 FixedDiscountPolicy에서 getPrice 메소드는 discountValue만큼 할인된 값을 반환한다. PercentDiscountPolicy에서 getPrice 메소드는 discountValue %만큼 할인된 값을 반환한다.
  - 추상 클래스와 인터페이스의 공통점과 차이점을 아래 링크를 참고하여 소감에 작성하세요. 인터페이스와 추상 클래스의 차이점

- **처리조건**
  - interface
  - extends
  - 업캐스팅
  - 오버라이딩
  - 입출력

- **실행결과**

```
------------------------------------------------

상품의 가격을 입력하세요: 31000
discountValue 설정 값을 입력하세요: 1000
할인 정책을 선택하세요(1. 고정 금액 할인, 2. 퍼센트 할인): 1
할인된 상품 가격: 30000원

------------------------------------------------

상품의 가격을 입력하세요: 30000
discountValue 설정 값을 입력하세요: 10
할인 정책을 선택하세요(1. 고정 금액 할인, 2. 퍼센트 할인): 2
할인된 상품 가격: 27000원

------------------------------------------------
```

실습과제 안내 [2/2]

- **실습과제 2**
  - 동아리 인사 홍보를 담당하는 집행원이 실수로 같은 동아리원의 정보를 두 번 입력하여 동아리원 목록에 중복된 데이터가 누적되었다. 기존 Member 클래스(10주차 실습 과제 참고)에 student id를 이용해 같은 동아리원인지 판별하는 equals 메소드를 구현하고 중복된 동아리원의 student id와 이름을 출력하라.
  - Crew 클래스를 사용하지 않아도 됩니다.
  - 업캐스팅, 다운캐스팅을 사용하지 않고 equals를 구현할 경우 어떤 단점이 있는지 소감에 작성하세요.

- **처리조건**
  - extends
  - 업캐스팅, 다운캐스팅
  - override

- **실행결과**

```
일반 동아리원 수를 입력하시오: 2
집행부 수를 입력하시오: 2

---Input Crew Information---
Student ID: 1011
Name: CREW1
Department: PRESIDENT

Student ID: 1213
Name: CREW2
Department: SECRETARY


---Input Member Information---
Student ID: 1234
Name: ST1

Student ID: 5678
Name: ST2
```

```
---Crew Information---
Student ID: 1011
Name: CREW1
Department: PRESIDENT

Student ID: 1213
Name: CREW2
Department: SECRETARY


---Member Information---
Student ID: 1234
Name: ST1

Student ID: 5678
Name: ST2
```