﻿1. 상속, 캡슐화, 다형성, 추상화에 대하여 설명하시오.

상속 : 기존의 코드를 재활용하기 위한 기법으로 이미 작성된 클래스(부모클래스)를 이어받아서 새로운 클래스(자식클래스)를 생성하는 것

캡슐화 : 서로 관련된 데이터와 알고리즘을 캡슐에 넣어서 포장하는 것. 관련 있는 데이터와 알고리즘이 묶여 있으면 사용하기가 편리해지고 객체의 실제 구현 내용을 외부에 감추기 위해(정보은닉)사용

다형성 : 객체가 취하는 동작이 상황에 따라서 달라지는 것 서로 다른 타입에 속하는 객체들이 같은 이름의 멤버 함수에 응답하여서 서로 다른 동작을 보여주는 것이 가능

추상화 : 불필요한 정보는 숨기고 중요한 정보만을 표현함으로써 프로그램을 간단히 만드는 것.

공통적인 부분을 뺀 뒤 하위 클래스에서 추상적 기능을 다시 맞게 구현함으로써 코드의 재사용을 높임

2.

<코드>

**public** **class** 문제1 **implements** Transport{

**public** **static** **void** main(String[] args) {

Transport[] tp = **new** Transport[3];

tp[0] = **new** Bus("Bus", 2);

tp[1] = **new** Truck("Truck", 4);

tp[2] = **new** Ship("Ship", 200);

**for** (**int** i = 0; i<3; i++) {

**for**(**int** j=0; j<3; j++)

tp[i].ride();

tp[i].move();

tp[i].stop();

}

}

**public** **void** ride() {

System.***out***.println("탔습니다.");

}

**public** **void** stop() {

System.***out***.println("멈춥니다.");

}

**public** **void** move() {

System.***out***.println("출발힙니다.");

}

**public** **void** getoff() {

System.***out***.println("하차합니다.");

}

}

**interface** Transport{

**public** **void** ride();

**public** **void** stop();

**public** **void** move();

**public** **void** getoff();

}

**class** Bus **extends** 문제1{

String type = "";

**int** limit = 0;

**int** Count = 0;

**public** Bus(String type, **int** size){

**this**.type = type;

limit = size;

}

**public** **void** ride()

{

**if**(Count >= limit)

System.***out***.println(type+"가 만원입니다.");

**else** {

System.***out***.println(type+"에 승객이 탔습니다.");

Count++;

}

}

**public** **void** stop()

{

System.***out***.println(type+"가 멈춥니다.");

}

**public** **void** move() {

System.***out***.println(type+"가 출발합니다.");

}

**public** **void** getoff() {

**if**(Count <= 0)

System.***out***.println(type+"에 타고있는 승객이 없습니다.");

**else** {

System.***out***.println(type+"에 승객이 내렸습니다.");

Count--;

}

}

}

**class** Truck **extends** 문제1{

String type = "";

**int** limit = 0;

**int** Count = 0;

**public** Truck(String type, **int** size){

**this**.type = type;

limit = size;

}

**public** **void** ride()

{

**if**(Count >= limit)

System.***out***.println(type+"에 짐이 가득 찼습니다.");

**else** {

System.***out***.println(type+"에 짐을 실었습니다.");

Count++;

}

}

**public** **void** stop() {

System.***out***.println(type+"이 멈춥니다.");

}

**public** **void** move() {

System.***out***.println(type+"이 출발합니다.");

}

**public** **void** getoff() {

**if**(Count <= 0)

System.***out***.println(type+"에 짐이 없습니다.");

**else** {

System.***out***.println(type+"에 짐을 뺐습니다.");

Count--;

}

}

}

**class** Ship **extends** 문제1{

String type = "";

**int** limit = 0;

**int** Count = 0;

**public** Ship(String type, **int** size){

**this**.type = type;

limit = size;

}

**public** **void** ride()

{

**if**(Count>=limit)

System.***out***.println(type+"가 만원입니다.");

**else** {

System.***out***.println(type+"에 승객이 탔습니다.");

Count++;

}

}

**public** **void** stop() {

System.***out***.println(type+"가 멈춥니다.");

}

**public** **void** move() {

System.***out***.println(type+"가 출발합니다.");

}

**public** **void** getoff() {

**if**(Count <= 0)

System.***out***.println(type+"에 타고있는 승객이 없습니다.");

**else** {

System.***out***.println(type+"에 승객이 내렸습니다.");

Count--;

}

}

}

<실행결과>

