|  |  |  |
| --- | --- | --- |
| java2\_week02\_클래스(1) | 학번 : 20237107 | 이름 : 하태영 |

* 응용 프로그래밍

1. **[생성형 AI 활용]** 다음 질문에 대하여 답하세요
2. 디폴트 생성자는 반드시 있어야 하는가? 반드시 있어야 하는 경우는?
3. 클래스는 반드시 필드와 메소드 모두 있어야 하는가?
4. 객체는 클래스없이 생성할 수 있는가?

|  |  |
| --- | --- |
| 생성형 AI 플랫폼 | chatGPT |
| 질문 | 1. 디폴트 생성자는 필수인가? 필수인 경우는? 2. 클래스는 필수적으로 필드와 메소드 모두 있어야 하는가? 3. 객체는 클래스 없이 생성할 수 있는가? |
| 답변 | 1.  디폴트 생성자는 **필수는 아닙니다**, 그러나 특정 상황에서는 **필수적**일 수 있습니다. 디폴트 생성자에 대한 필요성을 이해하려면 몇 가지 중요한 경우를 살펴볼 필요가 있습니다.  **1. 디폴트 생성자가 무엇인가?**  디폴트 생성자는 **매개변수가 없는 생성자**로, 객체를 생성할 때 특별한 인자가 없이 자동으로 호출되는 생성자입니다. 예를 들어, 다음과 같습니다:  class MyClass {  public:  MyClass() {  // 디폴트 생성자  }  };  **2. 디폴트 생성자가 반드시 필요한 경우**  디폴트 생성자가 필요할 수 있는 주요 상황은 다음과 같습니다:  **1) 객체 배열 생성**  배열로 객체를 생성할 때 각 객체를 초기화하기 위해 디폴트 생성자가 필요합니다. 예를 들어, MyClass 객체를 10개 생성하는 배열을 선언할 때:  MyClass arr[10]; // 디폴트 생성자가 필요  이 경우, arr 배열에 있는 각 객체가 디폴트 생성자를 통해 자동으로 초기화됩니다. 만약 MyClass가 디폴트 생성자를 제공하지 않으면, 이 배열을 선언할 수 없습니다.  **2) 객체를 동적으로 생성할 때**  new 연산자를 통해 객체를 동적으로 생성할 때도 디폴트 생성자가 필요합니다. 예를 들어:  MyClass\* obj = new MyClass(); // 디폴트 생성자가 필요  위와 같이 객체를 동적으로 생성할 때 디폴트 생성자가 필요합니다. 만약 디폴트 생성자가 없다면, 매개변수가 있는 생성자를 사용해야 하며, 그에 맞는 인자를 제공해야 합니다.  **3) 컨테이너에서 객체를 생성할 때**  C++ STL의 컨테이너, 예를 들어 std::vector와 같은 자료구조는 내부적으로 객체를 생성할 때 디폴트 생성자를 사용합니다. 예를 들어:  std::vector<MyClass> v(10); // MyClass의 디폴트 생성자가 필요  위 코드에서는 std::vector가 10개의 MyClass 객체를 디폴트 생성자를 사용하여 생성합니다. 이 경우에도 디폴트 생성자가 없으면 std::vector를 사용할 수 없습니다.  **4) 클래스 멤버가 다른 객체인 경우**  클래스의 멤버가 다른 객체일 때, 해당 객체의 디폴트 생성자가 필요할 수 있습니다. 예를 들어, MyClass 클래스의 멤버로 다른 클래스를 포함할 경우, 해당 클래스의 디폴트 생성자가 필요할 수 있습니다:  class OtherClass {  public:  OtherClass() {  // 디폴트 생성자  }  };  class MyClass {  private:  OtherClass obj; // 다른 클래스의 객체가 멤버로 있을 때  };  MyClass의 멤버 변수로 OtherClass 객체를 사용할 때, OtherClass의 디폴트 생성자가 필요합니다.  **3. 디폴트 생성자가 필수적이지 않은 경우**  디폴트 생성자가 항상 필요한 것은 아닙니다. 예를 들어:  **1) 매개변수가 있는 생성자만 정의할 때**  만약 클래스가 매개변수가 있는 생성자만 정의하고, 디폴트 생성자를 정의하지 않으면, 객체를 생성할 때 반드시 해당 생성자의 인자들을 제공해야 합니다. 예를 들어:  class MyClass {  public:  MyClass(int x) {  // x를 받는 생성자  }  };  이 경우, MyClass 객체를 생성할 때는 반드시 int 타입의 인자를 제공해야 하므로 디폴트 생성자는 필요하지 않습니다.  MyClass obj(10); // 생성할 때 반드시 인자 제공  **2) 디폴트 생성자를 명시적으로 삭제한 경우**  디폴트 생성자를 사용하지 않도록 막는 경우, delete 키워드를 사용할 수 있습니다. 이 경우, 디폴트 생성자는 호출할 수 없습니다.  class MyClass {  public:  MyClass() = delete; // 디폴트 생성자를 삭제  MyClass(int x) {  // 매개변수가 있는 생성자  }  };  위와 같이 디폴트 생성자를 삭제하면, MyClass obj;와 같은 방식으로 객체를 생성할 수 없고, 반드시 MyClass obj(10);과 같이 인자를 제공해야 합니다.  **4. 결론**  디폴트 생성자는 **필수적이지 않지만**, 특정 상황에서는 필수적입니다. 예를 들어:   * **배열 생성**, **동적 할당**, **컨테이너 사용** 등에서는 디폴트 생성자가 필요합니다. * 반면, **매개변수가 있는 생성자만 있는 경우**나 **디폴트 생성자를 명시적으로 삭제한 경우**에는 디폴트 생성자가 필요하지 않습니다.   따라서, 디폴트 생성자가 필요한지 여부는 클래스의 용도와 사용되는 맥락에 따라 달라집니다.  **------------------------------------------------------------------------------------------------**  2.  **클래스**는 필수적으로 \*\*필드(멤버 변수)\*\*와 \*\*메소드(멤버 함수)\*\*를 모두 가져야 하는 것은 아닙니다. 클래스는 객체 지향 프로그래밍에서 \*\*"타입"\*\*을 정의하는 설계도로, \*\*데이터(필드)\*\*와 \*\*동작(메소드)\*\*을 정의할 수 있지만, 둘 중 하나만 있을 수도 있습니다. 상황에 따라 필드만 있거나, 메소드만 있는 클래스도 가능합니다.  **1. 필드만 있는 클래스**  클래스는 **데이터 구조**를 표현하는 데 사용될 수 있습니다. 이 경우 클래스는 필드만 가지고 있으며, 메소드는 필요하지 않을 수 있습니다. 예를 들어, 다음은 좌표를 표현하는 Point 클래스입니다:  class Point {  public:  int x; // 필드  int y; // 필드  };  위 클래스는 단순히 x와 y라는 두 개의 데이터를 저장하는 데 사용됩니다. Point 객체는 필드만을 갖고 있으며, 메소드는 없습니다. 이와 같은 클래스는 **데이터 저장** 목적에만 사용됩니다.  **2. 메소드만 있는 클래스**  클래스는 **동작**이나 **기능**을 정의하는 데 사용할 수도 있습니다. 이 경우 클래스는 필드를 가지지 않고 메소드만 정의할 수 있습니다. 예를 들어:  class Utility {  public:  static void printHello() {  std::cout << "Hello, World!" << std::endl;  }  };  위의 Utility 클래스는 필드는 없고, printHello()라는 메소드만을 가지고 있습니다. 이 클래스는 특정 기능(문자열 출력)을 수행하는 데 사용됩니다. 이와 같은 클래스는 **기능 수행**만을 목적으로 할 수 있습니다.  **3. 필드와 메소드가 모두 있는 클래스**  가장 일반적으로 사용되는 클래스는 **필드와 메소드**를 모두 가지는 경우입니다. 예를 들어:  class Car {  private:  int speed; // 필드  public:  Car() : speed(0) {} // 생성자  void accelerate() { // 메소드  speed += 10;  }  int getSpeed() const { // 메소드  return speed;  }  };  위의 Car 클래스는 speed라는 필드를 가지며, 자동차의 속도를 증가시키거나 확인하는 메소드들을 정의하고 있습니다. 이 클래스는 **상태**(필드)와 **동작**(메소드)을 모두 포함하고 있습니다.  **4. 빈 클래스**  클래스가 필드나 메소드 없이 정의될 수도 있습니다. 이 경우 클래스는 **타입**을 정의하는 용도로 사용됩니다. 예를 들어:  class Empty {};  이 Empty 클래스는 필드도 메소드도 없지만, 여전히 하나의 타입으로 존재할 수 있습니다. 이와 같은 클래스는 **타입 정의**만을 위해 사용될 수 있습니다.  **결론**  클래스는 **필수적으로 필드와 메소드**를 모두 가져야 하는 것은 아닙니다. 클래스는 **필드만**, **메소드만**, 또는 **둘 다** 가질 수 있습니다. 클래스의 설계는 **목적**에 따라 달라지며, 데이터만 필요한 경우 필드만 있을 수 있고, 기능만 필요한 경우 메소드만 있을 수 있습니다.  3.  객체는 **클래스 없이 생성할 수 없습니다**. 객체 지향 프로그래밍에서 **객체**는 **클래스**의 **인스턴스**(instance)로 생성됩니다. 즉, 객체는 반드시 어떤 클래스에 속해야 하며, 클래스는 객체를 생성하는 **설계도** 역할을 합니다. 클래스를 통해 객체의 상태(필드)와 행동(메소드)을 정의합니다.  **1. 클래스 없이 객체를 만들 수 없는 이유**  객체는 클래스에 정의된 속성과 동작을 가지는 인스턴스입니다. 클래스는 객체의 **타입**을 정의하는 일종의 **템플릿** 역할을 합니다. 클래스가 없다면, 객체가 어떤 속성을 가질지, 어떤 동작을 할지를 정의할 수 없기 때문에 객체를 생성할 수 없습니다.  **예시: 클래스 없이 객체를 만들 수 없다는 점**  // 잘못된 예시: 클래스 없이 객체를 생성하려고 하면 컴파일 오류가 발생합니다.  MyClass obj; // MyClass라는 클래스가 없으면 오류 발생  이 코드에서 MyClass라는 클래스가 없으면, 컴파일러는 obj 객체를 어떻게 생성할지 알 수 없으므로 오류가 발생합니다.  **2. 클래스를 사용하지 않는 예외적인 경우**  일반적으로 객체는 클래스를 기반으로 생성되지만, **클래스를 사용하지 않는 특별한 경우**도 존재합니다. 예를 들어, **동적 메모리 할당**을 통해 객체처럼 동작하는 데이터를 만들 수는 있지만, 이는 **클래스를 사용하지 않는 객체 생성**이라고 할 수 없습니다. 이런 방식은 주로 C나 저수준 프로그래밍에서 사용됩니다.  **예시: struct나 union 사용**  C에서 클래스를 사용하지 않고 \*\*구조체(struct)\*\*를 사용해 데이터를 정의하고 객체처럼 다룰 수 있습니다.  #include <stdio.h>  struct Point {  int x;  int y;  };  int main() {  struct Point p1 = {10, 20}; // 구조체를 통해 객체처럼 사용  printf("x: %d, y: %d\n", p1.x, p1.y);  return 0;  }  여기서 struct Point는 **클래스**와 비슷한 역할을 하지만, **C에서는 클래스가 아니라 구조체**이기 때문에 엄밀히 말해 클래스 없이 객체처럼 데이터를 다룬다고 볼 수 있습니다. 하지만 struct는 C++의 class와 비슷한 역할을 하므로 **객체 지향 개념**에서의 "객체"는 여전히 클래스가 필요합니다.  **3. 동적 메모리 할당을 통해 객체처럼 사용하는 경우 (C++)**  C++에서는 new 연산자를 사용하여 클래스의 인스턴스를 동적으로 생성할 수 있습니다. 그러나 여전히 **클래스**가 있어야만 객체를 생성할 수 있습니다.  class MyClass {  public:  int value;  MyClass() : value(10) {}  };  int main() {  MyClass\* obj = new MyClass(); // 여전히 클래스가 필요  delete obj;  }  위 예시에서는 MyClass라는 클래스의 객체를 동적으로 생성했지만, 객체를 만들기 위해서는 MyClass 클래스가 반드시 필요합니다.  **결론**  **객체는 클래스 없이 생성할 수 없습니다**. 클래스는 객체의 설계도 역할을 하며, 객체는 해당 클래스를 기반으로 만들어집니다. 따라서 객체 지향 프로그래밍에서 객체를 생성하려면 반드시 해당 클래스가 있어야 합니다. C나 C++와 같은 언어에서 클래스 없이 객체처럼 동작하는 구조체나 데이터를 사용할 수 있지만, 그것은 객체 지향 프로그래밍에서 말하는 "객체"와는 다릅니다. |

1. **[생성형 AI 활용]** Contact 객체를 배열에 저장하고 검색 작업을 수행할 수 있는 클래스를 만들고 테스트 하는 프로그램을 작성하세요. (Contact 배열 저장, 검색 클래스) 클래스 총 3개..(1. Contact, 배열 저장, 검색)

class Contact{

private String name, phoneNumber;

private int age;

public Contact() {

this(null, null, 0);

}

public Contact(String name) {

this(name, null, 0);

}

public Contact(String name, String phoneNumber) {

this(name, phoneNumber, 0);

}

public Contact(String name, String phoneNumber, int age) {

this.name = name;

this.phoneNumber = phoneNumber;

this.age = age;

}

public String getPhoneNumber() {

return phoneNumber;

}

public void setAge(int age) {

this.age = age;

}

public void show() {

System.*out*.println("name:"+name+"\tphone:"+phoneNumber+"\tage :"+age);

}

boolean search(String name) {

return this.name.equals(name);

}

boolean search(int age) {

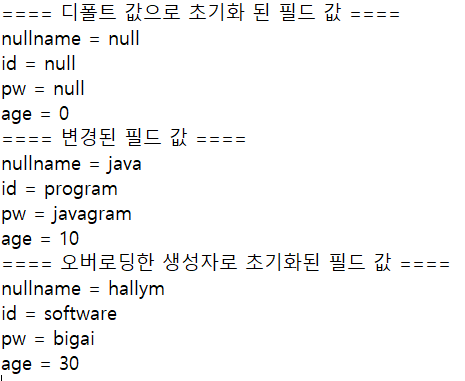
return this.age == age;

}

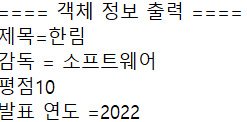
}

|  |  |
| --- | --- |
| 생성형 AI 플랫폼 | chatGPT |
| 질문 | 위의 클래스를 이용하여, 배열에 저장하는 클래스와 검색할 수 있는 클래스 만들어줘 |
| 답변 | 위의 Contact 클래스를 이용해, 여러 Contact 객체를 배열에 저장하고 검색할 수 있는 클래스를 작성해보겠습니다. 이 클래스에서는 배열에 Contact 객체를 저장하고, 이름 또는 나이로 검색할 수 있는 기능을 제공하도록 합니다.  **ContactManager 클래스**  이 클래스는 Contact 객체들을 저장할 배열을 관리하고, 이름과 나이로 검색하는 기능을 제공합니다.  public class ContactManager {  private Contact[] contacts;  private int count;  // 생성자: 배열의 크기를 정하고 초기화  public ContactManager(int size) {  contacts = new Contact[size];  count = 0;  }  // Contact 객체를 배열에 추가  public void addContact(Contact contact) {  if (count < contacts.length) {  contacts[count++] = contact;  } else {  System.out.println("더 이상 추가할 수 없습니다. 배열이 가득 찼습니다.");  }  }  // 이름으로 연락처를 검색  public Contact searchByName(String name) {  for (int i = 0; i < count; i++) {  if (contacts[i].search(name)) {  return contacts[i];  }  }  return null; // 찾지 못한 경우  }  // 나이로 연락처를 검색  public Contact searchByAge(int age) {  for (int i = 0; i < count; i++) {  if (contacts[i].search(age)) {  return contacts[i];  }  }  return null; // 찾지 못한 경우  }  // 연락처 목록 출력  public void showContacts() {  for (int i = 0; i < count; i++) {  contacts[i].show();  }  }  }  **Main 클래스 (테스트용)**  ContactManager 클래스와 Contact 클래스의 기능을 테스트할 수 있는 Main 클래스를 작성해봅니다.  public class Main {  public static void main(String[] args) {  // 연락처 관리 객체 생성  ContactManager manager = new ContactManager(5); // 최대 5개의 연락처 저장 가능  // 연락처 객체 생성  Contact contact1 = new Contact("Alice", "123-4567", 25);  Contact contact2 = new Contact("Bob", "234-5678", 30);  Contact contact3 = new Contact("Charlie", "345-6789", 22);  // 연락처 추가  manager.addContact(contact1);  manager.addContact(contact2);  manager.addContact(contact3);  // 모든 연락처 출력  System.out.println("모든 연락처:");  manager.showContacts();  // 이름으로 검색  String searchName = "Bob";  Contact resultByName = manager.searchByName(searchName);  if (resultByName != null) {  System.out.println("\n이름으로 찾은 결과: ");  resultByName.show();  } else {  System.out.println("\n이름 '" + searchName + "'에 해당하는 연락처를 찾을 수 없습니다.");  }  // 나이로 검색  int searchAge = 22;  Contact resultByAge = manager.searchByAge(searchAge);  if (resultByAge != null) {  System.out.println("\n나이로 찾은 결과: ");  resultByAge.show();  } else {  System.out.println("\n나이 '" + searchAge + "'에 해당하는 연락처를 찾을 수 없습니다.");  }  }  }  **설명:**   1. **ContactManager 클래스**: 이 클래스는 Contact 객체들을 배열에 저장하고, 연락처를 추가하는 기능(addContact), 이름으로 검색하는 기능(searchByName), 나이로 검색하는 기능(searchByAge), 저장된 모든 연락처를 출력하는 기능(showContacts)을 제공합니다. 2. **Main 클래스**: Main 클래스에서 ContactManager를 사용하여 Contact 객체들을 추가하고, 이름 및 나이로 검색할 수 있는 예시를 보여줍니다. |
| 실행 결과 | **실행 결과:**  이 프로그램을 실행하면 연락처 목록을 출력하고, 이름과 나이로 각각 검색한 결과를 출력합니다.  모든 연락처:  name:Alice phone:123-4567 age :25  name:Bob phone:234-5678 age :30  name:Charlie phone:345-6789 age :22  이름으로 찾은 결과:  name:Bob phone:234-5678 age :30  나이로 찾은 결과:  name:Charlie phone:345-6789 age :22  이 코드는 **ContactManager** 클래스에 연락처를 추가하고, 이름이나 나이로 검색하는 기능을 완벽하게 구현한 예제입니다. |

1. 다음과 같은 문제를 해결하는 프로그램을 제시된 조건에 따라 해결하세요
2. 현실 세계의 회원을 Member 클래스로 모델링하려고 합니다. 회원의 데이터로 이름, 아이디, 패스워드, 나이가 있습니다. 이 데이터들을 가지는 Member 클래스를 선언해 보세요
3. 1번에서 작성한 클래스를 사용하여 객체를 생성하는 문장을 작성해 보세요. 단, 디폴트 생성자호출
4. 1에서 생성한 객체의 필드 값을 출력한 후 임의의 값으로 초기화하세요
5. 이름, 아이디, 패스워드, 나이를 매개변수로 받아 필드로 초기화하는 생성자를 정의하세요
6. 4번의 생성자를 사용하여 객체를 생성하는 문장을 작성해 보세요
7. 5번에서 생성된 객체의 필드값을 출력하는 문장을 작성해 보세요.



|  |
| --- |
| [프로그램 소스] - 1번~6번에 해당하는 소스코드를 제시하면 됩니다.  package week2.task.java3;  class Member {  // 1.현실 세계의 회원을 Member 클래스로 모델링하려고 합니다. 회원의 데이터로 이름, 아이디, 패스워드, 나이가 있습니다. 이 데이터들을 가지는 Member 클래스를 선언해 보세요  private String name ;  private String id ;  private String password ;  private int age ;    // 2.디폴트 생성자 호출 ( 필드 멤버를 초기화하는 데이터를 생성자 매개변수로 받아 처리하여야 합니다 )  public Member() {  this.name = null ;  this.id = null ;  this.password = null ;  this.age = 0 ;  }    // 4.이름, 아이디, 패스워드, 나이를 매개변수로 받아 필드로 초기화하는 생성자를 정의하세요  // (오버로딩된 생성자를 사용하여 객체를 생성합니다)  public Member(String name, String id, String password, int age) {  this.name = name ;  this.id = id ;  this.password = password ;  this.age = age ;  }    // 3.1에서 생성한 객체의 필드 값을 출력한 후 임의의 값으로 초기화하세요  public void show() {  System.*out*.println("name: " + name + "\nid: " + id + "\npassword: " + password + "\nage: " + age);  }    // 설정자 메소드  public void setName (String name) {  this.name = name;  }    public void setId (String id) {  this.id = id;  }    public void setPassword (String password) {  this.password = password;  }    public void setAge (int age) {  this.age = age;  }  }  package week2.task.java3;  public class MemberTest {  public static void main(String[] args) {    System.*out*.println("==== 디폴트 값으로 초기화 된 필드 값 ====");  // 기본 생성자 사용  Member member = new Member();  member.show();    // 6. 5번에서 생성된 객체의 필드값을 출력하는 문장을 작성해 보세요.  member.setName("java");  member.setId("program");  member.setPassword("javagram");  member.setAge(10);  System.*out*.println("==== 변경된 필드 값 ====");  member.show();    // 5. 4번의 생성자를 사용하여 객체를 생성하는 문장을 작성해 보세요  System.*out*.println("==== 오버로딩한 생성자로 초기화된 필드 값 ====");  Member member2 = new Member("hallym", "software", "bigai", 30);  member2.show();  }    } |
| [실행 결과]  ==== 디폴트 값으로 초기화 된 필드 값 ====  name: null  id: null  password: null  age: 0  ==== 변경된 필드 값 ====  name: java  id: program  password: javagram  age: 10  ==== 오버로딩한 생성자로 초기화된 필드 값 ====  name: hallym  id: software  password: bigai  age: 30 |

1. 영화 Movie 클래스를 정의하여 하시오. Movie 클래스는 영화 제목, 평점, 감독, 발표된 연도 등의 필드를 가진다. Movie 클래스를 작성하고 객체를 생성하여 모든 필드 정보를 출력한다. 필드는 객체 생성시 생성자를 호출하여 초기화 한다.

|  |
| --- |
| [프로그램 소스]  package week2.task.java4;  class Movie {    private String title ; // 제목  private String director ; // 감독  private String year ; // 발표된 연도  private int rating ; // 평점    // 생성자 호출하여 초기화  public Movie(String title, String director, String year, int rating) {    this.title = title ;  this.director = director ;  this.year = year ;  this.rating = rating ;    }    // 모든 필드 정보 출력 메소드  public void show() {    System.*out*.println("영화 제목: " + title + "\n감독: " + director + "\n평점: " + rating + "\n발표 연도: " + year);    }    }  package week2.task.java4;  public class MovieTest {  public static void main(String[] args) {    Movie movie = new Movie("한림", "소프트웨어", "2022", 10);    System.*out*.println("=== 객체 정보 출력 ===");  movie.show();    }  } |
| [실행 결과]  === 객체 정보 출력 ===  영화 제목: 한림  감독: 소프트웨어  평점: 10  발표 연도: 2022 |

1. 아이디는 키보드로 입력 받으며, 비밀번호는 4자리수의 정수형 난수로 초기화하는 Info 클래스를 제시된 조건대로 작성하시오. 힌트) 난수 생성 : (int)(Math.*random*()\*100)+1; //1~100사이 난수 생성

* 필드 구성

- id : String, 아이디 저장

- pass : int, 비밀번호 저장

* 생성자 : 아이디 필드는 매개변수로 받은 값으로 초기화하고, 난수를 생성하여 비밀번호를 초기화

Info 객체를 생성하고 테스트하는 InfoTest 클래스를 작성 하시오.

* main() 메소드

- Info 객체 두개를 선언하고 아이디는 입력 받아서 생성자 매개변수로 전달

- 두 개의 객체 내용 출력

- 두번째 객체의 비밀번호를 난수를 사용하여 변경한 후 출력



|  |
| --- |
| [프로그램 소스]  package week2.test;  import java.util.Scanner;  // 1개의 파일에 2개의 클래스가 있는 경우 public은 1번만 사용해야 한다.  class InfoT {    private String id;  private int pass;    // 생성자는 누구든지 접근가능해야하므로 public  public InfoT(String id) {  this.id = id; // 필드 앞에 this를 붙여준다.(필드와 매개변수를 구분하기 위해서)  pass = password();  }    // 메소드  public String toString() {  return "id="+id+"\tpass="+pass;  }    public void changePass() {  pass = password();  }    private int password() {  return (int)(Math.*random*()\*9000)+1000;  }  }  public class InfoTestT {  // 코드 실행 순서는 메인 영역만 위에서 아래 순서이고, 클래스는 호출하면 실행되는 형식이다.  public static void main(String[] args) {  Scanner key = new Scanner(System.*in*);  InfoT info1=null; // 참조 변수 같은 경우는 null값으로 초기화 해주어야 오류 발생하지 않는다.    System.*out*.print("id input >> ");  String id = key.next(); // 아이디를 입력받는 변수  info1 = new InfoT(id);  System.*out*.println(info1); // toString()은 오버라이딩이 되어있다(생략 가능)    info1.changePass();    System.*out*.println("=======================");  System.*out*.println(info1);    key.close(); // Scanner를 닫아주는 의미    }  } |
| [실행 결과]  id input >> hallym  id=hallym pass=9312  =======================  id=hallym pass=4897 |

1. **[생성형 AI 활용]** **AI가 제시하는 클래스 활용 문제를 프로그램하고 결과를 제시하세요. AI가 문제 해결을 위한 코드는 답변에서 제외 되어야 합니다**

|  |  |
| --- | --- |
| 생성형 AI 플랫폼 | chatGPT |
| 질문 | (6) [생성형 AI 활용] AI가 제시하는 클래스 활용 문제를 프로그램하고 결과를 제시하세요. AI가 문제 해결을 위한 코드는 답변에서 제외 되어야 합니다 |
| 답변 | AI를 활용한 문제를 해결하는 방법은 다양한 방식으로 접근할 수 있습니다. 아래는 생성형 AI를 사용하여 클래스를 활용한 문제 해결 예시입니다.  **문제 예시:**  **문제**: 학생의 성적을 관리하는 프로그램을 작성하세요. 이 프로그램에서는 학생 클래스(Student)를 정의하고, 각 학생의 이름, 나이, 성적 등을 속성으로 가지며, 성적 평균을 계산하는 메서드를 포함해야 합니다. 여러 명의 학생 데이터를 처리할 수 있도록 배열로 학생 객체를 관리해야 하며, 최종적으로 전체 학생 정보와 전체 성적 평균을 출력합니다.  **해결 방법:**   1. **클래스 설계**:    * Student 클래스에는 학생의 이름, 나이, 성적 정보를 담을 수 있도록 속성(attribute)을 정의합니다.    * 성적 평균을 계산하는 메서드(calculate\_average)를 클래스 안에 정의합니다. 2. **학생 목록 관리**:    * 학생 객체를 배열에 저장하여 여러 학생을 관리할 수 있게 합니다. 3. **성적 평균 계산**:    * 배열에 저장된 학생들의 성적 평균을 계산하여 전체 평균 점수를 출력합니다. 4. **결과 출력**:    * 저장된 학생들의 정보와 전체 평균 점수를 출력합니다. |

|  |
| --- |
|  |
| [프로그램 소스]  package week2.task.java6;  class Student {    // 학생 이름, 나이, 점수 객체 선언  private String name ;  private int age ;  private int[] scores;  // 생성자  public Student(String name, int age, int[] scores) {  this.name = name ;  this.age = age ;  this.scores = scores;  }    // 성적 평균 메소드  public int average() {  if(scores.length == 0) {  return 0;  }  int total = 0;    for(int i=0; i<scores.length; i++) {  total += scores[i];  }  return total / scores.length;  }    // 성적 평균출력 메소드  public void averageShow() {  int avarage = average();  System.*out*.println("전체 성적 평균: " + avarage);  System.*out*.println();  }    // 학생 출력 메소드  public void show() {  System.*out*.print("이름: " + name + ",\t나이: " + age + ",\t성적: ");  for(int i=0; i < scores.length; i++) {  if(i == 0) {  System.*out*.print(scores[i] + "점["+(i+1)+"번째]");  } else {  System.*out*.print(", "+ scores[i] + "점["+(i+1)+"번째]");  }  }  System.*out*.println();  }    } |
| package week2.task.java6;  import java.util.Scanner;  public class StudentManagerTest {  public static void main(String[] args) {    Scanner input = new Scanner(System.*in*);  // 학생 수 입력  System.*out*.print("학생수를 입력하세요:");  int numStudents = input.nextInt();    // 학생 배열 선언  Student[] students = new Student[numStudents];    // 학생 배열 값 넣기  for(int i=0; i < numStudents; i++) {    System.*out*.println("========" + (i+1) + "번째 학생 " + "정보 입력========");    // 학생 이름 입력  System.*out*.print("이름: ");  String name = input.next();    // 학생 나이 입력  System.*out*.print("나이: ");  int age = input.nextInt();    System.*out*.println("======================================");    // 성적 수 입력  System.*out*.print((i+1)+"번째 학생의 성적 개수를 입력하세요:");  int numScores = input.nextInt();    // 성적 배열 선언  int[] scores = new int[numScores];    // 성적 배열 값 넣기  for(int j=0; j < numScores; j++) {  System.*out*.print((j+1)+"번째 성적: ");  scores[j] = input.nextInt();  }  students[i] = new Student(name, age, scores);  }    System.*out*.println("=======전체 학생 정보 ========");  for(Student student : students) {  student.show(); // 전체 학생 정보 출력  student.averageShow(); // 학생 평균 출력  }  input.close();    }  } |
| [실행 결과]  학생수를 입력하세요:2  ========1번째 학생 정보 입력========  이름: 홍길동1  나이: 23  ======================================  1번째 학생의 성적 개수를 입력하세요:2  1번째 성적: 90  2번째 성적: 70  ========2번째 학생 정보 입력========  이름: 홍길동2  나이: 21  ======================================  2번째 학생의 성적 개수를 입력하세요:3  1번째 성적: 60  2번째 성적: 70  3번째 성적: 95  =======전체 학생 정보 ========  이름: 홍길동1, 나이: 23, 성적: 90점[1번째], 70점[2번째]  전체 성적 평균: 80  이름: 홍길동2, 나이: 21, 성적: 60점[1번째], 70점[2번째], 95점[3번째]  전체 성적 평균: 75 |