2024학년도 1학기

신입생 Java 교육

교육부장 20 이어진

교육 커리큘럼

1주차 변수,연산자 + 조건문(if)
2주차 조건문(switch) + 반복문
3주차 배열

<중간고사>

4주차 객체, 메소드 오버로딩과 생성자
5주차 상속과 오버라이딩
6주차 다형성과 인터페이스
7주차 예외처리

<기말고사>

Part 1

상속

Part 1

상 속

부모가 가진 것을 자식에게 물려주는 것

노트북은 컴퓨터의 한 종류다.

침대는 가구의 한 종류다. 혹은 침대는 가구다.

소방차는 자동차다.

이렇게 말할 수 있는 관계를 is a 관계 혹은 kind of 관계라고 한다.

Part 1 상속 예시

```
class Car {

}

class Bus extends Car {

}
```

자바는 클래스 이름 뒤에 extends 키워드를 적고 부모 클래스 이름을 적게 되면 부모 클래스가 가지고 있는 것을 상속받을 수 있게 된다.

상속이란 부모가 가지고 있는 것을 자식이 물려받는 것을 말한다. 즉, 부모가 가지고 있는 것을 자식이 사용할 수 있게 된다.

Part 1 상속 예시

```java
public class Main {
    Run | Debug
    public static void main(String[] args) {
        Bus bus = new Bus();
        bus.run();
    }
}

class Car {
    public void run() {
        System.out.println("Car의 run 메서드");
    }
}

class Bus extends Car {

}
```

```
Car의 run 메서드
```

Bus class 는 아무런 코드를 가지지 않는다.
그럼에도 run 이라는 메소드를 사용하는데 문제가 발생되지 않는다.

Part 2

오버라이딩

Part 2

오 버 라 이 딩

부모가 가지고 있는 메소드와
똑같은 모양의 메소드를 자식이 가지고 있는 것.

즉, 메소드를 재정의 하는 것.

Part 2

오버라이딩 조건

메소드의 이름, 매개변수, 리턴타입이 같아야 한다.

```java
class Car {
    public void run() { }
}

class Bus extends Car {
    public void run() { } // 오버라이딩
    public void runrun() { } // 오버라이딩 X (메소드 이름이 다름)
    public void run(int n) { } // 오버라이딩 X (매개변수 타입이 다름)
    public int run() { return 1; } // 오버라이딩 X (리턴 타입이 다름)
}
```

Part 2 오버라이딩 예시

```java
public class Main {
    Run | Debug
    public static void main(String[] args) {
        Bus bus = new Bus();
        bus.run();
    }
}

class Car {
    public void run() {
        System.out.println("Car의 run 메서드");
    }
}

class Bus extends Car {
    @Override
    public void run() {
        System.out.println("Bus의 run 메서드");
    }
}
```

부모(Car class)의 run()과 같은 이름의 메소드를 자식(Bus class)에서 오버라이딩

=> bus의 run()을 실행하면 Car의 run()이 아닌 Bus의 run()이 실행된다.

@Override를 쓰는 이유?

⇒ 오버라이딩을 할 때 메소드 이름에 오타가 있을 경우 발견, 오버라이딩하는 메소드가 부모 메소드에 없다면 컴파일러 오류 발생.

⇒ 필수는 아니지만, 권장

Part 2 super

```java
public class Main {
    Run | Debug
    public static void main(String[] args) {
        Bus bus = new Bus();
        bus.run();
    }
}

class Car {
    public void run() {
        System.out.println("Car의 run 메서드");
    }
}

class Bus extends Car {
    @Override
    public void run() {
        super.run();
        System.out.println("Bus의 run 메서드");
    }
}
```

```
Car의 run 메서드
Bus의 run 메서드
```

오버라이딩 한다고 해서 부모의 메소드가 사라지는 것은 아니다.

super 키워드를 사용하면, 부모의 메소드를 호출 할 수 있다.

Part 2 오버로딩 vs 오버라이딩

오버로딩(overloading) - 기존에 없는 새로운 메서드를 정의하는 것(new)
오버라이딩(overriding) - 상속받은 메서드의 내용을 변경하는 것(change, modify)

```
class Parent {
    void parentMethod() {}
}

class Child extends Parent {
    void parentMethod() {}          // 오버라이딩
    void parentMethod(int i) {}     // 오버로딩

    void childMethod() {}
    void childMethod(int i) {}      // 오버로딩
    void childMethod() {}           // 에러!!! 중복정의임
}
```

Part 3

패키지

Part 3 패 키 지 (p a c k a g e)

패키지란 서로 관련이 있는 클래스 또는 인터페이스들을 묶어 놓은 묶음이다

Part 3 패키지의 선언


```
package first.second;

public class test {

}
```

- 패키지는 소스파일에 첫 번째 문장(주석 제외)으로 단 한 번 선언한다.
- 하나의 소스파일에 둘 이상의 클래스가 포함 된 경우, 모두 같은 패키지에 속하게 된다. (하나의 소스파일에 단 하나의 public 클래스만 허용)
- 모든 클래스는 하나의 패키지에 속하며, 패키지가 선언되지 않은 클래스는 자동적으로 이름없는(unnamed) 패키지에 속하게 됨.

```
package first.second;

public class test {

}

public class test2 {

}
```

```
test.java:7: error: class test2 is public, should be declared in a file named test2.java
public class test2 {
       ^
1 error
```

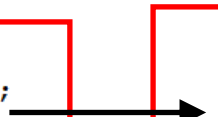
Part 4

import

Part 4 import문

- 사용할 클래스가 속한 패키지를 지정하는데 사용
- import문을 사용하면 클래스를 사용할 때 패키지명을 생략할 수 있다.

```
class ImportTest {
    java.util.Date today = new java.util.Date();
    // ...
}
```



```
import java.util.*;

class ImportTest {
    Date today = new Date();
}
```

- java.lang 패키지의 클래스는 import 하지 않고도 사용할 수 있다.
(String, Object, System, Thread, …)

```
import java.lang.*;

class ImportTest2
{
    public static void main(String[] args)
    {
        System.out.println("Hello World!");
    }
}
```

```
public static void main(java.lang.String[] args)
{
    java.lang.System.out.println("Hello World!");
}
```

Part 4

import문의 선언

- import문은 패키지문과 클래스선언의 사이에 선언한다.

일반적인 소스파일(*.java)의 구성은 다음의 순서로 되어 있다.

① package문
② import문
③ 클래스 선언

- import문을 선언하는 방법은 다음과 같다.

```
import 패키지명.클래스명;
  또는
import 패키지명.*;
```

```java
1  package com.javachobo.book;
2
3  import java.text.SimpleDateFormat;
4  import java.util.*;
5
6  public class PackageTest {
7      public static void main(String[] args) {
8          // java.util.Date today = new java.util.Date();
9          Date today = new Date();
10         SimpleDateFormat date = new SimpleDateFormat("yyyy/MM/dd");
11     }
12 }
```

실습문제

[문제]
이름, 나이, 급여를 입력 받고 Person일 때와 Manager일 때를 출력한다.
[조건]
· Manager 클래스는 Person 클래스를 상속받는다.
· 보너스는 급여의 0.5배이다.
· super 사용

입력 예시

```
이름: Admin
나이: 20
급여: 2000
```

출력 예시

```
===== Person =====
이름: Admin
나이: 20
급여: 2000
===== Manager =====
이름: Admin
나이: 20
급여: 2000
보너스: 1000.0
```

해설

```java
import java.util.Scanner;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("이름: ");
        String name = sc.next();
        System.out.print("나이: ");
        int age = sc.nextInt();
        System.out.print("급여: ");
        int pay = sc.nextInt();

        System.out.println("===== Person =====");
        Person p = new Person(name, age, pay);
        p.print();

        System.out.println("===== Manager =====");
        Manager m = new Manager(name, age, pay);
        m.print();
    }
}
```

```java
class Person {
    String name;
    int age;
    int pay;

    Person(String name, int age, int pay) {
        this.name = name;
        this.age = age;
        this.pay = pay;
    }

    void print() {
        System.out.println("이름: " + name);
        System.out.println("나이: " + age);
        System.out.println("급여: " + pay);
    }
}

class Manager extends Person {
    Manager(String name, int age, int pay) {
        
    }

    void print() {
        
    }
}
```

super를 사용해서 초기화

super를 사용해서 출력

해설

```java
import java.util.Scanner;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("이름: ");
        String name = sc.next();
        System.out.print("나이: ");
        int age = sc.nextInt();
        System.out.print("급여: ");
        int pay = sc.nextInt();

        System.out.println("===== Person =====");
        Person p = new Person(name, age, pay);
        p.print();

        System.out.println("===== Manager =====");
        Manager m = new Manager(name, age, pay);
        m.print();
    }
}
```

```java
class Person {
    String name;
    int age;
    int pay;

    Person(String name, int age, int pay) {
        this.name = name;
        this.age = age;
        this.pay = pay;
    }

    void print() {
        System.out.println("이름: " + name);
        System.out.println("나이: " + age);
        System.out.println("급여: " + pay);
    }
}

class Manager extends Person {
    Manager(String name, int age, int pay) {
        super(name, age, pay);
    }

    void print() {
        super.print();
        System.out.println("보너스: " + pay * 0.5);
    }
}
```

super를 사용해서 초기화

super를 사용해서 출력

Q&A

감사합니다