|  |  |  |
| --- | --- | --- |
| java2\_week06\_인터페이스(1) | 학번 : 20237107 | 이름 : 하태영 |

* 응용 프로그래밍

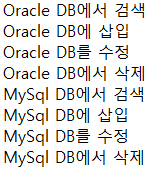
1. **[생성형 AI 활용]** 자바에서 인터페이스는 어떤 경우에 사용하나요?

|  |  |
| --- | --- |
| 생성형 AI 플랫폼 | chatGPT |
| 질문 | 자바에서 인터페이스는 어떤 경우에 사용하나요? |
| 답변 | 좋은 질문이에요! 자바에서 \*\*인터페이스(Interface)\*\*는 여러 상황에서 유용하게 사용되는데, 대표적인 경우는 다음과 같아요: ✅ 1. **다형성(Polymorphism)을 구현할 때** 인터페이스는 **다형성을 실현하는 핵심 도구**예요.   * 예를 들어, Animal이라는 인터페이스를 만들고 Dog, Cat 클래스가 이를 구현하면, Animal 타입의 변수로 Dog도 Cat도 모두 참조할 수 있어요.   interface Animal {  void sound();  }  class Dog implements Animal {  public void sound() {  System.out.println("멍멍!");  }  }  class Cat implements Animal {  public void sound() {  System.out.println("야옹~");  }  }  public class Main {  public static void main(String[] args) {  Animal a1 = new Dog();  Animal a2 = new Cat();    a1.sound(); // 멍멍!  a2.sound(); // 야옹~  }  } ✅ 2. **서로 관련 없는 클래스들에게 공통 기능을 부여할 때** 예를 들어, Printable이라는 인터페이스를 정의해서, 프린트 가능한 객체들(문서, 이미지 등)에 \*\*공통 동작(print 메서드)\*\*을 강제할 수 있어요.  interface Printable {  void print();  }  class Document implements Printable {  public void print() {  System.out.println("문서 인쇄 중...");  }  }  class Image implements Printable {  public void print() {  System.out.println("이미지 인쇄 중...");  }  } ✅ 3. **콜백 구조나 이벤트 리스너 구현 시** 자바에서는 버튼 클릭 등 이벤트 처리 시 인터페이스 기반의 콜백을 자주 사용해요.  interface OnClickListener {  void onClick();  }  class Button {  OnClickListener listener;  void setOnClickListener(OnClickListener listener) {  this.listener = listener;  }  void click() {  if (listener != null) listener.onClick();  }  } ✅ 4. **다중 상속 효과를 얻기 위해** 자바는 클래스의 **다중 상속을 지원하지 않지만**, 인터페이스는 여러 개 구현 가능해요.  interface Flyable {  void fly();  }  interface Swimmable {  void swim();  }  class Duck implements Flyable, Swimmable {  public void fly() {  System.out.println("오리 날아요!");  }  public void swim() {  System.out.println("오리 헤엄쳐요!");  }  } ✅ 5. **프레임워크나 라이브러리에서 기능 확장을 위한 Hook으로** 예: Spring의 Controller, Repository, Service 등도 인터페이스 기반으로 되어 있어서 **유연한 구조와 테스트의 용이성**을 제공해요.  필요하다면 구체적인 예제나 상황별 추천 구조도 설명해줄게요. 혹시 지금 과제나 프로젝트에서 인터페이스 쓰려는 부분 있어? |

1. **[생성형 AI 활용]** 인터페이스 멤버로 가능한 모든 형태와 구체화하는 방법을 예와 함께 설명하세요. 인터페이스를 구현할 때 주의할 점은 무엇인가요?

|  |  |
| --- | --- |
| 생성형 AI 플랫폼 | chatGPT |
| 질문 | 1.  인터페이스 멤버로 가능한 모든 형태와 구체화하는 방법을 예와 함께 설명하세요. 인터페이스를 구현할 때 주의할 점은 무엇인가요?  2.  예제 코드 main도 알려줘 |
| 답변 | 1.  좋은 질문이에요! 인터페이스는 자바에서 **계약(Contract)** 역할을 하며, 클래스가 \*\*"이 기능을 반드시 구현해야 해!"\*\*라는 약속을 하게 해주는 도구예요. 자, 하나씩 차근차근 정리해볼게요 👇 🔷 인터페이스 멤버로 가능한 모든 형태 자바 인터페이스에서 정의할 수 있는 멤버는 다음과 같아요:   | **멤버 종류** | **설명** | **특징** | | --- | --- | --- | | **상수 (Constant)** | public static final 상수만 가능 | 생략해도 자동으로 붙어요 | | **추상 메서드 (Abstract Method)** | public abstract 메서드만 가능 | 생략해도 자동으로 붙어요 | | **디폴트 메서드 (Default Method)** | 기본 구현이 있는 메서드 | default 키워드 필요 | | **정적 메서드 (Static Method)** | 인터페이스 자체에서 호출하는 정적 메서드 | static 키워드 필요 | | **private 메서드** | 공통 로직 추출용 (Java 9+) | 인터페이스 내부에서만 사용 가능 |  🔷 예제 코드 interface MyInterface {  // 1. 상수  int MAX = 100; // public static final이 생략됨  // 2. 추상 메서드  void doSomething(); // public abstract가 생략됨  // 3. 디폴트 메서드 (Java 8+)  default void defaultMethod() {  System.out.println("디폴트 메서드입니다.");  }  // 4. 정적 메서드 (Java 8+)  static void staticMethod() {  System.out.println("정적 메서드입니다.");  }  // 5. private 메서드 (Java 9+)  private void helper() {  System.out.println("인터페이스 내부용 private 메서드입니다.");  }  } 🔷 인터페이스 구현(구체화) 방법 class MyClass implements MyInterface {  // 추상 메서드 반드시 구현해야 함  public void doSomething() {  System.out.println("doSomething()을 구현했습니다.");  }  // defaultMethod(), staticMethod()는 선택 구현 or 직접 호출  } ✅ 인터페이스 구현 시 주의할 점  | **항목** | **설명** | | --- | --- | | 🔸 **추상 메서드는 반드시 구현** | 구현하지 않으면 해당 클래스도 abstract이어야 함 | | 🔸 **다중 인터페이스 구현 시 이름 충돌 주의** | 같은 시그니처의 디폴트 메서드가 여러 인터페이스에 있을 경우, 직접 재정의해야 함 | | 🔸 **인터페이스의 상수는 변경 불가** | public static final이라서 무조건 상수! 값 변경 불가 | | 🔸 **인터페이스는 상태(state)를 가질 수 없음** | 필드는 모두 상수. 인스턴스 변수나 생성자는 가질 수 없음 |  ⚠️ 이름 충돌 예시 (다중 인터페이스) interface A {  default void hello() {  System.out.println("Hello from A");  }  }  interface B {  default void hello() {  System.out.println("Hello from B");  }  }  class C implements A, B {  // 오류 발생! hello()가 충돌함  // 해결 방법: 직접 오버라이딩  public void hello() {  A.super.hello(); // 또는 B.super.hello()  }  } 📌 정리하면...  * 인터페이스는 상수, 추상 메서드, default/static/private 메서드를 가질 수 있음 * 구현 클래스는 **추상 메서드를 모두 오버라이드**해야 함 * default 메서드는 선택 사항이지만, 충돌 시 직접 오버라이딩해야 함 * 상태(state)나 생성자는 가질 수 없음   혹시 지금 인터페이스 구현 중 충돌 나는 코드나 더 보고 싶은 예제가 있다면 보여줘도 좋아요. 같이 분석해볼게요! 😊  2.  물론이죠! 아까 설명드렸던 인터페이스 멤버들을 모두 포함한 예제를 main() 메서드까지 포함해서 정리해볼게요 😊 ✅ 인터페이스 예제 (전체 코드) interface MyInterface {  // 1. 상수 (자동으로 public static final)  int MAX = 100;  // 2. 추상 메서드 (자동으로 public abstract)  void doSomething();  // 3. 디폴트 메서드  default void defaultMethod() {  System.out.println("디폴트 메서드입니다.");  }  // 4. 정적 메서드  static void staticMethod() {  System.out.println("정적 메서드입니다.");  }  // 5. private 메서드 (Java 9+)  private void helper() {  System.out.println("private 헬퍼 메서드 (내부에서만 호출 가능)");  }  }  // 인터페이스를 구현한 클래스  class MyClass implements MyInterface {  // 추상 메서드 구현  public void doSomething() {  System.out.println("doSomething()을 구현했습니다.");  System.out.println("MAX 값은: " + MAX); // 상수 사용  }  }  // main() 함수 포함  public class Main {  public static void main(String[] args) {  // 객체 생성  MyInterface obj = new MyClass();  // 추상 메서드 구현체 호출  obj.doSomething();  // 디폴트 메서드 호출  obj.defaultMethod();  // 정적 메서드는 클래스 이름으로 호출  MyInterface.staticMethod();  // 상수 접근  System.out.println("인터페이스 상수: " + MyInterface.MAX);  }  } 🧾 실행 결과 (예상 출력) doSomething()을 구현했습니다.  MAX 값은: 100  디폴트 메서드입니다.  정적 메서드입니다.  인터페이스 상수: 100  이 코드로 인터페이스의 **모든 멤버 종류**와 **구현 방식**, 그리고 **사용 방법**을 한눈에 확인할 수 있어요. 궁금한 점이나 에러 난 코드 있으면 언제든지 보여줘요. 같이 수정해볼게요! 😊 |
| 실행결과 | 텍스트, 폰트, 스크린샷, 디자인이(가) 표시된 사진  AI가 생성한 콘텐츠는 부정확할 수 있습니다. |

1. 제시된 실행 결과를 보고 DataAccessObject 인터페이스와 OracleDB와 MySqlDB 구현 클래스를 작성하시오.



|  |
| --- |
| public class InterfaceTest {  public static void dbWork(DataAccessObject dao) {  dao.select();  dao.insert();  dao.update();  dao.delete();  }  public static void main(String[] args) {  *dbWork*(new OracleDB("Oracle DB"));  *dbWork*(new MySqlDB("MySql DB"));  }  } |

|  |
| --- |
| [프로그램 소스]  package practice3;  interface DataAccessObject {  void select();  void insert();  void update();  void delete(); }  package practice3;  class OracleDB implements DataAccessObject {  private String dbName;   public OracleDB(String dbName) {  this.dbName = "Oracle DB";  }   @Override  public void select() {  System.*out*.println(dbName + "에서 검색");  }  @Override  public void insert() {  System.*out*.println(dbName + "에 삽입");  }  @Override  public void update() {  System.*out*.println(dbName + "를 수정");  }  @Override  public void delete() {  System.*out*.println(dbName + "에서 삭제");  } }  package practice3;  class MySqlDB implements DataAccessObject {  private String dbName;   public MySqlDB(String dbName) {  this.dbName = "MySql DB";  }  @Override  public void select() {  System.*out*.println(dbName + "에서 검색");  }  @Override  public void insert() {  System.*out*.println(dbName + "에 삽입");  }  @Override  public void update() {  System.*out*.println(dbName + "를 수정");  }  @Override  public void delete() {  System.*out*.println(dbName + "에서 삭제");  } }  package practice3;  public class InterfaceTest {  public static void dbWork(DataAccessObject dao) {  dao.select();  dao.insert();  dao.update();  dao.delete();  }   public static void main(String[] args) {  *dbWork*(new OracleDB("Oracle DB"));  *dbWork*(new MySqlDB("MySql DB"));  } } |
| [실행 결과]  텍스트, 폰트, 스크린샷, 블랙이(가) 표시된 사진  AI가 생성한 콘텐츠는 부정확할 수 있습니다. |

1. 조건대로 프로그램을 작성하고 테스트 하시오

>>SmartDevice 인터페이스를 정의하여 다음과 같은 메소드를 선언한다

void turnOn() -> 장치 켜기, void turnOff()->장치 끄기, Boolean isToggle() -> 상태 변경

>>SmartLight, SmartThermostat, SmartTV 클래스들을 구현하여 각 장치의 turnOn()과 turnOff(), isToggle 메서드를 정의한다. 각 클래스는 상태를 나타내는 필드를 갖는다

>>SmartHomeController 클래스를 작성하여 여러 스마트 장치들을 한 번에 제어할 수 있게 한다. 프로그램을 작성하고 테스트 하시오.

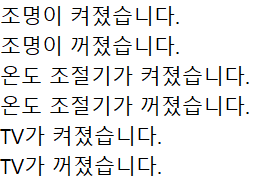
**public** **static** **void** main(String[] args) {

SmartHomeController controller = **new** SmartHomeController();

SmartDevice light = **new** SmartLight();

SmartDevice thermostat = **new** SmartThermostat();

SmartDevice tv = **new** SmartTV();

 //장치 제어

controller.controlDevice(light);

controller.controlDevice(light);

controller.controlDevice(thermostat);

controller.controlDevice(thermostat);

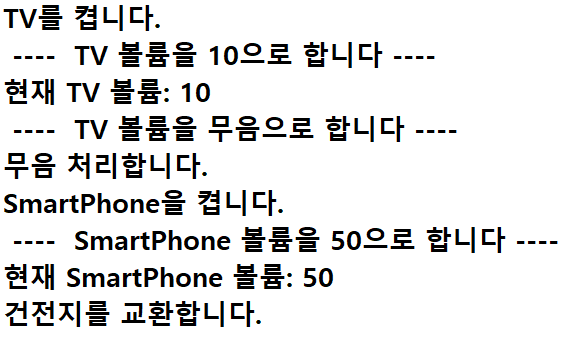
controller.controlDevice(tv);

controller.controlDevice(tv);

}

|  |
| --- |
| [프로그램 소스]  package practice4;  interface SmartDevice {  void turnOn(); // 장치 켜기  void turnOff(); // 장치 끄기  Boolean isToggle(); // 상태 변경 }  package practice4;  class SmartLight implements SmartDevice {  String controlDevice;  boolean isOn;   public SmartLight() {}   @Override  public void turnOn() {  System.*out*.println("조명이 켜졌습니다.");  isOn = true;  }   @Override  public void turnOff() {  System.*out*.println("조명이 꺼졌습니다.");  isOn = false;  }   @Override  public Boolean isToggle() {  return isOn;  } }  package practice4;  class SmartTV implements SmartDevice {   boolean isOn;   public SmartTV(){}   @Override  public void turnOn() {  System.*out*.println("TV가 켜졌습니다.");  isOn = true;  }   @Override  public void turnOff() {  System.*out*.println("TV가 꺼졌습니다.");  isOn = false;  }   @Override  public Boolean isToggle() {  return isOn;  } }  package practice4;  class SmartThermostat implements SmartDevice {   boolean isOn;   public SmartThermostat(){}   @Override  public void turnOn() {  System.*out*.println("온도 조절기가 켜졌습니다.");  isOn = true;  }   @Override  public void turnOff() {  System.*out*.println("온도 조절기가 꺼졌습니다.");  isOn = false;  }   @Override  public Boolean isToggle() {  return isOn;  } }  package practice4;  public class SmartHomeController {  public void controlDevice(SmartDevice device) {  if(device.isToggle() == true) {  device.turnOff();  } else {  device.turnOn();  }  } }  package practice4;  public class Main {  public static void main(String[] args) {  SmartHomeController controller = new SmartHomeController();   SmartDevice light = new SmartLight();  SmartDevice thermostat = new SmartThermostat();  SmartDevice tv = new SmartTV();   //장치 제어  controller.controlDevice(light);  controller.controlDevice(light);  controller.controlDevice(thermostat);  controller.controlDevice(thermostat);  controller.controlDevice(tv);  controller.controlDevice(tv);  }  } |
| [실행 결과]  텍스트, 폰트, 스크린샷, 디자인이(가) 표시된 사진  AI가 생성한 콘텐츠는 부정확할 수 있습니다. |

1. 다음과 같은 인터페이스를 구현하는 클래스 TV와 SmartPhone를 작성하고 main()에서 테스트 하시오. 제시된 결과를 참조하시오



public interface RemoteControl {

//상수 필드 선언

public final static int *MAX\_VOLUME* = 100;

public int *MIN\_VOLUME*=0;

//추상 메소드 선언

public void turnOn();

public void turnOff();

public void setVolume(int volume);

default void setMute(boolean mute) { //디폴트 메소드

if(mute) {

System.*out*.println("무음 처리합니다.");

} else {

System.*out*.println("무음 해제합니다.");

}

}

static void changeBattery() {

System.*out*.println("건전지를 교환합니다.");

}

}

|  |
| --- |
| [프로그램 소스]  package practice5;  public interface RemoteControl {  //상수 필드 선언  public final static int *MAX\_VOLUME* = 100;  public int *MIN\_VOLUME*=0;   //추상 메소드 선언  public void turnOn();  public void turnOff();  public void setVolume(int volume);   default void setMute(boolean mute) { //디폴트 메소드  if(mute) {  System.*out*.println("무음 처리합니다.");  } else {  System.*out*.println("무음 해제합니다.");  }  }  static void changeBattery() {  System.*out*.println("건전지를 교환합니다.");  }  }  package practice5;  class TV implements RemoteControl {  private String name;  private int volume;  private boolean mute;   public TV() {  this.name = "TV";  this.volume = 0;  this.mute = false;  }   @Override  public void turnOn(){  System.*out*.println(name + "를 켭니다.");  }  @Override  public void turnOff(){  System.*out*.println(name + "를 끕니다.");  }  @Override  public void setVolume(int volume) {  this.volume = volume;  System.*out*.println("---- " + name + " 볼륨을 " + volume + "으로 합니다 ----");  }  @Override  public void setMute(boolean mute) { //디폴트 메소드  if(mute) {  System.*out*.println("---- " + name + " 볼륨을 무음으로 합니다 ----");  System.*out*.println("무음 처리합니다.");  } else {  System.*out*.println("무음 해제합니다.");  }  }  @Override  public String toString() {  return "현재 " + name + " 볼륨: " + volume;  } }  package practice5;  class SmartPhone implements RemoteControl {  private String name;  private int volume;  private boolean mute;   public SmartPhone() {  this.name = "SmartPhone";  this.volume = 0;  this.mute = false;  }   @Override  public void turnOn(){  System.*out*.println(name + "을 켭니다.");  }  @Override  public void turnOff(){  System.*out*.println(name + "을 끕니다.");  }  @Override  public void setVolume(int volume) {  this.volume = volume;  System.*out*.println("---- " + name + " 볼륨을 " + volume + "으로 합니다 ----");  }  @Override  public void setMute(boolean mute) { //디폴트 메소드  if(mute) {  System.*out*.println("---- " + name + " 볼륨을 무음으로 합니다 ----");  System.*out*.println("무음 처리합니다.");  } else {  System.*out*.println("무음 해제합니다.");  }  }   @Override  public String toString() {  return "현재 " + name + " 볼륨: " + volume;  }  }  package practice5;   public class Main {  public static void main(String[] args) {  RemoteControl rc;   rc = new TV();  rc.turnOn();  rc.setVolume(10);  System.*out*.println(rc.toString());  rc.setMute(true);   rc = new SmartPhone();  rc.turnOn();  rc.setVolume(50);  System.*out*.println(rc.toString());   RemoteControl.*changeBattery*(); // static 메서드 호출  } } |
| [실행 결과] |

1. 다음과 같은 조건을 만족하는 프로그램을 작성하시오

(a) 인터페이스 IGraphics를 작성한다.

- double perimeter()과 void draw()가 선언, 매개변수는 없음

(b) IGraphics을 구현하는 Rectangle 클래스를 작성한다.

- Rectangle 클래스는 2개의 private 실수 필드인 length와 width을 가진다.

- 두개의 필드를 매개값으로 초기화 하는 생성자

- draw()에서는 “도형 Rectangle을 그립니다.”를 출력한다

- perimeter()에서는 사각형 둘레를 리턴 한다.

(c) IGraphics을 구현하는 Circle클래스를 작성한다

- Rectangle 클래스는 1개의 private 실수 필드인 radius을 가진다.

- 한 개의 필드를 매개값으로 초기화 하는 생성자

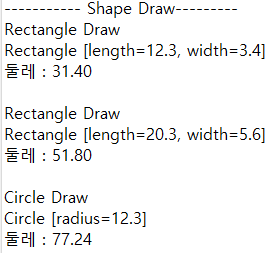
- draw()에서는 “도형 Circle을 그립니다.”를 출력한다.

- perimeter()에서는 원 둘레를 리턴 한다.

ShapeTest 클래스를 작성한다.

- 메인 메소드에서 IGraphics 타입의 배열 arrayOfShapes를 3개생성하고, Rectangle, Circle 객체로 초기화

- 작성된 메소드를 호출하여 제시된 결과처럼 출력한다.



|  |
| --- |
| [프로그램 소스]  package practice6;  interface IGraphics {  public void draw();  public double perimeter(); }  package practice6;  class Rectangle implements IGraphics {  private String shape;  private double length;  private double width;   public Rectangle(double length, double width) {  this.shape = "Rectangle";  this.length = length;  this.width = width;  }   @Override  public void draw() {  System.*out*.println(shape + " Draw");  }   @Override  public double perimeter() {  return 2 \* (width + length);  }   @Override  public String toString() {  return shape + " [length=" + length + ", width=" + width + "]";  } }  package practice6;  class Circle implements IGraphics {  private String shape;  private double radius;   public Circle(double radius) {  this.shape = "Circle";  this.radius = radius;  }   @Override  public void draw() {  System.*out*.println(shape + " Draw");  }   @Override  public double perimeter() {  return 2 \* Math.*PI* \* radius;  }   @Override  public String toString() {  return shape + " [radius=" + radius + "]";  } }  package practice6;  public class ShapeTest {  public static void main(String[] args) {  IGraphics[] arrayOfIGraphics = new IGraphics[3];   arrayOfIGraphics[0] = new Rectangle(12.3, 3.4);  arrayOfIGraphics[1] = new Rectangle(20.3, 5.6);  arrayOfIGraphics[2] = new Circle(12.3);   for(IGraphics iGraphics : arrayOfIGraphics) {  iGraphics.draw();  System.*out*.println(iGraphics.toString());  System.*out*.printf("둘레 : %.2f\n", iGraphics.perimeter());  System.*out*.println();  }  } } |
| [실행 결과]  텍스트, 스크린샷, 폰트이(가) 표시된 사진  AI가 생성한 콘텐츠는 부정확할 수 있습니다. |

1. 아래의 설명에 따라 인터페이스와 클래스들을 정의하고 프로그램을 테스트하시오.

- IComparable 인터페이스는 int compareTo(Object other) 형태의 추상 메소드를 가지며 현재 객체가 other 객체보다 키가 크면 1, 같으면 0, 작으면 –1을 반환한다.

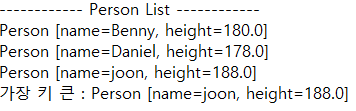
- Person 클래스는 이름(name), 키(height) 필드와 객체의 정보를 출력하는 toString()을 가진다. Person 클래스의 생성자에서는 전달된 값을 이름과 키 필드에 저장한다. Person 클래스는 IComparable 인터페이스를 구현한다.

- class PersonTest는 main()과 getMaximum()으로 구성 된다.

- main()에서는 Person 타입의 배열을 선언하여 세 사람의 이름과 키를 저장하고 getMaximum()을 호출하여 가장 키가 큰 사람의 정보를 출력한다.

- getMaximum()에서는 이 IComparable 인터페이스를 이용하여서 가장 키가 큰 사람의 객체를 반환한다.

- 프로그램을 수행하면 아래의 그림과 같은 결과를 보인다. 단, 제시된 main()메소드는 그대로 사용합니다



public static void main(String[] args) {

Person[] per = { new Person("Benny", 180), new Person("Daniel", 178), new Person("joon", 188)};

System.*out*.println("------------ Person List ------------");

for (Person p : per)

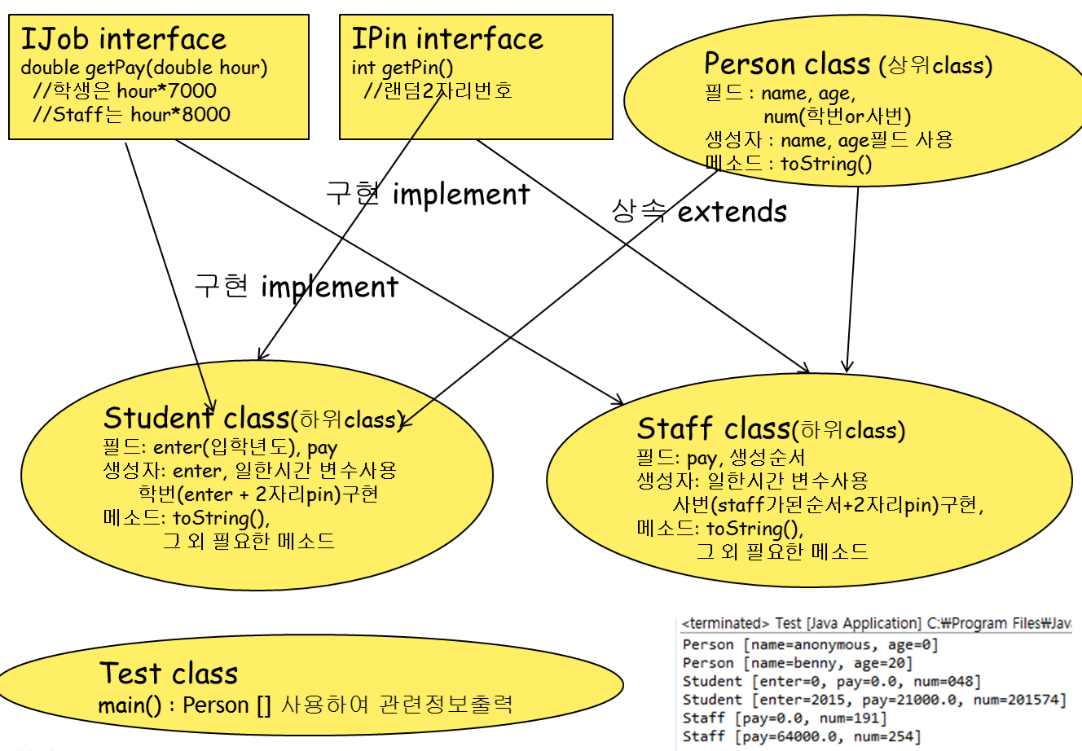
System.*out*.println(p);

System.*out*.println("가장 키 큰 : " + *getMaximum*(per));

}

|  |
| --- |
| [프로그램 소스]  package practice7;  interface IComparable {  public int compareTo(Object other); }  package practice7;  class Person implements IComparable {  private String name;  private double height;   public Person(String name, double height) {  this.name = name;  this.height = height;  }   @Override  public int compareTo(Object other) {  if(other instanceof Person) {  Person p = (Person) other;  if (this.height > p.height) {  return 1;  } else if (this.height == p.height) {  return 0;  } else {  return -1;  }  }  // 잘못된 타입의 객체가 들어오더라도 프로그램이 에러 없이 동작하도록 처리  return -1;  }   public String toString() {  return "Person [name=" + name + ", height=" + height + "]";  } }  package practice7;  public class Main {  public static void main(String[] args) {  Person[] per = {  new Person("Benny", 180),  new Person("Daniel", 178),  new Person("joon", 188)  };   System.*out*.println("------------ Person List ------------");   for (Person p : per)  System.*out*.println(p);   System.*out*.println("가장 키 큰 : " + *getMaximum*(per));  }   private static String getMaximum(Person[] per) {  Person max = per[0]; // max를 0번째로 임시지정  // int i=1인 이유 : 0번째를 max로 저장했기 때문에 비교할 필요가 없다.  for (int i = 1; i < per.length; i++) {  // compareTo()는 키 비교 기준이므로, 키가 더 크면 양수(1) 반환  if (per[i].compareTo(max) > 0) {  // max 갱신  max = per[i];  }  }  return max.toString();  }  } |
| [실행 결과]  텍스트, 폰트, 스크린샷이(가) 표시된 사진  AI가 생성한 콘텐츠는 부정확할 수 있습니다. |

1. 다음 그림과 같은 프로그램을 완성하시오. 단, 제시된 main()메소드는 그대로 사용합니다.



public static void main(String[] args) {

Person [] per = new Person[6];

per[0] = new Person();

per[1] = new Person("benny", 20);

per[2] = new Student();

per[3] = new Student(2015, 3);

per[4] = new Staff();

per[5] = new Staff(8);

for(Person obj : per) {

System.*out*.println(obj);

}

}

|  |
| --- |
| [프로그램 소스]  package practice8;  interface IJob {  double getPay(double hour); }  package practice8;  interface IPin {  int getPin(); }  package practice8;  class Person {  protected String name;  protected int age;  protected String num;   public Person(){  this("anonymous", 0);  }   public Person(String name, int age) {  this.name = name;  this.age = age;  }   @Override  public String toString() {  return "Person [name=" + name + ", age=" + age + "]";  } }  package practice8;  class Student extends Person implements IJob, IPin {  private int enter;  private double pay;   public Student() {  this(0, 0);  }   public Student(int enter, double hour){  super();  this.enter = enter;  this.pay = getPay(hour);  int pin = getPin();  this.num = enter + String.*format*("%02d", pin);  }   @Override  public double getPay(double hour){  return hour \* 7000;  }   @Override  public int getPin() {  return (int)(Math.*random*() \* 100);  }   @Override  public String toString(){  return "Student [enter=" + enter + ", pay=" + pay + ", num="+ num + "]";  } }  package practice8;  class Staff extends Person implements IJob, IPin {  private double pay;  protected static int *count* = 1; // 생성 순서   public Staff(){  this(0);  }   public Staff(double hour){  super();  this.pay = (int)getPay(hour);  this.num = String.*valueOf*(*count*++) + String.*format*("%02d", getPin());  }   @Override  public double getPay(double hour){  return hour \* 8000;  }   @Override  public int getPin() {  return (int)(Math.*random*() \* 100);  }   @Override  public String toString(){  return "Staff [pay=" + pay + ", num=" + num + "]";  }   }  package practice8;  public class Test {  public static void main(String[] args) {  Person [] per = new Person[6];   per[0] = new Person();  per[1] = new Person("benny", 20);  per[2] = new Student();  per[3] = new Student(2015, 3);  per[4] = new Staff();  per[5] = new Staff(8);   for(Person obj : per) {  System.*out*.println(obj);  }  } } |
| [실행 결과]  텍스트, 폰트, 스크린샷이(가) 표시된 사진  AI가 생성한 콘텐츠는 부정확할 수 있습니다. |