협성대학교 소프트웨어공학과

인터넷프로그래밍 2 기말고사 정리

9, 10, 11, 13 주차 강의 내용 정리. (교재 5, 6, 7, 9 장 정리.)

이시헌 제작

2023-12-12

9주차 수업 정리

상속: extends 키워드로 수행하면 된다.

```
class Point { }
class ColorPoint extends Point { }
```

주의사항.

1. 서브클래스 객체는 슈퍼클래스 객체와 별개다. (서브클래스 객체는 슈퍼클래스의 멤버를 포함한다.)
2. C++은 다중 상속이 되지만, JAVA는 다중상속이 불가능하다. (단, Interface는 다중상속이 가능하다.)
3. 모든 JAVA 클래스는 묵시적으로 Object 클래스를 상속받는다. (java.lang.Object 는 모든 클래스의 슈퍼클래스다.)

상속시의 접근 제한자 동작. – p.197

1. 동일 패키지 내부에서의 상속: private멤버만 차단된다. (서브클래스가 슈퍼클래스의 private에 접근 불가.)
2. 다른 패키지로부터의 상속: private와 default가 차단된다.

(참고! 패키지: 다수의 컴파일된 .class파일을 저장하고 관리하는 디렉터리.)

서브클래스 객체 생성시의 생성자 호출: 다른 언어들과 동일함.

➔ 서브클래스 생성자 호출 –〉 슈퍼클래스 생성자 호출 –〉 슈퍼클래스 생성자 실행 –〉 서브클래스 생성자 실행.

상속시의 슈퍼클래스 생성자 선택

1. 개발자의 명시적 선택이 가능함.
   A. super( )키워드로 슈퍼클래스에서 실행될 생성자 선택 가능.
   B. ‘super( )’ 자체가 슈퍼클래스의 생성자다. 이 생성자에 집어넣는 인자를 통하여 원하는 생성자를 호출하는 것.

C. super( )를 서브클래스의 생성자의 첫번째 행에 서술해야 한다. 항상!

2. 컴파일러가 자동으로 골라준다.

A. 명시하지 않으면 항상 슈퍼클래스의 기본 생성자 호출함. (기본 생성자가 없을 때 오류 발생. 다른 언어들과 다를 것 없음.)

업캐스팅 / 다운캐스팅 - p.204

1. 업캐스팅: 서브클래스 객체를 슈퍼클래스 타입으로 변환하는 것. 말 그대로 업캐스팅이다

A. 슈퍼클래스 객체에 서브클래스 타입 객체를 대입하면, 자동으로 수행됨.

2. 다운캐스팅: 업캐스팅 된 객체를 원래대로 되돌리는 것이다.

A. 반드시 강제로 수행해야 한다. (명시적으로 캐스팅을 해야 한다.)

3. 캐스팅 된 객체가 현재 어떤 타입의 객체인지 확인이 어렵다 -〉 'instanceof' 연산자를 사용하면 확인할 수 있다.

A. 객체 레퍼런스 instanceof 클래스 타입

B. 위의 형태로 사용하며, 위의 식 자체가 true/false 값을 갖는 논리식이다.

업캐스팅 / 다운캐스팅 예시.

```
class Person{
    String name;
    String id;

    public Person(String name){
        this.name = name;
    }
}
class Student extends Person {
    String grade;
    String department;
    public Student(String name){
        super(name);
    }
}
public class UpcastingEx{
    public static void main(String[] args){
        Person p;
        Student s = new Student("시헌");
        p = s;  //업캐스팅.  Student -> Person

        System.out.println(p.name);

        //p.grade = "A"  :  오류 발생.
        //p.department = "Com"  :  오류 발생.

        Student s = (Student)p;  //다운캐스팅. Person -> Student

        if(p instanceof Student)  //true. p는 Student 타입이다.
    }
}
```

메소드 오버라이딩(Method Overriding)

1. 서브클래스에서 슈퍼클래스의 메소드를 재정의 하는 것.
2. 오버라이딩 수행 시, 서브클래스에서 오버라이딩 한 메소드가 실행됨을 보장한다. (상속해준 슈퍼클래스의 함수 정의를 무시함.)

동적 바인딩

1. 실행할 메소드를 실행 도중에 결정짓는 것.

2. 레퍼런스타입 변수에 클래스 객체를 동적 생성하여 집어넣은 경우, 해당 레퍼런스에서 '.'으로 멤버를 호출한다면, 동적 바인딩이 일어난다. (실행 도중에 동적 생성된 객체의 코드를 호출한다.)

super키워드

1. 자바에는 'super'가 슈퍼클래스의 레퍼런스로 정의되어 있다.
2. 즉, 특정한 서브클래스 내부에서 super키워드 뒤에 '.'을 붙이는 것으로, 슈퍼클래스의 멤버에 접근할 수 있다는 것이다.
3. 이때, super키워드는 컴파일러에 의하여 미리 예비되는 주소를 가리키므로, 정적 바인딩이라 볼 수 있다. (실행 이전에 사용자에 의하여 지시된다는 점에서도 정적 바인딩이라고 볼 수 있다.)

추상 클래스: 추상 메소드를 가지거나, abstract키워드로 선언된 클래스.

1. 추상 메소드: abstract키워드로 선언된 메소드와, 원형만 선언되고 내용이 정의되지 않은 메소드를 추상 메소드라 칭한다.
2. 추상 메소드로 인하여 인스턴스 생성이 불가능한 클래스를 추상 클래스라고 하는 것이다.
3. 추상 클래스를 상속받는 경우, 상속받은 서브클래스도 추상 클래스로 선언해야 한다. 이때 서브클래스에서 모든 추상 메소드를 오버라이딩 한 경우, 서브클래스의 인스턴스가 생성 가능해진다. (이를 추상 클래스의 구현이라 한다.)
4. abstract키워드는 class 키워드 앞에 쓰면 된다.
5. abstract class A { abstract public int add(int x, int y); }

   //위: 추상 메소드를 가지는 추상 클래스의 선언 예시.

인터페이스: 입출력을 위한 규격.

1. interface키워드를 사용하여 클래스를 선언하는 것과 동일한 방법으로 선언한다.
2. 클래스가 구현해야 할 메소드들이 선언된 추상형을, 인터페이스 클래스라 한다.
3. 인터페이스 클래스는 내부에 멤버 변수를 선언할 수 없다. (그래서 추상형이다.)
4. 인터페이스는 다음의 다섯 가지 요소를 포함할 수 있다.
    A. 상수
    B. 추상 메소드
    C. default 메소드 <- 코드가 작성되어 있어야 함. (정의되어 있어야 함.)
    D. private 메소드 <- 코드가 작성되어 있어야 함. (정의되어 있어야 함.)
    E. static 메소드 <- 코드가 작성되어 있어야 함. (정의되어 있어야 함.)
5. 인터페이스를 상속받은 클래스를 작성할 때는, extends 대신, implements 키워드를 사용하여 상속받아야 한다.

예시:

```
interface PhoneInterface{
    final int TIMEOUT = 10000;  //상수.
    default void printLogo(){ ... }  //default 메소드.
    //기타 등등 정의...
}

class SamsungPhone implements Phoneinterface {
    //클래스 정의...
}
```

9주차 수업 정리 끝.

10주차 수업 정리.

자바는 서로 다른 디렉터리에 코드를 모아서, 각 디렉터리간 식별자의 중복을 허용한다. (동일한 이름이 다른 디렉터리에 존재 가능. + 이 디렉터리를 '패키지'라 한다.)

패키지: 서로 관련된 인터페이스와 클래스를 컴파일한 클래스 파일들을 모아 놓은 디렉터리.

모듈: 여러 패키지와 프로그램에 필요한 모든 것을 모아서 지칭하는 용어. (이미지든 바이너리 코드든 무엇이든!)

1. 다른 말로 하면, 모듈은 패키지들을 담는 컨테이너다.
2. .jmod파일을 모듈 파일이라고 칭한다.
3. jmod extract ".jmod파일의 절대경로" 〈- 이 명령어로 모듈의 압축을 해제할 수 있다.
4. 모듈화를 성공하면 유지보수가 쉽다. 단, 작은 프로그램에서 굳이 모듈화를 할 이유는 없다.
5. JAVA9부터 자바 API가 모듈화 되어서, 필요한 모듈만 실행 이미지에 넣을 수 있다. -〉 메모리 효율성 증가. 실행 이미지 경량화.
6. jmods 디렉터리에 zip포맷으로 압축된 .jmod 확장자인 99개의 모듈 파일이 존재한다. (JDK 10 기준.)
7. 자주 사용하던 java.util.scanner 를 설명하면,
   A. java디렉터리의 util패키지의 scanner클래스를 말하는 것.

주요 패키지들:

1. java.lang - 가장 기본적인 것 들 포함.
2. java.util - 날짜, 시간, 벡터 등등의 유틸리티 포함.
3. java.io - 입출력
4. java.awt - GUI용 인터페이스
5. java.swing - 스윙 GUI

클래스를 소스코드에서 접근 가능하도록 할 때의 방법들.

1. 전체 경로 명시 – java.util.Scanner scanner = new java.util.Scanner……; (누구도 이렇게 안함.)
2. import 키워드로 삽입.
    A. import java.util.Scanner; –〉 이후 Scanner s = new Scanner(System.in);
3. 패키지의 모든 클래스 불러오기.
    A. import java.util.*;

패키지 선언 〈- 컴파일된 .class 파일을 저장할 패키지 이름 선언.

1. 클래스가 소속될 패키지 이름은 package 키워드로 선언 가능.
    A. package 패키지 명;
    B. .java파일을 컴파일 한 .class 파일은 반드시 어떤 패키지에 소속되어야 한다. 위의 선언문으로 파일을 특정한 패키지에 소속시키면 된다.

Object 클래스: 모든 클래스가 묵시적으로 상속받는 최상위 슈퍼클래스.

1. 모든 클래스가 지녀야 하는 기능을 포함하는 클래스다.
2. .toString( ): 객체의 정보를 문자열로 리턴하는 메소드.
    A. 〈객체+문자열〉 연산시, 컴파일러가 자동으로 〈객체.toString( )+문자열〉 로 변환한다.
    B. 이 메소드의 기본 리턴 값은 보기 불편하므로, 적절히 필요한 기능을 직접 오버라이딩으로 구현하면 된다.
3. .equals( ): 두 객체가 갖고 있는 레퍼런스가 동일한지 비교한다.
    A. 예: str1.equals(str2) 〈- 이것 자체가 참, 거짓 리턴함.

Wrapper 클래스: 자바의 기본 타입을 클래스화 한 8개 클래스를 지칭하는 용어.

1. 이런 이름의 클래스가 실존하는 것은 아니다. 8개의 클래스를 지칭하는 관습적인 용어인 것이다.
2. 객체가 어떤 값을 편하게 처리할 수 있도록, 값을 객체로 바꾸기 위해 사용한다. 사용 예시 – p.264.
3. 개인적으로 생각하기에는 굉장히 추한 방식이다. 차라리 파이썬처럼 모든 변수를 레퍼런스 타입으로 했다면? 차라리 C++처럼 연산자 오버로딩을 더 자유롭게 만들어서 객체와 기본타입의 연산을 쉽게 처리할 수 있도록 했다면? 답답하다….
4. boxing: 기본 타입을 Wrapper객체로 만드는 것.
5. unboxing: Wrapper –〉 기본 타입.

```
Integer ten = Integer.valueOf(10);  //boxing
int n = ten.intValue();  //unboxing

//이런 방식으로 작성해도 알아서 컴파일러가 위처럼 바꿔서 해석한다.
Integer ten = 10;
int n = ten;
```

String 클래스 – p.266

1. 사용법은 간단하니 책을 열어보라.
2. new String( )으로 String 객체 생성시 힙 영역에 저장되고, 각 리터럴은 공유되지 않는다. (메모리 공간에서 동일 문자열도 따로 취급된다는 말. = 하나의 문자열은 단 하나의 레퍼런스 타입 변수만 가리킬 수 있음.)
3. 미리 선언된 String 객체는 JVM의 ‘스트링 리터럴 테이블’에 저장되며, 각 리터럴은 공유된다. (정적으로 선언된 String 객체는 동일 문자열이 여러 객체에 공유된다. = 여러 레퍼런스 타입 변수가 메모리상의 동일 문자열을 가리킬 수 있다.)
4. String 객체간 비교 연산자 사용이 불가능하다.
5. 교수님이 언급하신 메소드 목록. – p.268
   A. equals( ): 같은지 비교.

B. compareTo( ): 같은지 비교.

C. trim( ): 문자열 앞뒤의 공백 제거.

D. length( ): 길이 반환.

E. toLowerCase( ) / toUpperCase( ): 대소문자 변환.

StringBuffer 클래스: 내부에 가변 크기의 버퍼를 지닌 String 클래스.

1. C++의 string 클래스와 유사하다.
2. String 클래스와 달리, 문자열의 수정이 가능하다.

StringTokenizer 클래스: 문자열을 특정 토큰을 기준으로 쪼갤 때 사용.

1. StringTokenizer st = new StringTokenizer(String객체이름, “토큰.”);
2. 메소드 nextToken( )으로 st 내부의 문자열 조각들을 읽을 수 있다. (Scanner와 유사하게 동작.)

10주차 수업 정리 끝.

11주차 수업 정리.

제네릭: 프로그래밍 기법 중 하나. 다양한 자료형을 기반으로 코드를 일반화 한다.

1. 타입 매개 변수를 사용하여 정의한다.
2. '<E>'같은 모양으로 선언하며, 이때 '<'와 '>'는 연산자이고, E는 매개변수다.

컬렉션: 가변 개수의, 복수 자료의 모임인 자료형. (객체들의 컨테이너.)

1. 컬렉션은 클래스로 구현되어 있다. 삽입, 삭제 등의 모든 기능을 각 클래스의 메소드로 구현해 놓은 것이다.
2. 다양한 컬렉션이 있는데 필요할 때 찾아서 사용하면 될 것이다.

Vector<E>: 배열을 강화한 것. 가변크기 + 삽입, 삭제에 따른 요소 위치 자동 조정.

1. 객체, null, Wrapper타입의 기본타입을 삽입 가능.
2. 생성 문법: Vector<Integer> v = new Vector<Integer>( );
   A. 생성자에 크기를 인자로 넘긴다. (디폴트 값 = 10.)
3. p.295의 메소드 목록을 한번 열어보라. 긴 시간을 들여서 설명하셨다.
4. .add(값) / .remove(인덱스) / .size( ) – 요소 개수 / .capacity( ) – 크기 / .lastElement( ) – 마지막 요소

ArrayList<E>: Vector<E>와 거의 유사함. (스레드 동기화의 지원 여부만 다름)

1. ArrayList<E>는 스레드 동기화를 수동으로 해야 함.
2. Vector<E>는 자동으로 스레드 동기화 수행함.

Iterator: 컬렉션의 순차 검색을 위한 인터페이스.

➔ 엄청 짧게 설명하고 넘어가셨다. 별로 중요하게 여기지 않으시는 듯 하다. 개인적으로 동의한다.

➔ 이 타입의 객체를 만들고, 컬렉션의 위치를 대입해서 사용한다. 다른 언어와 유사하다.

HashMap<K, V>: 키와 값을 대응시켜 저장하는, 해시맵을 구현한 클래스.

1. put( ) / get( ) / remove( ) 메소드로 삽입, 읽기, 삭제 수행.
2. size( )의 경우는, 한 쌍을 하나로 인식한다. 키와 값은 한 덩어리다.

```
HashMap<String, String> h = new HashMap<String, String>( );
h.put("apple", "사과");  //put() 예시.
//순서 상관 없이 h 객체 내부의 빈 공간에 적당히 들어간다.

String kor = h.get("apple");  //get() 예시. "사과"가 저장됨.

h.remove("apple");  //remove() 예시.
```

제네릭 클래스 선언 예시

```
public class MyClass<T>{
    T val;  //T 타입 변수 선언.
    void set(T a){ val = a; }  //T 타입 매개변수 사용 가능.
}
```

11주차 수업 정리 끝.

12주차 수업 정리.

12주차 수업은 Swing 수업이었는데, 해당 내용이 담긴 8장은 시험에 출제하지 않으신다고 하셨다.

12주차 수업 정리 끝.

13주차 수업 정리.

이벤트 기반 프로그래밍 / 배치 프로그래밍

1. 이벤트 기반: 이벤트에 의하여 프로그램의 실행을 제어.
2. 배치(batch): 개발자가 프로그램 흐름을 제어.

이벤트의 종류

1. 사용자 입력: 마우스, 키보드…….
2. 센서 입력: 조도센서, 가속센서…….
3. 다른 응용 프로그램이나 스레드의 입력.

이벤트 처리 방법: 이벤트 리스너의 등록으로 구현한다.

1. 모든 이벤트는 특정한 이벤트 리스너를 보유한다.
2. 이때, 이벤트 리스너는 '루틴'이라고도 한다.

이벤트 객체: 발생한 이벤트에 대한 여러 정보를 포함하는 객체.

1. 이벤트 발생시에, 이벤트 객체가 생성된다. (JVM이 만들어 준다.)
2. 이벤트마다 만들어지는 객체의 종류가 다르다. - 〈p.358, 표 9-1〉
3. 이벤트 종류, 이벤트 소스, 화면 좌표, 마우스 버튼 종류, 눌려진 키, 등등…의 다양한 정보를 포함한다. 그리고 이 정보들을 객체의 메소드로 가져올 수 있다.
   A. 예: getSource( )

이벤트 리스너: 이벤트 리스너 인터페이스를 상속받고, 추상 메소드를 구현한, 이벤트 처리 용 클래스.

1. 이벤트 리스너 인터페이스가 제공하는 추상 메소드를 사용자가 원하는 대로 정의한 클래스를 말한다.

2. 상속받아 정의할 수 있는 추상 메소드의 목록은 p.360의 표 9-2 참고.
3. 작성 방법은 세 가지가 있다.
   A. 독립 클래스: 여러 곳에서 리스너 사용시, 독립 클래스로 선언.
   B. 내부 클래스: 특정 클래스 안에서만 리스너 사용시, 내부 클래스로 선언.
   C. 익명 클래스: 리스너 코드가 아주 간단한 경우 사용.
4. 어댑터 클래스: 복수의 추상 메소드를 갖는 일부 리스너 인터페이스를 대체하여, 리스너 인터페이스의 구성 요소 중 하나만(단일 추상 메소드) 리턴하는 클래스.
   A. 이는 리스너 인터페이스의 다양한 메소드의 대부분이 필요치 않을 때, 하나씩 받아서 오버라이딩 할 수 있도록 구현된 클래스이다.
   B. 오버라이딩 하지 않을 추상 메소드의 코드 무더기를 모두 상속받으면, 리소스 낭비가 심하다. 그래서 어댑터 클래스를 사용한다.

키 이벤트(어떤 KeyListener 의 메소드가 실행되는지) - p.371

1. 키를 누르는 순간. - void keyPressed(KeyEvent e)
2. 키를 떼는 순간. - void KeyReleased(KeyEvent e)
3. 유니코드 키를 떼는 순간. - void KeyTyped(KeyEvent e)

키 입력시 호출되는 이벤트 리스너 종류

1. 유니코드 키 입력시: KeyPressed -> KeyTyped -> KeyReleased 호출.
2. 유니코드 아닌 키 입력시: KeyPressed ------→ KeyReleased 호출.

마우스 이벤트 처리.

1. 드래그시 mouseDragged( )가 드래그 도중에 계속 반복 호출되는 것 주의.
2. 리스너 종류
   A. MouseListener: 1회에 한하는 동작들만 처리.
   B. MouseMotionListener: 마우스 움직임과 드래그 처리.

이벤트 리스너 예제 코드 – p.363

```
JButton btn = new JButton("Action");   //버튼 생성.
btn.addActionListener(new MyActionListener());  //버튼에 이벤트 리스너 장착.

~~~~~

private class MyActionListener implements ActionListener{
//내부 클래스로 리스너 정의.
    public void actionPerformed(ActionEvent e){
    //ActionEvent 발생시 이 함수 호출됨.
        jButton b = (JButton)e.getSource();
        //이벤트가 발생한 버튼의 정보 가져옴.

        //리스너 동작 정의.
    }
}
```

익명 클래스로 이벤트 리스너 등록 예시. – p.365

```
JButton btn = new JButton("Action");   //버튼 생성.
btn.addActionListener(new MyActionListener(){
//익명 클래스를 정의와 동시에 동적 생성.
    public void actionPerformed(ActionEvent e){
    //ActionEvent 발생시 이 함수 호출됨.
        jButton b = (JButton)e.getSource();
        //이벤트가 발생한 버튼의 정보 가져옴.
        //리스너 동작 정의.
    }
});
```

13주차 수업 정리 끝.

이상으로 시험 범위의 모든 내용을 정리했다. (*)