|  |  |  |
| --- | --- | --- |
| java2\_week03\_클래스(2) | 학번 : 20237107 | 이름 : 하태영 |

* 응용 프로그래밍(1,2,3,4,5,7)

1. **[생성형 AI 활용]** 객체의 집합 관계와 사용관계, 정적메소드를 모두 활용하는 예를 제시하세요

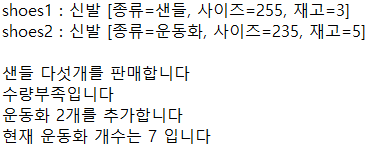
|  |  |
| --- | --- |
| 생성형 AI 플랫폼 | SmartLead |
| 질문 | 자바와 배열을 사용하여 객체의 집합 관계와 사용관계, 정적메소드를 모두 활용하는 예를 제시하세요 |
| 답변 | // Engine 클래스: 자동차의 엔진을 나타내는 클래스  class Engine {      private String type;      public Engine(String type){          this.type = type;      }      public String getType() {          return type;      }  }  // Car 클래스: 자동차를 나타내는 클래스  class Car {      private Engine engine; // 집합 관계: Car는 Engine을 포함      private String model;      public Car(String model, Engine engine){          this.model = model;          this.engine = engine;      }      public void displayCarInfo() {          System.out.println("Car Model: " + model);          System.out.println("Engine Type: " + engine.getType());      }      // 정적 메소드: 여러 자동차 정보를 배열을 통해 출력      public static void displayCarsInfo(Car[] cars) {          for (Car car : cars) {              car.displayCarInfo();              System.out.println("---------------");          }      }  }  public class Main {      public static void main(String[] args) {          Engine engine1 = new Engine("V8");          Engine engine2 = new Engine("Electric");          Car car1 = new Car("Mustang", engine1);          Car car2 = new Car("Tesla", engine2);          Car[] cars = {car1, car2}; // 사용 관계: 메인 메소드에서 Car 배열을 사용          Car.displayCarsInfo(cars); // 정적 메소드 호출      }  } |
| 실행결과 |  |

1. **[생성형 AI 활용]** 단 한 개의 게임 아이템을 생성하는 클래스를 작성하고테스트 하세요

|  |  |
| --- | --- |
| 생성형 AI 플랫폼 | SmartLead |
| 질문 | 자바를 이용하여 단 한 개의 게임 아이템을 생성하는 클래스를 작성하고테스트 하세요 |
| 답변 | **public** **class** GameItem {  // 이른 초기화 방식으로 싱글톤 인스턴스를 생성  **private** **static** GameItem *instance* = **new** GameItem();    // 인스턴스 생성을 방지하기 위한 private 생성자  **private** GameItem() {  // 초기화 코드 여기에  }  // 인스턴스에 대한 접근을 제공하는 공개 메소드  **public** **static** GameItem getInstance() {  **return** *instance*;  }  // GameItem 클래스의 다른 메소드와 속성  **public** **void** useItem() {  // 아이템을 사용하는 코드  System.***out***.println("아이템 사용!");  }  }  **public** **class** Main {  **public** **static** **void** main(String[] args) {  // GameItem의 싱글톤 인스턴스를 가져옴  GameItem item = GameItem.*getInstance*();    // 아이템 사용  item.useItem();    // 또 다른 인스턴스를 가져와 사용 시도  GameItem anotherItem = GameItem.*getInstance*();  anotherItem.useItem();    // 두 참조 변수가 같은 인스턴스를 가리키는지 확인  **if** (item == anotherItem) {  System.***out***.println("두 참조 변수가 같은 인스턴스를 가리킵니다.");  } **else** {  System.***out***.println("참조 변수가 다른 인스턴스를 가리킵니다.");  }  }  } |
| 실행결과 |  |

1. 신발 매장에서 상품관리를 위한 클래스를 작성하고 테스트하시오

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

 Shoes shoes1=**new** Shoes(255, 3, "샌들" );

Shoes shoes2=**new** Shoes(235, 5, "운동화" );

System.**out**.printf("shoes1 : %s\n",shoes1);

System.**out**.printf("shoes2 : %s\n\n",shoes2);

System.**out**.println("샌들 다섯개를 판매합니다");

**if**(shoes1.sell(5))

System.**out**.println("현재 샌들의 개수는 " + shoes1.getCnt() + " 입니다");

**else**

System.**out**.println("수량부족입니다");

System.**out**.println("운동화 2개를 추가합니다");

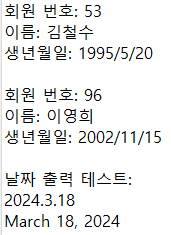
shoes2.add(2);

System.**out**.println("현재 운동화 개수는 " + shoes2.getCnt() + " 입니다");

}

|  |
| --- |
| [프로그램 소스]  package week3.practice3;  class Shoes {  private int size;  private int cnt;  private String type;  public Shoes(int size, int cnt, String type) {  this.size = size;  this.cnt = Math.max(cnt, 0);  this.type = (type != null) ? type : "알 수 없음";  }    int getSize() {  return size;  }    int getCnt() {  return cnt;  }    String getType() {  return type;  }    boolean sell(int cnt) {  if(cnt <= 0 || this.cnt < cnt) {  return false;  }  this.cnt -= cnt;  return true;  }  int add(int cnt) {  if(cnt <=0) {  return this.cnt = cnt;  } else {  return this.cnt += cnt;  }  }    @Override  public String toString() {  return "신발 [종류=" + type + ", 사이즈=" + size + ", 재고=" + cnt +"]";  }  } |
| [실행 결과] |

1. 날짜를 나타내는 클래스 Date를 만들어보자. Date는 연도, 월, 일 등의 속성을 가지며, 날짜를 “2012.7.12”과 같이 출력하는 메소드 print1(), 날짜를 “July 12, 2012”와 같이 출력하는 print2() 등의 메소드를 갖는다. 회원정보를 나타내는 Member 클래스를 만들어 보자. 회원 번호(int, 두 자리 난수)와 이름(String), 생년월일(Date 클래스) 속성을 가지며 객체 정보를 문자열로 반환하는 toString() 메소드를 갖는다. 테스트 하고 실행 하시오

public class Test {

public static void main(String[] args) {

// 회원 객체 생성 (이름과 생년월일 입력)

Member member1 = new Member("김철수", 1995, 5, 20);

Member member2 = new Member("이영희", 2002, 11, 15);

// 회원 정보 출력 (toString() 메서드 자동 호출)

System.*out*.println(member1);

System.*out*.println();

System.*out*.println(member2);

// 날짜 출력 테스트

System.*out*.println("\n날짜 출력 테스트:");

Date date = new Date(2024, 3, 18);

date.print1(); // 2024.3.18

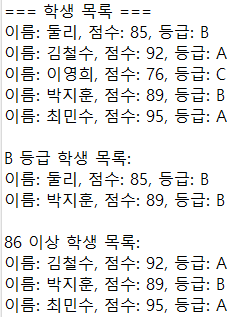
date.print2(); // March 18, 2024

}

}

|  |
| --- |
| [프로그램 소스]  package week3.practice4;  class Date {  // 필드 선언  private int year, month, day;    // 생성자  public Date(int year, int month, int day) {  this.year = year;  this.month = month;  this.day = day;  }    public void print1() {  System.*out*.println(year + "." + month + "." + day);    }  public void print2() {  String[] strMonth= {"Januar", "February", "March", "April", "May", "June", "July",  "August", "September", "October", "November", "December"};      System.*out*.println(strMonth[month-1] + " " + day + ", " + year);  }  public String toString() {  return year+"/"+month+"/"+day;  }  }  package week3.practice4;  class Member {  private int memberId;  private String name;  private Date birthDate; // 집합 관계    public Member(String name, int year, int month, int day) {  this.memberId=(int)(Math.*random*()\*90)+10;  this.name = name;  this.birthDate = new Date(year, month, day); // 사용 관계  }    @Override  public String toString() {  return "회원 번호: " + memberId + "\n이름: " + name + "\n생년월일: " + birthDate;    }  } |
| [실행 결과] |

1. 학생을 나타내는 클래스 Student를 만들어보자. 학생은 이름(private name)과 점수(private score), 등급(private grade)을 가진다. 객체 내용(이름, 점수, 등급)을 문자열로 반환하는 메소드 toString()를 가진다. 이름과 점수를 전달하는 생성자도 필요하다. 점수를 변경하는 메소드와 등급과 점수를 반환하는 메소드도 필요하다. 제시한 점수 이상의 학생을 검색하는 정적 메소드와 특정 등급의 학생만 검색하는 정적 메소드를 멤버로 갖는 클래스를 작성한다. 필드는 외부에 공개하지 않는다

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

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

// 1. 이름 없는 학생 객체 생성 후 점수 설정

Student student1 = **new** Student("둘리");

student1.setScore(85);

// 2. 이름과 점수를 전달한 학생 객체 생성

Student student2 = **new** Student("김철수", 92);

// 3. 배열에 5명의 학생 저장

Student[] students = { student1, student2, **new** Student("이영희", 76), **new** Student("박지훈", 89),

**new** Student("최민수", 95) };

// 4. 모든 학생 정보 출력

System.***out***.println("\n=== 학생 목록 ===");

**for** (Student student : students) {

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

}

// 5. 등급과 성적으로 학생 정보 출력

StudentControl.*searchGrade*(students, 'B');

StudentControl.*searchScore*(students, 86);

}

}

|  |
| --- |
| [프로그램 소스]  package week3.practice5;  public class Student {  // 학생은 이름(private name)과 점수(private score), 등급(private grade)을 가진다.  private String name, grade;  private int score;    // 이름과 점수를 전달하는 생성자도 필요하다.  public Student(String name) { // 점수, 등급 초기화  this.name = name;  this.score = 0;  this.grade = "F";  }  public Student(String name, int score) { // 이름, 점수 입력 받음  this.name = name;  this.score = score;  this.grade = calculateGrade(score);  }    // 점수를 변경하는 메소드와 등급과 점수를 반환하는 메소드도 필요하다.  public String calculateGrade(int score) {  switch (score / 10) {  case 10: return "A";  case 9: return "A";  case 8: return "B";  case 7: return "C";  case 6: return "D";  default: return "F";  }  }    public void setScore(int score) {  this.score = score;  this.grade = calculateGrade(score);  }    public int getScore() {  return score;  }    public String getName() {  return name;  }    public String getGrade() {  return grade;  }  // 객체 내용(이름, 점수, 등급)을 문자열로 반환하는 메소드 toString()를 가진다.  @Override  public String toString() {  return "이름:" + name + ", 점수:" + score + ", 등급:" + grade;  }  }  package week3.practice5;  public class StudentControl {  public static void searchGrade(Student[] students, char grade) {  System.*out*.println("\n" + grade + " 등급 학생 목록:");    for(Student student : students) {  if (student.getGrade().charAt(0) == grade) { // getGrade().charAt(0) 문자열->문자 형변환하고, 첫번째 값을 반환  System.*out*.println(student);  }  }    }  public static void searchScore(Student[] students, int score) {  System.*out*.println("\n점수 " + score + "이상 학생 목록:");  for(Student student : students) {  if (student.getScore() >= score) {  System.*out*.println(student);  }  }  }    } |
| [실행 결과] |

1. 아이디는 키보드로 입력 받으며, 비밀번호는 아이디+객체생성순서+2자리수 정수형 난수를 연결하여 초기화하는 Info 클래스를 제시된 조건대로 작성하고 테스트 하세요.

* 필드 구성 – 외부에 공개하지 않는다

- id : String, 아이디 저장

- pass : String, 비밀번호 저장

* 생성자 : 아이디 필드는 매개변수로 받은 값으로 초기화하고, 비밀번호는 아이디 +객체 생성순서 + 두자리 난수로 초기화
* 메소드 구성

- 객체 내용을 문자열로 반환하는 toString()

- 필드 값을 변경하고 반환하는 getter, setter()

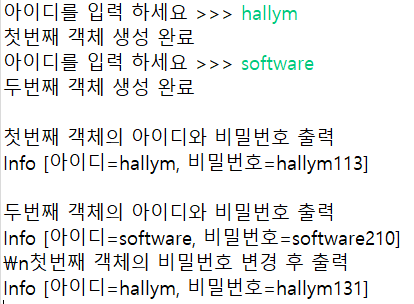
Info 객체를 생성하고 테스트하는 InfoTest 클래스를 작성 하시오.

* main() 메소드

- Info 객체 두개를 선언하고 아이디는 입력 받아서 생성자 매개변수로 전달

- 두 개의 객체 내용 출력

- 첫번째 객체의 비밀번호를 난수를 사용하여 변경한 후 출력



|  |
| --- |
| [프로그램 소스]  package week3.practice6;  class Info {  // id, pass 객체 선언  private String id, pass;  private static int *count* = 0;  // 생성자  public Info(String id) {  this.id = id;  *count*++; // 객체가 생성될 때마다 증가  this.pass = generatePassword(id, *count*);  }    // 비밀번호 생성 메서드 (Math.random() 사용)  private String generatePassword(String id, int count) {  int randomNum = (int)(Math.*random*() \* 90) + 10; // 10~99 사이의 난수 생성  return id + count + randomNum;  }    // getter, setter  public String getId() {  return id;  }  public void setId(String id) {  this.id = id;  }  public String getPass() {  return pass;  }  public void setPass() {  this.pass = generatePassword(this.id, *count*);  }  // toString  @Override  public String toString() {  return "Info [아이디=" + id + ", 비밀번호=" + pass + "]";  }    }  package week3.practice6;  import java.util.Scanner;  public class InfoTest {  public static void main(String[] args) {    Scanner sc = new Scanner(System.*in*);    // 첫 번째 아이디 입력 및 객체 생성  System.*out*.print("아이디를 입력 하세요 >>> ");  String id1 = sc.next();  Info info1 = new Info(id1);  System.*out*.println("첫번째 객체 생성 완료");    // 두 번째 아이디 입력 및 객체 생성  System.*out*.print("아이디를 입력 하세요 >>> ");  String id2 = sc.next();  Info info2 = new Info(id2);  System.*out*.println("두번째 객체 생성 완료");    System.*out*.println("\n첫번째 객체의 아이디와 비밀번호 출력");  System.*out*.println(info1);  System.*out*.println("\n두번째 객체의 아이디와 비밀번호 출력");  System.*out*.println(info2);    info1.setPass();    System.*out*.println("\n첫 번째 객체의 비밀번호 변경 후 출력");  System.*out*.println(info1);    sc.close();  }  } |
| [실행 결과] |

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

|  |  |
| --- | --- |
| 생성형 AI 플랫폼 | ChatGPT |
| 질문 | AI가 제시하는 객체 배열 활용 문제를 프로그램하고 결과를 제시하세요. 문제 해결을 위한 코드는 답변에서 제외 되어야 합니다 |
| 답변 | 문제  객체 배열을 사용한 도서 관리 시스템 구현  배경  책 정보를 관리하기 위한 시스템을 만들고자 합니다.  각각의 책은 제목, 저자, 발행년도, 가격 정보를 가지고 있습니다.  사용자가 책을 추가하고, 가격이 특정 금액 이상인 책들을 찾고, 저자별로 책 목록을 조회하는 기능을 구현해야 합니다.  요구 사항  Book 클래스: 각 책은 제목, 저자, 발행년도, 가격을 속성으로 가지는 클래스입니다.  도서 목록: 여러 개의 Book 객체를 배열에 저장합니다.  책 추가: 사용자가 책을 추가할 수 있도록 합니다.  가격 조건 검색: 특정 가격 이상인 책들만 출력할 수 있는 기능을 구현합니다.  저자별 책 조회: 특정 저자의 책들을 조회할 수 있는 기능을 구현합니다.  **입력 예시**:  책 추가: "The Great Gatsby", "F. Scott Fitzgerald", 1925, 10000  책 추가: "1984", "George Orwell", 1949, 15000  책 추가: "To Kill a Mockingbird", "Harper Lee", 1960, 12000  책 추가: "Animal Farm", "George Orwell", 1945, 8000  가격 12000 이상 책 검색  => "1984" by George Orwell, 15000  => "To Kill a Mockingbird" by Harper Lee, 12000  "George Orwell"의 책들 조회  => "1984" by George Orwell, 15000  => "Animal Farm" by George Orwell, 8000 |

|  |
| --- |
|  |
| [프로그램 소스]  package week3.practice7;  class Book {  // 제목, 저자, 발행년도, 가격  private String title, writer;  private int year, price;    // 생성자  public Book(String title, String writer, int year, int price) {  this.title = title;  this.writer = writer;  this.year = year;  this.price = price;  }    // getter, setter  public String getTitle() {  return title;  }  public void setTitle(String title) {  this.title = title;  }  public String getWriter() {  return writer;  }  public void setWriter(String writer) {  this.writer = writer;  }  public int getYear() {  return year;  }  public void setYear(int year) {  this.year = year;  }  public int getPrice() {  return price;  }  public void setPrice(int price) {  this.price = price;  }    // toString  @Override  public String toString() {  return "=> " + title + " by " + writer + ", " + price;  }    }  package week3.practice7;  public class BookControl {  public static void searchPrice(Book[] books, int price) {  System.*out*.println("가격 " + price + "이상 책 검색");    for(Book book : books) {  if (book.getPrice() >= price) {  System.*out*.println(book);  }  }  }  public static void searchTitle(Book[] books, String writer) {  System.*out*.println("\n" + writer + "의 책들 조회");    for(Book book : books) {  if (book.getWriter().equals(writer)) {  System.*out*.println(book);  }  }  }  }  package week3.practice7;  import java.util.Scanner;  /\* 입력 예시  1. The Great Gatsby, F. Scott Fitzgerald, 1925, 10000  2. 1984, George Orwell, 1949, 15000  3. To Kill a Mockingbird, Harper Lee, 1960, 12000  4. Animal Farm, George Orwell, 1945, 8000  \*/  public class Test {  public static void main(String[] args) {    String title, writer;  int year, price;  Book[] books = new Book[4];    Scanner sc = new Scanner(System.*in*);    for(int i = 0; i < 4; i++) {  System.*out*.print("책 추가: ");  String str = sc.nextLine();  String[] arr = str.split(",");    title = arr[0].trim();  writer = arr[1].trim();  year = Integer.*parseInt*(arr[2].trim());  price = Integer.*parseInt*(arr[3].trim());    books[i] = new Book(title, writer, year, price);  }  BookControl.*searchPrice*(books, 12000);  BookControl.*searchTitle*(books, "George Orwell");    sc.close();  }  } |
| [실행 결과] |