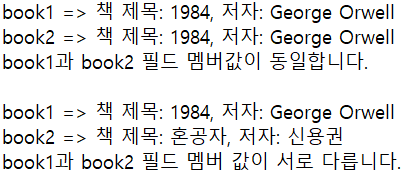
|  |  |  |
| --- | --- | --- |
| java2\_week05\_상속(1) | 학번 : 20237107 | 이름 : 하태영 |

* 응용 프로그래밍

1. **[생성형 AI 활용]** 제시된 main()과 결과를 참고하여 Book 객체의 동일성 여부(내용)를 판단하는 Book 클래스를 작성하세요. 단, equals() 메소드를 재정의하지 않으며 필드는 전용 멤버이어야 하고, 객체 내부에서 비교할 수 있도록 합니다



public static void main(String[] args) {

Book book1 = new Book("1984", "George Orwell"); //책 제목과 지은이를 매개변수로 전달합니다.

Book book2 = new Book("1984", "George Orwell");

Book book3 = new Book("혼공자", "신용권");

*isCheck*(book1, book2);

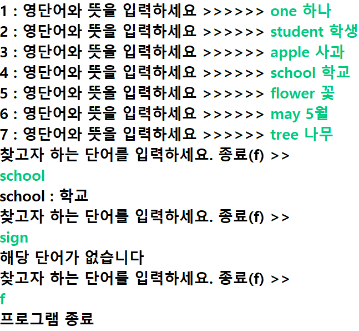
*isCheck*(book2, book3);

}

|  |  |
| --- | --- |
| 생성형 AI 플랫폼 | chatGPT |
| 질문 | public static void main(String[] args) { Book book1 = new Book("1984", "George Orwell"); //책 제목과 지은이를 매개변수로 전달합니다. Book book2 = new Book("1984", "George Orwell"); Book book3 = new Book("혼공자", "신용권"); isCheck(book1, book2); isCheck(book2, book3); }  단, equals() 메소드를 재정의하지 않으며 필드는 전용 멤버이어야 하고, 객체 내부에서 비교할 수 있도록 합니다 |
| 답변 | public class Book {  private String title;  private String author;  // Constructor  public Book(String title, String author) {  this.title = title;  this.author = author;  }  // Getter methods for title and author  public String getTitle() {  return title;  }  public String getAuthor() {  return author;  }  // 내부에서 비교할 수 있는 메소드  public boolean isEqual(Book otherBook) {  // this 객체의 title과 author와 비교  return this.title.equals(otherBook.getTitle()) && this.author.equals(otherBook.getAuthor());  }  } |
| 실행결과 |  |

1. **[생성형 AI 활용]** 상속관계가 적용된 간단한 예를 제시하고, 어떤 경우에 상속 관계를 활용하는지 설명하세요

|  |  |
| --- | --- |
| 생성형 AI 플랫폼 | chatGPT |
| 질문 | 상속관계가 적용된 간단한 예를 제시하고, 어떤 경우에 상속 관계를 활용하는지 설명하세요 |
| 답변 | // 부모 클래스: Animal  class Animal {  String name;  // 생성자  public Animal(String name) {  this.name = name;  }  // 공통된 행동 (메소드)  public void makeSound() {  System.out.println("Some sound");  }  }  // 자식 클래스: Dog  class Dog extends Animal {  // 생성자  public Dog(String name) {  super(name); // 부모 클래스 생성자 호출  }  // 메소드 오버라이드: 개는 '멍멍' 소리를 냅니다.  @Override  public void makeSound() {  System.out.println(name + " says: Woof!");  }  }  // 자식 클래스: Cat  class Cat extends Animal {  // 생성자  public Cat(String name) {  super(name); // 부모 클래스 생성자 호출  }  // 메소드 오버라이드: 고양이는 '야옹' 소리를 냅니다.  @Override  public void makeSound() {  System.out.println(name + " says: Meow!");  }  }  public class Main {  public static void main(String[] args) {  Animal animal = new Animal("Generic Animal");  animal.makeSound();  Dog dog = new Dog("Buddy");  dog.makeSound();  Cat cat = new Cat("Whiskers");  cat.makeSound();  }  }  **상속을 활용하는 경우:**   1. **코드 재사용**:    * 부모 클래스에서 정의된 공통적인 속성과 메소드를 자식 클래스들이 상속받아 반복적인 코드를 줄일 수 있습니다.    * 예: Animal 클래스에서 공통된 행동인 makeSound() 메소드를 정의하고, 이를 자식 클래스들이 오버라이드하여 각자의 특성에 맞게 사용. 2. **유지보수 용이**:    * 상속을 사용하면 공통된 코드가 부모 클래스에 있기 때문에, 변경이 필요할 경우 부모 클래스만 수정하면 되므로 유지보수가 쉬워집니다. 3. **다형성**:    * 부모 클래스의 참조 변수로 자식 클래스 객체를 다룰 수 있어, 코드의 유연성을 높일 수 있습니다. 예를 들어, Animal animal = new Dog("Buddy");와 같이 사용하여 Dog 객체가 Animal 타입으로 다뤄지도록 할 수 있습니다. 4. **확장성**:    * 새로운 동물 클래스(예: Bird, Fish)를 추가할 때, 부모 클래스인 Animal을 상속받고 필요한 메소드만 오버라이드하면 되어 코드 확장이 용이합니다.   **결론:**  상속은 코드의 재사용성과 확장성을 높이고, 유지보수와 다형성을 통해 코드의 효율성을 증가시킬 수 있습니다. Animal 클래스처럼 공통적인 속성이나 행동을 부모 클래스로 정의하고, 자식 클래스에서 이를 특화시켜 사용하는 방식은 객체지향 프로그래밍에서 매우 중요한 개념입니다. |
| 실행결과 |  |

1. 키보드로 영 단어를 입력하면 단어 뜻을 한글로 출력하는 프로그램을 작성하세요. 영 단어와 한글은 객체 배열 생성시 초기화 합니다

class Word{

//영어 단어 저장을 위한 필드, private

//영 단어에 대한 뜻을 저장하는 필드, private

//생성자 – 매개변수로 받은 값을 필드로 초기화

//모든 필드에 대한 getter 메소드

//매개변수로 받은 문자열이 현재 객체가 가지는 영 단어와 일치하는지 검사하여 true, false를 반환하는 wordComp() 메소드

//객체 내용을 문자열로 반환하는 toString() 메소드

}

class WordTest{one

public static void main(String[] args){

//객체 배열을 선언 & 생성, 배열 크기는 7- 본인 작성

//입력한 값으로 배열 초기화 - 본인 작성

//wordComp() 메소드를 사용하여 객체 배열에 저장된 영 단어와 입력된 영단어와 일치하면 그에 해당하는 뜻을 출력 - 본인 작성

}

}

|  |
| --- |
| [프로그램 소스]  package week4.practice3;  class Word{  //영어 단어 저장을 위한 필드, private  //영 단어에 대한 뜻을 저장하는 필드, private  private String eng, kor;  //생성자 – 매개변수로 받은 값을 필드로 초기화  public Word(String eng, String kor) {  this.eng = eng;  this.kor = kor;  }  //모든 필드에 대한 getter 메소드  public String getEng() {  return eng;  }  public String getKor() {  return kor;  }    //매개변수로 받은 문자열이 현재 객체가 가지는 영 단어와 일치하는지 검사하여 true, false를 반환하는 engComp() 메소드  public boolean wordComp(String search) {  return this.eng.equals(search);  }  //객체 내용을 문자열로 반환하는 toString() 메소드  @Override  public String toString() {  return "eng [eng=" + eng + ", kor=" + kor + "]";  }    }  package week4.practice3;  import java.util.Scanner;  class WordTest{  public static void main(String[] args){  //객체 배열을 선언 & 생성, 배열 크기는 7- 본인 작성  //입력한 값으로 배열 초기화 - 본인 작성  Word[] dim = new Word[7];  Scanner in = new Scanner(System.*in*);    // 배열의 각 객체 초기화  for(var i=0; i < dim.length; i++) {  System.*out*.print((i+1) + " : 영단어와 뜻을 입력하세요 >>>>> ");      String str = in.nextLine();  String[] a = str.split(" "); // split : 문자열을 잘라주는 함수 => 배열로 처리됨  dim[i] = new Word(a[0], a[1]);    // dim[i] = new Word(in.next(), in.next());    }  //wordComp() 메소드를 사용하여 객체 배열에 저장된 영 단어와 입력된 영단어와 일치하면 그에 해당하는 뜻을 출력 - 본인 작성  while(true) {  boolean flag = true;  System.*out*.println("찾고자 하는 단어를 입력하세요. 종료(f)");    String search = in.next();    if(search.equalsIgnoreCase("f"))  break;    for (Word temp : dim) {  if(temp.wordComp(search)) {  System.*out*.println(search + ":" + temp.getKor());  flag = false;  break;  }  }  if (flag)  System.*out*.println("해당 단어가 없습니다.");  }  System.*out*.println("프로그램 종료");  }    } |
| [실행 결과] |

1. Person 클래스를 설계합니다. Person 클래스는 이름, 주소, 전화번호를 필드로 가집니다. 하나 이상의 생성자를 정의하고 각 필드에 대하여 접근자/설정자 메소드를 작성하세요. Person 클래스를 상속 받아 Customer 클래스를 작성합니다. Customer는 고객 번호와 마일리지를 필드로 가집니다. 한 개 이상의 생성자를 작성하고 접근자/설정자 메소드를 작성하세요. 두개의 클래스 모두 객체 정보를 문자열로 반환하는 toString() 메소드를 정의합니다. 테스트 클래스를 작성하여 위 클래스 타입의 객체를 만들고 각 객체의 모든 정보를 출력하세요.

public class PersonTest {

public static void main(String[] args) {

public static void main(String[] args) {

//이름, 주소, 고객 번호, 마일리지

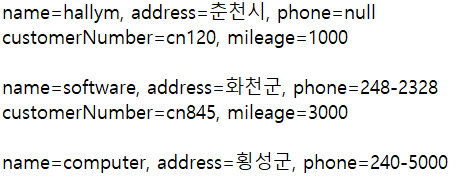
Customer customer1=new Customer("hallym", "춘천시", "cn120", 1000);

//이름, 주소, 전화번호,고객 번호, 마일리지

Customer customer2=new Customer("software", "화천군", "248-2328", "cn845", 3000);

//이름, 주소, 전화번호

Person person= new Person("computer", "횡성군", "240-5000");

 System.***out***.println(customer1);

System.***out***.println(customer2);

System.***out***.println(person);

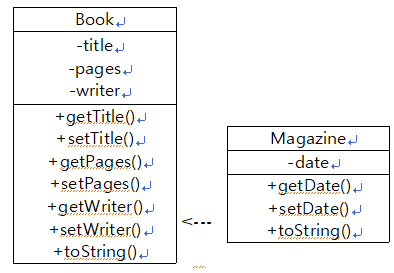
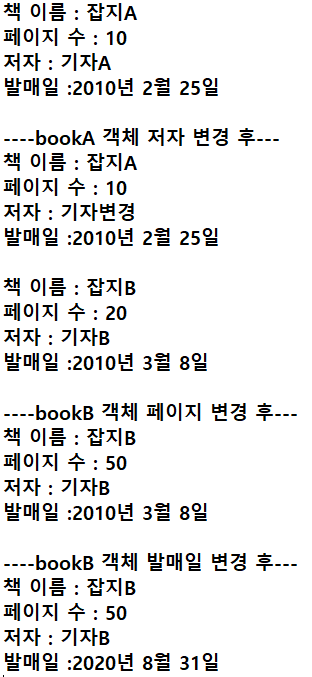
}

}

}

|  |
| --- |
| [프로그램 소스]  package week4.practice4;  class Person {    private String name, address, phone;  public Person(String name, String address, String phone) {  this.name = name;  this.address = address;  this.phone = phone;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getAddress() {  return address;  }  public void setAddress(String address) {  this.address = address;  }  public String getphone() {  return phone;  }  public void setphone(String phone) {  this.phone = phone;  }  @Override  public String toString() {  return "name=" + name + ", address=" + address + ", phone=" + phone + "\n";  }    }  package week4.practice4;  class Customer extends Person {    private String customerNumber;  private int mileage;    public Customer(String name, String address, String tel, String customerNumber, int mileage) {  super(name, address, tel);  this.customerNumber = customerNumber;  this.mileage = mileage;  }  @Override  public String toString() {  String str = super.toString();  return str + "customerNumber=" + customerNumber + ", mileage=" + mileage + "\n";  }  }  package week4.practice4;  public class PersonTest {    public static void main(String[] args) {  //이름, 주소, 고객 번호, 마일리지  Customer customer1=new Customer("hallym", "춘천시", null, "cn120", 1000);    //이름, 주소, 전화번호,고객 번호, 마일리지  Customer customer2=new Customer("software", "화천군", "248-2328", "cn845", 3000);    //이름, 주소, 전화번호  Person person= new Person("computer", "횡성군", "240-5000");  System.*out*.println(customer1);  System.*out*.println(customer2);  System.*out*.println(person);  }  } |
| [실행 결과] |

1. 일반적인 책을 나타내는 Book 클래스를 상속받아서 잡지를 나타내는 Magazine 클래스를 작성합니다. Book 클래스는 제목, 페이지수, 저자 등의 정보를 가집니다. Magazine 클래스는 추가로 발매일 정보를 가집니다. 생성자, 접근자, 설정자를 포함하여서 각각의 클래스를 작성하세요. 이들 클래스들의 객체를 만들고 각 객체의 모든 정보를 출력하는 테스트 클래스를 작성하세요.

**public** **class** Test {

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

Magazine bookA = **new** Magazine("잡지A", 10, "기자A", "2010년 2월 25일");

Magazine bookB = **new** Magazine("잡지B", 20, "기자B", "2010년 3월 8일");

//bookB의 타입을 다음과 같이 Book으로 선언 하였을 때 발생하는 문제는?

//Book bookB = new Magazine("잡지B", 20, "기자B", "2010년 3월 8일");

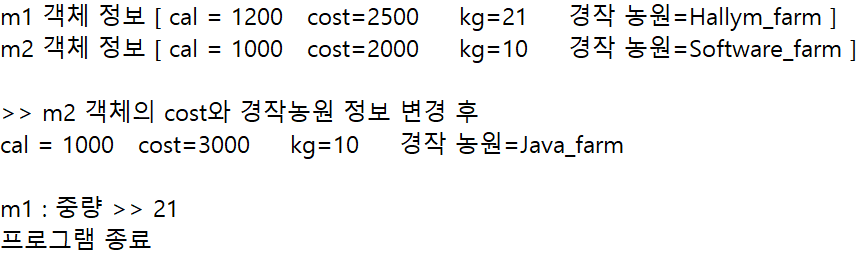
//제시된 결과처럼 출력될 수 있도록 완성 – 본인작성

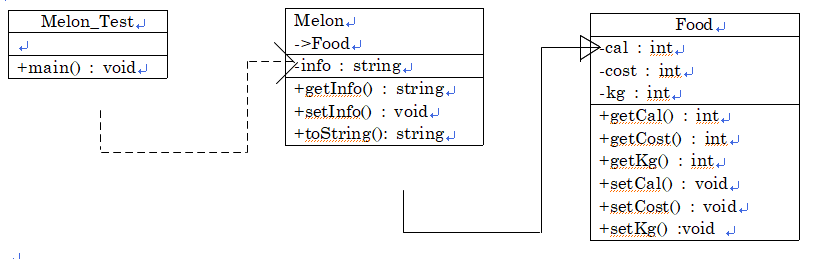
}

}

|  |
| --- |
| [프로그램 소스]  package week4.practice5;  class Magazine extends Book {    // 객체 선언  private String date;    // 생성자  public Magazine(String title, int pages, String writer, String date) {  super(title, writer, pages);  this.date = date;  }  // getter, setter  public String getDate() {  return date;  }  public void setDate(String date) {  this.date = date;  }  // toString  @Override  public String toString() {  return super.toString() + "\n발매일: " + date + "\n";  }    }  package week4.practice5;  class Book {  // 객체 선언  private String title, writer;  private int pages;    // 생성자  public Book(String title, String writer, int pages) {  this.title = title;  this.writer = writer;  this.pages = pages;  }    // getter, setter  public String getTitle() {  return title;  }  public void setTitle(String title) {  this.title = title;  }  public int getPages() {  return pages;  }  public void setPages(int pages) {  this.pages = pages;  }  public String getWriter() {  return writer;  }  public void setWriter(String writer) {  this.writer = writer;  }    // toString  @Override  public String toString() {  return "책이름: " + title + "\n페이지수: " + pages + "\n저자: " + writer;  }    }  package week4.practice5;  public class Test {  public static void main(String[] args) {  Magazine bookA = new Magazine("잡지A", 10, "기자A", "2010년 2월 25일");  Magazine bookB = new Magazine("잡지B", 20, "기자B", "2010년 3월 8일");  //bookB의 타입을 다음과 같이 Book으로 선언 하였을 때 발생하는 문제는?  //Book 타입에서 Magazine 클래스의 고유 속성이나 메서드에 접근할 수 없음  //Book bookB = new Magazine("잡지B", 20, "기자B", "2010년 3월 8일");  //제시된 결과처럼 출력될 수 있도록 완성 – 본인작성  System.*out*.println(bookA);  System.*out*.println("---bookA 객체 저자 변경 후---");  bookA.setWriter("기자변경");  System.*out*.println(bookA);  System.*out*.println(bookB);  System.*out*.println("---bookB 객체 페이지 변경 후---");  bookB.setPages(50);  System.*out*.println(bookB);  System.*out*.println("---bookB 객체 발매일 변경 후---");  bookB.setDate("2020년 8월 31일");  System.*out*.println(bookB);  }  } |
| [실행 결과] |

1. 일반적인 음식을 나타내는 Food 클래스를 상속받아서 멜론을 나타내는 Melon 클래스를 작성합니다. Food 클래스는 칼로리, 가격, 중량 등의 정보를 가집니다. Melon 클래스는 추가로 경작농원정보를 가집니다. 생성자, 접근자, 설정자를 포함하여서 각각의 클래스를 작성합니다. 이들 클래스들의 객체를 만들고 각 객체의 모든 정보를 출력하는 테스트 클래스도 작성합니다.





|  |
| --- |
| [프로그램 소스]  package week4.practice6;  class Food {    private int cal, cost, kg;  public Food(int cal, int cost, int kg) {  super();  this.cal = cal;  this.cost = cost;  this.kg = kg;  }  public int getCal() {  return cal;  }  public void setCal(int cal) {  this.cal = cal;  }  public int getCost() {  return cost;  }  public void setCost(int cost) {  this.cost = cost;  }  public int getKg() {  return kg;  }  public void setKg(int kg) {  this.kg = kg;  }  }  package week4.practice6;  class Melon extends Food {  private String info;  public Melon(int cal, int cost, int kg, String info) {  super(cal, cost, kg);  this.info = info;  }  public String getInfo() {  return info;  }  public void setInfo(String info) {  this.info = info;  }  @Override  public String toString() {  return "cal = " + super.getCal() + " cost=" + super.getCost() + " kg=" + super.getKg() + " 경작농원=" + info;  }    }  package week4.practice6;  public class Main {  public static void main(String[] args) {    Melon m1 = new Melon(1200, 2500, 21, "Hallym\_farm");  Melon m2 = new Melon(1000, 2000, 10, "Software\_farm");    System.*out*.println("m1 객체정보 [ " + m1 + "]");  System.*out*.println("m2 객체정보 [ " + m2 + "]\n");  System.*out*.println(">> m2 객체의 cost와 경작농원 정보 변경 후");  m2.setCost(3000);  m2.setInfo("java\_farm");  System.*out*.println(m2 + "\n");    System.*out*.println("m1: 중량 >> " + m1.getKg());  System.*out*.println("프로그램 종료");  }  } |
| [실행 결과] |

1. 생성형 AI가 제시하는 상속을 활용하는 문제를 프로그램하고 결과를 제시 하세요. 문제 해결을 위한 코드는 답변에서 제외 되어야 합니다

|  |  |
| --- | --- |
| 생성형 AI 플랫폼 | chatGPT |
| 질문 | 생성형 AI가 제시하는 상속을 활용하는 문제를 프로그램하고 결과를 제시 하세요. 문제 해결을 위한 코드는 답변에서 제외 되어야 합니다. (다형성 제외) |
| 답변 | 문제: 상속을 활용한 동물 클래스 설계 문제 (다형성 제외)  다음 요구 사항을 만족하는 프로그램을 작성하십시오:  Animal 클래스:  name: 동물의 이름을 저장하는 속성.  age: 동물의 나이를 저장하는 속성.  sound: 동물이 내는 소리를 저장하는 속성.  Dog 클래스:  Animal 클래스를 상속받는다.  bark(): 개가 짖는 소리("Woof!")를 출력하는 메서드.  Dog 클래스에서 sound 속성은 "Woof!"로 설정.  Cat 클래스:  Animal 클래스를 상속받는다.  meow(): 고양이가 우는 소리("Meow!")를 출력하는 메서드.  Cat 클래스에서 sound 속성은 "Meow!"로 설정.  Bird 클래스:  Animal 클래스를 상속받는다.  chirp(): 새가 지저귀는 소리("Chirp!")를 출력하는 메서드.  Bird 클래스에서 sound 속성은 "Chirp!"로 설정.  요구 사항:  각 동물 클래스에서 sound 속성은 각 동물의 고유한 소리를 가지고 있도록 설정합니다.  다형성을 사용하지 않고, 각 동물 클래스에서 직접 메서드를 호출하여 각 동물의 소리를 출력합니다.  각 클래스에서 동물의 이름과 나이, 그리고 소리를 출력하는 메서드를 구현하십시오.  예시 출력:  Dog: 이름은 Max, 나이는 5살, 소리는 Woof!  Cat: 이름은 Whiskers, 나이는 3살, 소리는 Meow!  Bird: 이름은 Tweety, 나이는 2살, 소리는 Chirp! |

|  |
| --- |
| [프로그램 소스]  package week4.practice7;  class Animal {    // 객체 선언  private String name, sound;  private int age;    // 생성자  public Animal(String name, int age) {  super();  this.name = name;  this.age = age;  this.sound = "";  }  // getter, settter  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getSound() {  return sound;  }  public void setSound(String sound) {  this.sound = sound;  }  public int getAge() {  return age;  }  public void setAge(int age) {  this.age = age;  }  // Sound method  public void bark() {}  public void meow() {}  public void chirp() {}    // toString()  @Override  public String toString() {  return "이름은 " + name + ", 나이는 " + age + "살, 소리는 ";  }    }  package week4.practice7;  class Dog extends Animal {    public Dog(String name, int age) {  super(name, age);  }  public void bark() {  System.*out*.println("Woof!");  }    }  package week4.practice7;  class Cat extends Animal {    public Cat(String name, int age) {  super(name, age);  }  public void meow() {  System.*out*.println("Meow!");  }  }  package week4.practice7;  class Bird extends Animal {  public Bird(String name, int age) {  super(name, age);  }    public void chirp() {  System.*out*.println("Chiurp!");  }    }  package week4.practice7;  public class Main {  public static void main(String[] args) {  Animal dog = new Dog("Max", 5);  Animal cat = new Cat("Whiskers", 3);  Animal bird = new Bird("Tweety", 2);      System.*out*.print("Dog: " + dog);  dog.bark();  System.*out*.print("Cat: " +cat);  cat.meow();  System.*out*.print("Bird: " +bird);  bird.chirp();  }  } |
| [실행 결과] |