

# ExamA1. 자바 입출력과 main() 프로그램 작성

## ● 내용

- package : exam
- 학생의 이름과 과목을 문자열로 입력 받는다.
- 성적은 정수로 입력 받는다.
- 성적이 60점 미만이면 과락, 이상이면 합격
- 입력 받은 데이터와 통과 여부를 console에 출력

## ● 입력 예

- 이름 : 홍길동
- 과목 : 국어
- 성적 : 75

## ● 출력 예

- 이름 = 홍길동, 과목 = 국어, 성적 = 75, 통과 = 합격

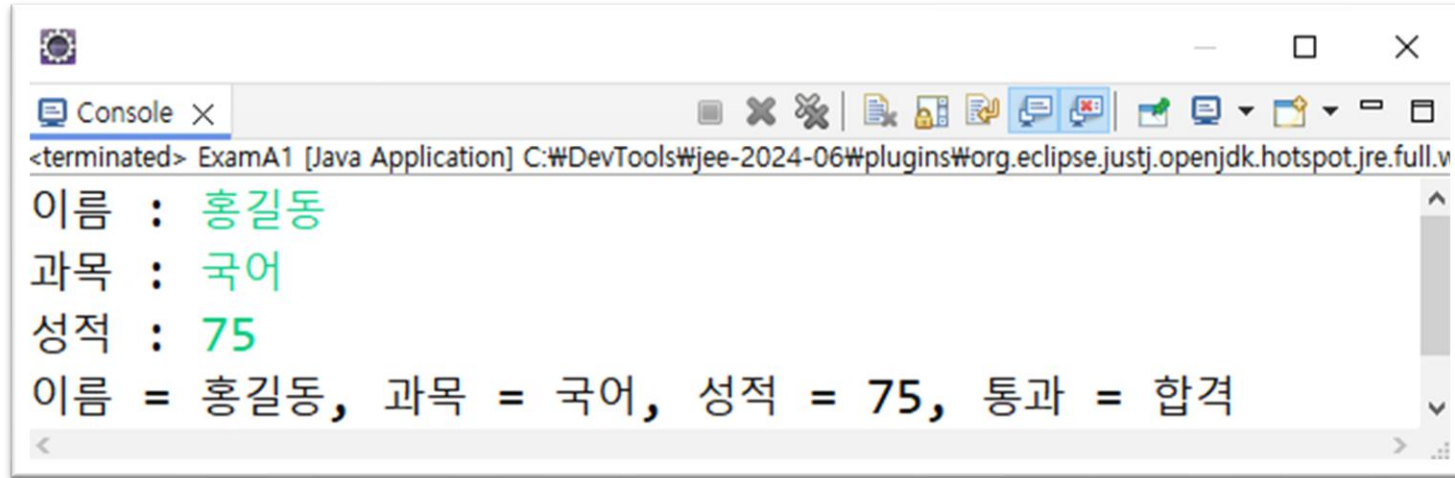
```
// 키보드 입력 객체 생성
Scanner sc = new Scanner(System.in);

// 키보드로 문자열 입력
sc.next();

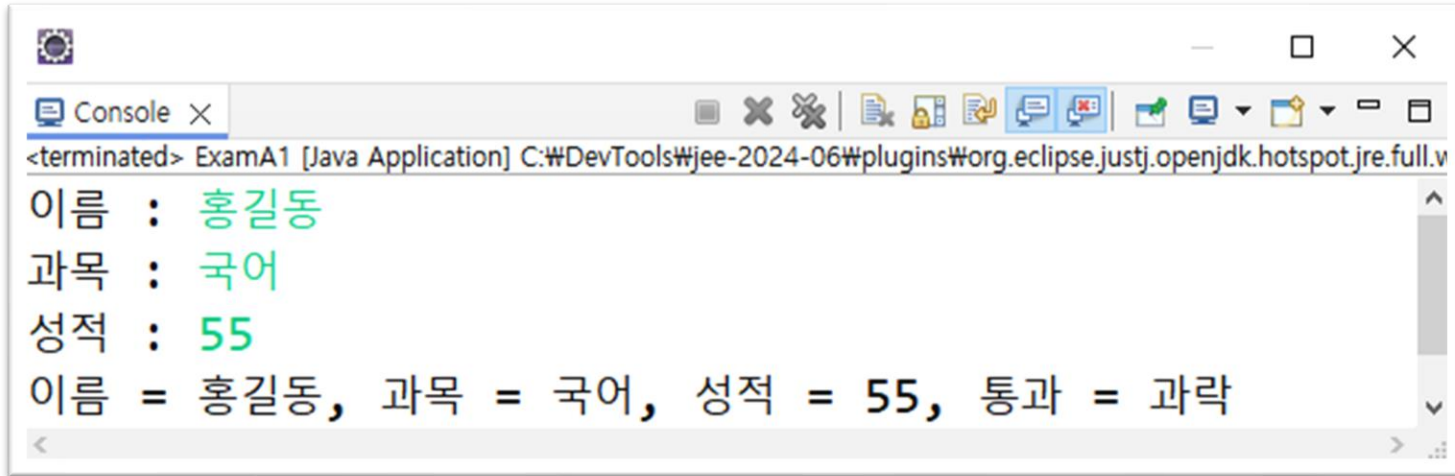
// 키보드로 정수 입력
sc.nextInt();
```

# ExamA1. 자바 입출력과 main() 프로그램 작성

- 실행 예



```
<terminated> ExamA1 [Java Application] C:\DevTools\jee-2024-06\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.v
이름 : 홍길동
과목 : 국어
성적 : 75
이름 = 홍길동, 과목 = 국어, 성적 = 75, 통과 = 합격
```



```
<terminated> ExamA1 [Java Application] C:\DevTools\jee-2024-06\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.v
이름 : 홍길동
과목 : 국어
성적 : 55
이름 = 홍길동, 과목 = 국어, 성적 = 55, 통과 = 과락
```

# ExamA2. 조건문과 반복문


- 내용

- ExamA1.java를 ExamA2.java로 복사해서 수정한다.
- 과목별로 과락/통과 기준을 다르게 설정한다.
  - 국어 : 60
  - 수학 : 50
  - 영어 : 70
  - **if-else** 또는 **switch**
  - 성적이 0보다 작거나 100보다 크면 **continue**
- 학생 5명을 반복해서 입력 받아서 출력
  - **while / for / do-while**
  - **break (optional)**

- 입력/출력은 exam1.java와 동일

## ExamA2. 조건문과 반복문

- 실행 예



```
<terminated> ExamA2 [Java Application] C:\Program Files\Java\jdk-21.0.4\bin\javaw.exe (2024. 8. 24. 오후)
이름[1] : 홍길동
국어, 수학, 영어 : 100 100 100
이름 = 홍길동, 국어 = 100, 수학 = 100, 영어 = 100 : 통과
이름[2] : 홍이동
국어, 수학, 영어 : 90 90 90
이름 = 홍이동, 국어 = 90, 수학 = 90, 영어 = 90 : 통과
이름[3] : 홍삼동
국어, 수학, 영어 : 80 80 80
이름 = 홍삼동, 국어 = 80, 수학 = 80, 영어 = 80 : 통과
이름[4] : 홍사동
국어, 수학, 영어 : 70 70 70
이름 = 홍사동, 국어 = 70, 수학 = 70, 영어 = 70 : 통과
이름[5] : 홍오동
국어, 수학, 영어 : 60 60 60
이름 = 홍오동, 국어 = 60, 수학 = 60, 영어 = 60 : 과락
End~~
```

# ExamA3. 배열

## ● 내용

- 학생 5명의 이름을 1차원 문자열 배열로 정의
  - `String[] names = { "홍길동", "김유신", "계백", "강감찬", "을지문덕" }`
- 5개의 교과목 이름을 1차원 문자열 배열로 정의
  - `String[] subjs = { "수학", "국어", "영어", "과학", "역사" }`
- 학생별 과목별 점수를 2차원 배열로 정의
  - `int[][] scores = { { 85, 90, 78, 88, 92 }, { 75, 80, 85, 90, 95 }, { 65, 70, 75, 80, 85 }, { 95, 92, 88, 84, 91 }, { 88, 76, 85, 79, 90 } }`

## ● 출력

- 삼항연산자/switch~case 문을 이용하여 학생별, 과목별 점수, 통과여부 출력
- 5개 과목의 총점 및 평균 점수를 테이블로 출력
- 과목별 최대, 최소 점수를 구하고, 해당 점수의 학생 이름을 테이블로 출력

# ExamA3. 배열

## ● 실행 예

>>>학생별, 과목별 점수, 통과여부 출력

[홍길동]

수학 : 85 => 통과  
국어 : 90 => 통과  
영어 : 78 => 통과  
과학 : 88 => 통과  
역사 : 92 => 통과

[김유신]

수학 : 75 => 통과  
국어 : 80 => 통과  
영어 : 85 => 통과  
과학 : 90 => 통과  
역사 : 95 => 통과

[계백]

수학 : 65 => 통과  
국어 : 70 => 통과  
영어 : 75 => 통과  
과학 : 80 => 통과  
역사 : 85 => 과락

[강감찬]

수학 : 95 => 통과  
국어 : 92 => 통과  
영어 : 88 => 통과  
과학 : 84 => 통과  
역사 : 91 => 통과

[을지문덕]

수학 : 88 => 통과  
국어 : 76 => 통과  
영어 : 85 => 통과  
과학 : 79 => 과락  
역사 : 90 => 통과

>>>과목 총점 및 평균 점수를 테이블로 출력

이름	수학	국어	영어	과학	역사
홍길동	85	90	78	88	92
김유신	75	80	85	90	95
계백	65	70	75	80	85
강감찬	95	92	88	84	91
을지문덕	88	76	85	79	90

>>>과목별 최대, 최소 점수를 구하고, 해당 점수의 학생 이름을 테이블로 출력

과목	점수	이름
수학	95	강감찬
국어	92	강감찬
영어	88	강감찬
과학	90	김유신
역사	95	김유신

End~~

# ExamA4. 클래스 객체 배열

## ○ 내용

- 클래스 Student를 사용하여 학생 이름, 과목배열, 성적배열을 객체로 생성
- 필드
  - name, subjects, scores를 모두 private으로 선언 → Getter/Setter를 이용한 접근
- 메서드
  - average(), sumScore(), sumScore(int bounds), averageScore(), isPassed() 를 구현

## ○ 출력

- 각 학생의 성적 테이블 출력: 학생 이름, 총점, 평균, 과목별 점수, 과목별 통과여부
- 과목별 최대/최소 점수 및 해당 학생 이름 출력: 과목명, 최대 점수, 최소 점수, 최대점수 학생, 최소점수 학생

```
// 과목 배열 정의
String[] subjects = {"수학", "국어", "영어", "과학", "역사"};
// 과목별 통과 기준 배열 정의
int[] criteria = { 50, 60, 70, 80, 90 };
// 학생 배열 생성
Student[] students = {
    //"홍길동", "김유신", "계백", "강감찬", "을지문덕"
    new Student("홍길동", subjects, criteria,
        new int[]{85, 90, 78, 88, 92}),
    new Student("김유신", subjects, criteria,
        new int[]{75, 80, 85, 90, 95}),
    new Student("계백", subjects, criteria,
        new int[]{65, 70, 75, 80, 85}),
    new Student("강감찬", subjects, criteria,
        new int[]{95, 92, 88, 84, 91}),
    new Student("을지문덕", subjects, criteria,
        new int[]{88, 76, 85, 79, 90})
};
```

# ExamA4. 클래스 객체 배열

- 출력 예

```
<terminated> ExamA4 [Java Application] C:\Program Files\Java\jdk-21.0.4\bin\javaw.exe (2024. 8. 24. 오후 3:06:00 - 오
>>>각 학생의 성적 테이블 출력
=====
이름      총점      평균      수학      국어      영어      과학      역사
-----
홍길동    433      86      85      90      78      88      92
김유신    425      85      75      80      85      90      95
계백      375      75      65      70      75      80      85(과락)
강감찬    450      90      95      92      88      84      91
을지문덕  418      83      88      76      85      79(과락) 90
=====
>>>과목별 최대/최소 점수 및 해당 학생 이름 출력
=====
과목      최고점      최저점      최고점학생      최저점학생
-----
수학      95      65      강감찬      계백
국어      92      70      강감찬      계백
영어      88      75      강감찬      계백
과학      90      79      김유신      을지문덕
역사      95      85      김유신      계백
=====
```



# ExamA5. 정적 메서드

- 내용

- ExamA4에서 구현한 기능을 정적 메서드로 구현한다.
- 각 학생의 성적 테이블을 출력하는 정적 메서드를 구현한다.
  - `public static void printScore1();`
- 과목별 최대/최소 점수 및 해당 학생을 출력하는 정적 메서드를 구현한다.
  - `public static void printScore2();`

# ExamB1. 온라인 상점 관리 시스템 구현

## ○ 내용

- 1) Item 클래스를 생성하고, 이름, 가격, 재고량을 초기화하는 생성자를 작성하세요.
- 2) 재고를 줄이거나 늘리는 메소드를 구현하세요.
- 3) 메소드를 호출해 실제로 재고를 관리하는 코드를 작성해보세요.
- 4) 각 고객이 주문한 제품명과 수량을 출력한다

```
class Item { //제품
    private String name; // 제품명
    private double price; //제품 가격
    private int stockQuantity; //재고량
    void reduceStock(int quantity) {
        //판매 시 재고 감소 메소드
    }
    void increaseStock(int quantity) {
    }
    public void show() {
        // name = **, price=** 등으로 출력
    }
    public String toString() {
        return "name = " + name;
    }
}
```

# ExamB1. 온라인 상점 관리 시스템 구현

```
class Customer {  
    private String cname;  
    private String city;  
    private int age;  
    public void show() {  
        // name = ***, city = *** 등으로 출력  
    }  
}
```

```
class Order {  
    private Customer customer; //고객명  
    private Item []items; //주문 제품들  
    private int []quantities; //주문 제품 수량들  
    private String []orderDates; //주문일자들  
    private int count = 0; //배열 인덱스  
    void addItem(Item item, int orderNumber) {}  
    double calculateTotal() { return 0.0; }  
    void printOrderSummary() {}  
}
```

```
package exam;
```

```
public class ExamB1 {
```

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

```
        // 아이템 생성
```

```
        Item laptop = new Item("노트북", 1200.00, 10);
```

```
        Item tshirt = new Item("티셔츠", 20.00, 50);
```

```
        // 고객 생성
```

```
        Customer boy = new Customer("홍길동", "부산", 21);
```

```
        Customer girl = new Customer("계백", "양산", 22);
```

```
        // 주문 생성
```

```
        Order order1 = new Order(boy);
```

```
        order1.addItem(laptop, 1);
```

```
        order1.addItem(tshirt, 2);
```

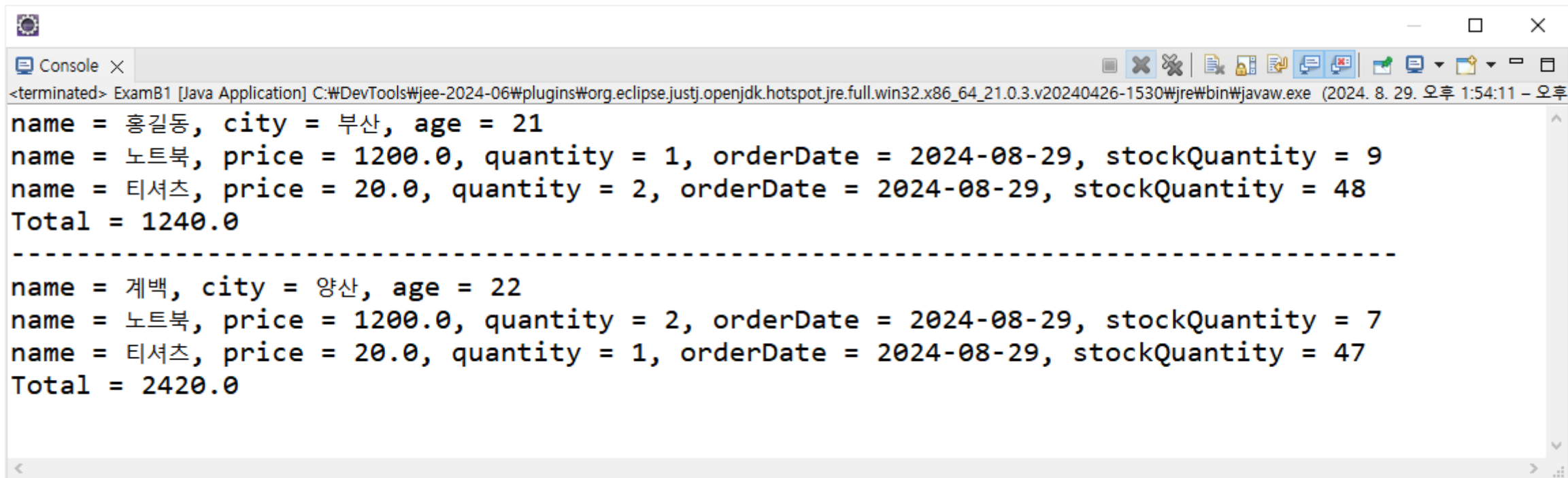
```
    }
```

```
}
```

```
}
```

# ExamB1. 온라인 상점 관리 시스템 구현

- 실행 예



The screenshot shows a Java console window titled "ExamB1 [Java Application]". The output displays two sets of data for a shopping system. The first set is for a user named 홍길동 (Hong Gildong) from 부산 (Busan), age 21, with a total order of 1240.0. The second set is for a user named 계백 (Gyeobak) from 양산 (Yangsan), age 22, with a total order of 2420.0. Each set lists items like 노트북 (laptop) and 티셔츠 (t-shirt) with their respective prices and quantities.

```
<terminated> ExamB1 [Java Application] C:\DevTools\jee-2024-06\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre\bin\javaw.exe (2024. 8. 29. 오후 1:54:11 - 오후  
name = 홍길동, city = 부산, age = 21  
name = 노트북, price = 1200.0, quantity = 1, orderDate = 2024-08-29, stockQuantity = 9  
name = 티셔츠, price = 20.0, quantity = 2, orderDate = 2024-08-29, stockQuantity = 48  
Total = 1240.0  
-----  
name = 계백, city = 양산, age = 22  
name = 노트북, price = 1200.0, quantity = 2, orderDate = 2024-08-29, stockQuantity = 7  
name = 티셔츠, price = 20.0, quantity = 1, orderDate = 2024-08-29, stockQuantity = 47  
Total = 2420.0
```

## ExamB2. 온라인 상점 관리 시스템 구현

- 내용

- ExamB1을 복사해서 ExamB2 생성
- Electronics와 Clothing 클래스를 Item 클래스로부터 상속받아 작성하세요.
- 각 서브 클래스에 고유한 속성(전자 제품의 경우 보증 기간, 의류의 경우 크기 및 색상)을 추가하세요.
- 상속된 메소드를 사용하여 각각의 객체를 생성하고 관리하세요.

# ExamB2. 온라인 상점 관리 시스템 구현

```
class Electronics extends Item {
    private int warranty; //제품 보증 기간
    //생성자를 Item 클래스를 생성자를 사용하여 구현, this를 사용
    @Override
    public void show() {} // name = ***, price=*** 등으로 출력
    public String toString() {
        return super.toString();
    }
}

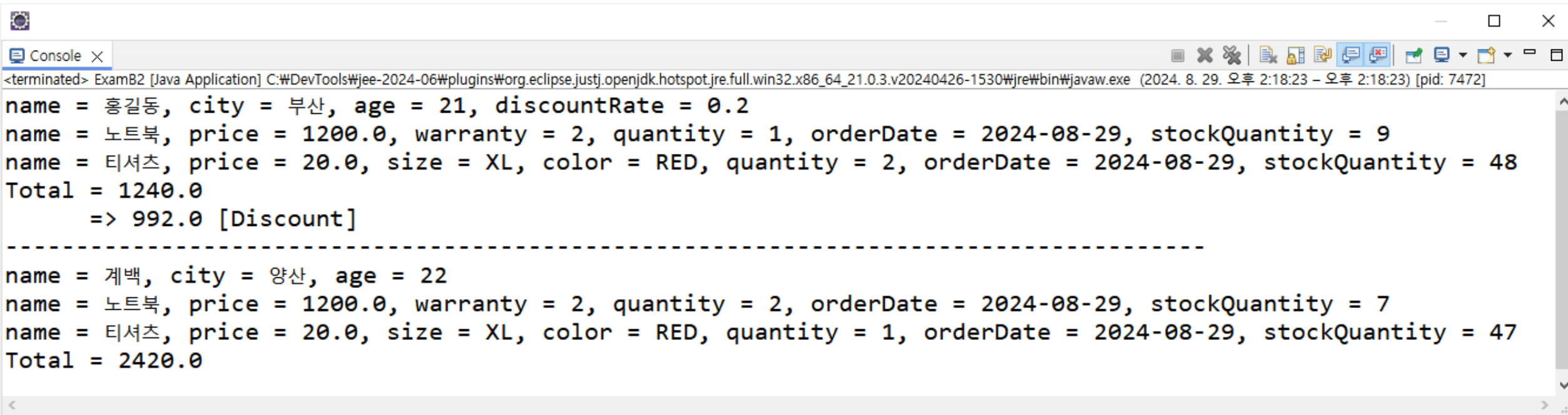
class Clothing extends Item {
    private String size;
    private String color;
    //생성자를 Item 클래스를 생성자를 사용하여 구현, this를 사용
    @Override
    public void show() {} // name = ***, price=*** 등으로 출력
    public String toString() {
        return super.toString();
    }
}
```

```
class PremiumCustomer extends Customer {
    private float discountRate;
    //생성자를 Customer 클래스 생성자를 사용하여 구현, this를 사용
    @Override
    public void show() {} // name = ***, price=*** 등으로 출력
    public String toString() {
        return super.toString();
    }
}

public class ExamB2 {
    public static void main(String[] args) {
        // 아이템 생성
        Item laptop = new Electronics("노트북", 1200.00, 10, 2);
        Item tshirt = new Clothing("티셔츠", 20.00, 50, "XL", "RED");
        // 고객 생성
        Customer boy = new PremiumCustomer("홍길동", "부산", 21, 0.2f);
        Customer girl = new Customer("계백", "양산", 22);
        // 주문 생성 (홍길동)
        Order order1 = new Order(boy);
    }
}
```

# ExamB2. 온라인 상점 관리 시스템 구현

- 실행 예



```
<terminated> ExamB2 [Java Application] C:\DevTools\jee-2024-06\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre\bin\javaw.exe (2024. 8. 29. 오후 2:18:23 - 오후 2:18:23) [pid: 7472]
name = 홍길동, city = 부산, age = 21, discountRate = 0.2
name = 노트북, price = 1200.0, warranty = 2, quantity = 1, orderDate = 2024-08-29, stockQuantity = 9
name = 티셔츠, price = 20.0, size = XL, color = RED, quantity = 2, orderDate = 2024-08-29, stockQuantity = 48
Total = 1240.0
=> 992.0 [Discount]
-----
name = 계백, city = 양산, age = 22
name = 노트북, price = 1200.0, warranty = 2, quantity = 2, orderDate = 2024-08-29, stockQuantity = 7
name = 티셔츠, price = 20.0, size = XL, color = RED, quantity = 1, orderDate = 2024-08-29, stockQuantity = 47
Total = 2420.0
```

# ExamB3. 온라인 상점 관리 시스템 구현

## ◦ 내용

### 1) main() 함수에서 객체 생성

```
public static void main(String[] args) {  
    // 고객 생성  
    // 아이템 생성  
    // 할인 정책 결정  
    // 일반 고객의 주문 생성  
    // 프리미엄 고객의 주문 생성  
}
```

### 2) 인터페이스 정의

- Discountable 인터페이스를 정의하고, 이를 구현하는 SeasonalDiscount 클래스를 작성하세요.
- Customer 추상 클래스를 정의하고, 이를 상속받는 RegularCustomer와 PremiumCustomer 클래스를 작성하세요.



# ExamB3. 온라인 상점 관리 시스템 구현

```
interface Discountable {
    double getDiscountedPrice(double price);
}

class SeasonalDiscount implements Discountable {
    private double discountRate;

    // Discountable 인터페이스 메서드 구현
    @Override public double getDiscountedPrice(double price) {
        return price * (1 - discountRate);
    }
}

abstract class Customer {
    private String name;

    abstract double applyDiscount(double totalAmount);
}

class RegularCustomer extends Customer {
    double applyDiscount(double totalAmount) {
        return totalAmount;
    }
}

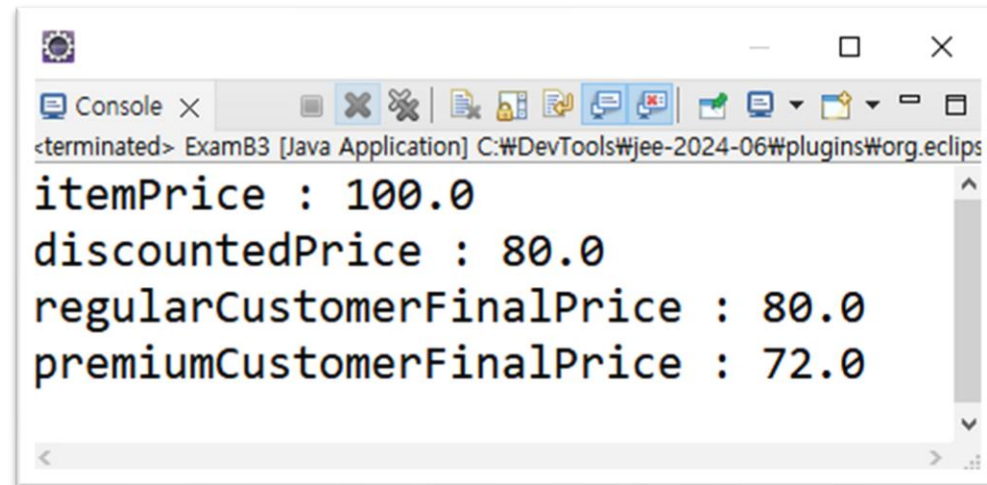
class PremiumCustomer extends Customer {
    static final double DISCOUNT_RATE = 0.1;
```

```
double applyDiscount(double totalAmount) {
    return totalAmount * (1 - DISCOUNT_RATE);
}

public class ExamB3 {
    public static void main(String[] args) {
        // 고객 생성
        Customer regularCustomer = new RegularCustomer("홍길동");
        Customer premiumCustomer = new PremiumCustomer("김유신");
        double itemPrice = 100.0; // 아이템 생성 (가격은 임의로 설정)
        // 할인 정책 결정 (계절 할인 예시)
        Discountable seasonalDiscount = new SeasonalDiscount(0.2);
        // 20% 할인
        double discountedPrice =
            seasonalDiscount.getDiscountedPrice(itemPrice);
        // 일반 고객의 주문 생성
        double regularCustomerFinalPrice =
            regularCustomer.applyDiscount(discountedPrice);
        // 프리미엄 고객의 주문 생성
        double premiumCustomerFinalPrice =
            premiumCustomer.applyDiscount(discountedPrice);
    }
}
```

# ExamB3. 온라인 상점 관리 시스템 구현

- 실행 예



The screenshot shows a console window from the Eclipse IDE. The title bar indicates the application is terminated. The output text is as follows:

```
<terminated> ExamB3 [Java Application] C:\DevTools\jee-2024-06\plugins\org.eclipse  
itemPrice : 100.0  
discountedPrice : 80.0  
regularCustomerFinalPrice : 80.0  
premiumCustomerFinalPrice : 72.0
```

## ExamB4. 온라인 상점 관리 시스템 구현

---

- 내용

- 1) Item 타입의 배열에 Electronics와 Clothing 객체를 추가하세요.
- 2) 반복문을 통해 배열의 모든 아이템의 이름과 가격을 출력하세요.

# ExamB4. 온라인 상점 관리 시스템 구현

```
//Item 추상 클래스
abstract class Item {
    private String name; // 제품명
    private double price; // 제품 가격
    private int stockQuantity; // 재고량
}

class Order {
    private Customer customer; // 고객명
    private Item [] items; // 주문 제품들
    private int [] quantities; // 주문 제품 수량들
    private String [] orderDates; // 주문일자들
    private static int count;
}

public class ExamB4 {
    public static void main() {
        // Item 타입의 배열 생성
        Item[] items = new Item[4];
```

```
// 배열에 Electronics와 Clothing 객체 추가
items[0] = new Electronics("Laptop", 1500.00, 24);
items[1] = new Clothing("T-Shirt", 19.99, );
items[2] = new Electronics("Smartphone", 800.00, 12);
items[3] = new Clothing("Jeans", 49.99, );
// 모든 아이템의 이름과 가격 출력
for (Item item : items) {
    // 동적 바인딩에 의해 각 클래스의 show() 메서드가 호출
    item.show();
}
// 고객 생성
Customer regularCustomer = new RegularCustomer("Alice");
Customer premiumCustomer = new PremiumCustomer("Bob");
// 주문 생성 및 계산 (RegularCustomer)
Order regularOrder = new Order(regularCustomer, items);
regularOrder.printOrderSummary();
// 고객별 주문 제품, 가격, 주문일자, 할인을
// 테이블 형태로 출력하는 정적 메소드 코드 구현
}
```

# ExamB4. 온라인 상점 관리 시스템 구현

- 내용

3) polymorphism을 활용하여, Order 클래스에서 고객의 타입에 따라 적절한 할인이 적용되도록 하세요.

Order 클래스를 작성해 고객의 주문을 관리하고, 각 고객의 할인 혜택이 반영된 총액을 계산하세요.

```
class Order {  
    private Customer customer; //고객명  
    private Item []items; //주문 제품들  
    private int []quantities;//주문 제품 수량들  
    private String []orderDates;//주문일자들  
    private static int count;  
    Discountable discountPolicy;  
    void addItem(Item item, int quantity){  
        //주문량이 재고량보다 적으면  
        items[i] = item;  
        item.reduceStock(quantity);  
    }  
}
```

```
double calculateTotal() {  
    //주문 제품들의 할인액을 모두 계산  
    return customer.applyDiscount(total);  
}  
void printOrderSummary() {  
    //고객 이름 출력  
    //고객이 주문한 제품이름과 가격을 출력  
    // 할인 총액을 출력  
}  
}
```

# ExamB4. 온라인 상점 관리 시스템 구현

- 실행 예

```
name = Laptop, price = 1500.0, warranty = 2
name = T-Shirt, price = 19.99, size = XL, color = RED
name = Smartphone, price = 800.0, warranty = 2
name = Jeans, price = 49.99, size = XL, color = BLUE
-----
name = Alice, city = NY, age = 20
name = Laptop, price = 1500.0, warranty = 2, quantity = 10, orderDate = 2024-08-29, stockQuantity = 10
name = T-Shirt, price = 19.99, size = XL, color = RED, quantity = 50, orderDate = 2024-08-29, stockQuantity = 50
name = Smartphone, price = 800.0, warranty = 2, quantity = 12, orderDate = 2024-08-29, stockQuantity = 12
name = Jeans, price = 49.99, size = XL, color = BLUE, quantity = 60, orderDate = 2024-08-29, stockQuantity = 60
Total = 28598.9
-----
name = Bob, city = LA, age = 30, discountRate = 0.2
name = Laptop, price = 1500.0, warranty = 2, quantity = 10, orderDate = 2024-08-29, stockQuantity = 10
name = T-Shirt, price = 19.99, size = XL, color = RED, quantity = 50, orderDate = 2024-08-29, stockQuantity = 50
name = Smartphone, price = 800.0, warranty = 2, quantity = 12, orderDate = 2024-08-29, stockQuantity = 12
name = Jeans, price = 49.99, size = XL, color = BLUE, quantity = 60, orderDate = 2024-08-29, stockQuantity = 60
Total = 28598.9
=> 22879.1 [Discount]
```

# ExamC1. 도서 관리 시스템

---

- 목표

- Book 클래스 생성 및 기본 컬렉션 사용

- 설계

- Book 클래스: 제목, 저자, 출판 연도, ISBN 등을 속성으로 가짐
- Library 클래스: 도서 목록을 관리하는 기능 (책 추가, 책 목록 출력)

- 과제

- Book 클래스를 작성하고 도서 정보를 담은 객체를 생성하세요.
- Library 클래스를 작성해 ArrayList를 사용하여 도서를 관리하고, 도서 목록을 출력하세요.

# ExamC1. 도서 관리 시스템

//Book 클래스

```
public class Book {  
    private String title;  
    private String author;  
    private int publicationYear;  
    private String isbn;  
  
    @Override  
    public String toString() {  
        // title="****", author= *** 등으로 출력  
        return null;  
    }  
}
```

//Library 클래스

```
public class Library {  
    private Book[] books;  
    private int top;  
    public void addBook(Book book) {}  
    public void printBooks() {}  
    public void sortBooksByTitle(){  
        //String의 compareTo() 사용  
        Arrays.sort(books, 0, count,  
            (b1,b2)->b1.getTitle().compareTo(b2.getTitle()));  
    }  
    public Book searchBookByTitle(String title) {  
        //equals() 사용  
    }  
}
```



# ExamC1. 도서 관리 시스템

```
public static void main(String[] args) {  
    Library library = new Library(5);  
    // 5개의 Book 객체 초기화  
    Book book1 = new Book("자바", "강감찬", 1995, "1234567890");  
    Book book2 = new Book("파이썬", "이순신", 2008, "1234567891");  
    Book book3 = new Book("자바스크립트", "을지문덕", 2008, "1234567892");  
    Book book4 = new Book("자료구조", "연개소문", 1994, "1234567893");  
    Book book5 = new Book("리액트", "김춘추", 1999, "1234567894");  
    // 책 추가  
    library.addBook(book1); library.addBook(book2); library.addBook(book3);  
    library.addBook(book4); library.addBook(book5);  
  
    library.printBooks();           // 도서 목록 출력  
    library.sortBooksByTitle();     // 도서 목록 정렬  
    library.printBooks();           // 정렬된 도서 목록 출력  
  
    // 특정 제목으로 도서 검색  
    String searchText = "자바";  
    Book foundBook = library.searchBookByTitle(searchText);  
}
```

# ExamC1. 도서 관리 시스템

- 실행 예

```
Console X
<terminated> ExamC1 [Java Application] C:\DevTools\jee-2024-06\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre\bin

title=자바,          author=강감찬,      publicationYear=1995,    isbn=1234567890
title=파이썬,        author=이순신,        publicationYear=2008,    isbn=1234567891
title=자바스크립트,  author=울지문덕,      publicationYear=2008,    isbn=1234567892
title=자료구조,      author=연개소문,      publicationYear=1994,    isbn=1234567893
title=리액트,        author=김춘추,        publicationYear=1999,    isbn=1234567894
-----
title=리액트,        author=김춘추,        publicationYear=1999,    isbn=1234567894
title=자료구조,      author=연개소문,      publicationYear=1994,    isbn=1234567893
title=자바,          author=강감찬,        publicationYear=1995,    isbn=1234567890
title=자바스크립트,  author=울지문덕,      publicationYear=2008,    isbn=1234567892
title=파이썬,        author=이순신,        publicationYear=2008,    isbn=1234567891
-----
searchBookByTitle by 자바
title=자바,          author=강감찬,        publicationYear=1995,    isbn=1234567890
```

# ExamC2. 도서 관리 시스템

## ○ 과제

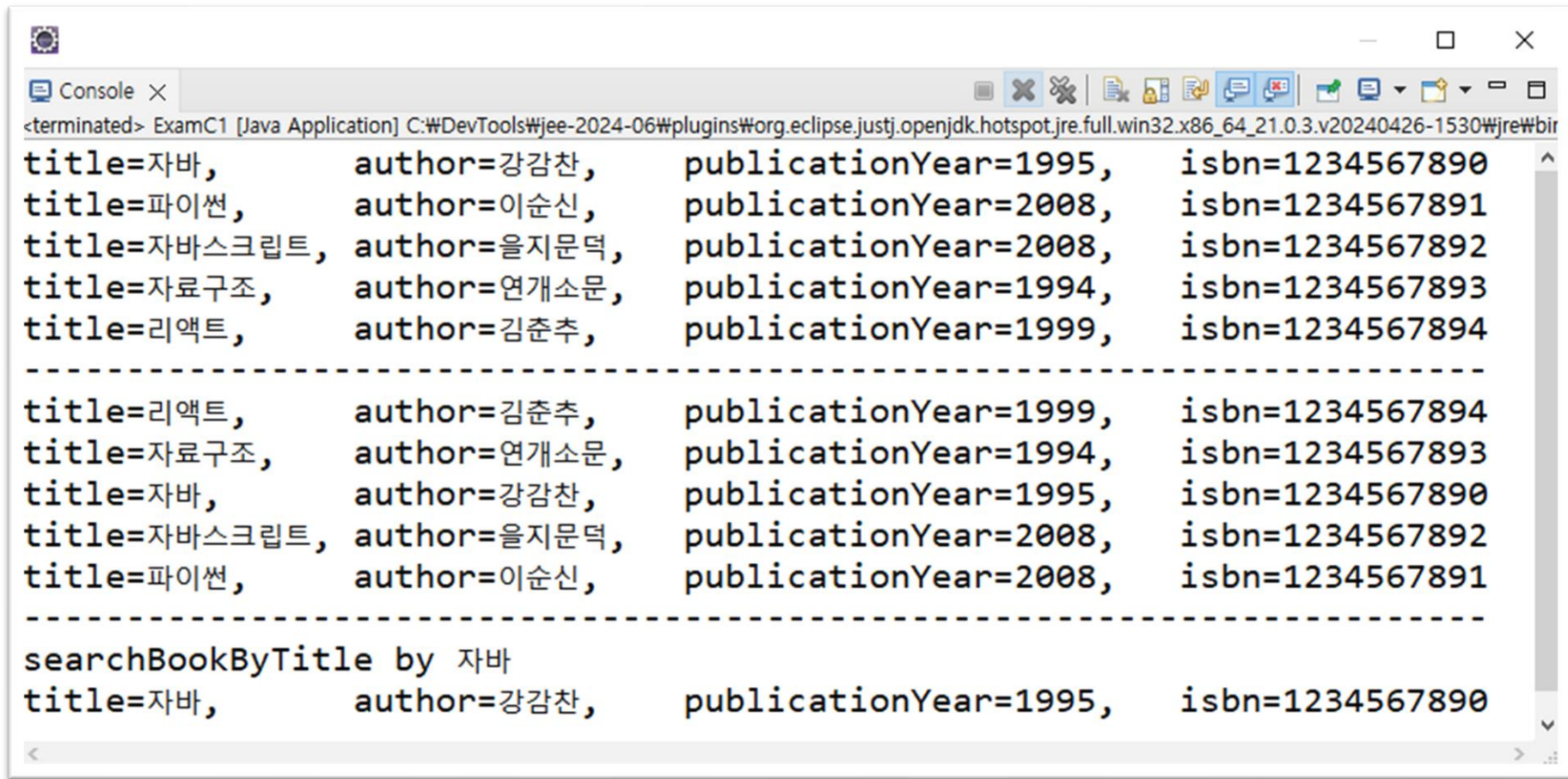
- 1) class Book implements Comparable {}
- 2) compareTo()의 구현
- 3) class Library는 배열 대신에 ArrayList를 사용

```
class Book implements Comparable<Book> {  
    @Override  
    public int compareTo(Object o){  
        Book other = (Book) o;  
        return this.title.compareTo(other.title);  
    }  
}
```

```
class Library {  
    private ArrayList<Book> books;  
    public Library() {  
        books = new ArrayList<>();  
    }  
    // 책 추가  
    public void addBook(Book book) {  
    }  
    // 책 제목 기준으로 정렬  
    public void sortBooksByTitle() {  
    }  
    // 특정 제목의 책 검색  
    public Book searchBookByTitle(String title) {  
    }  
}
```

# ExamC2. 도서 관리 시스템

- 실행 예



```
<terminated> ExamC1 [Java Application] C:\DevTools\jee-2024-06\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre\bin
title=자바,          author=강감찬,      publicationYear=1995,  isbn=1234567890
title=파이썬,        author=이순신,      publicationYear=2008,  isbn=1234567891
title=자바스크립트,  author=울지문덕,    publicationYear=2008,  isbn=1234567892
title=자료구조,      author=연개소문,    publicationYear=1994,  isbn=1234567893
title=리엑트,        author=김춘추,      publicationYear=1999,  isbn=1234567894
-----
title=리엑트,        author=김춘추,      publicationYear=1999,  isbn=1234567894
title=자료구조,      author=연개소문,    publicationYear=1994,  isbn=1234567893
title=자바,          author=강감찬,      publicationYear=1995,  isbn=1234567890
title=자바스크립트,  author=울지문덕,    publicationYear=2008,  isbn=1234567892
title=파이썬,        author=이순신,      publicationYear=2008,  isbn=1234567891
-----
searchBookByTitle by 자바
title=자바,          author=강감찬,      publicationYear=1995,  isbn=1234567890
```

# ExamC3. 도서 관리 시스템

## ○ 과제

- 예외 처리를 통해 리스트의 용량 초과(Overflow)와 빈 리스트에서의 삭제 시 발생할 수 있는 오류(Underflow)를 처리하는 기능을 추가

```
//Overflow 예외 클래스
class OverflowException extends RuntimeException {
    public OverflowException(String message) {
        super(message);
    }
}

//Underflow 예외 클래스
class UnderflowException extends RuntimeException {
    public UnderflowException(String message) {
        super(message);
    }
}
```

# ExamC3. 도서 관리 시스템

```
class Library {
    private int capacity;
    public Library(int capacity) {
        this.capacity = capacity;
        books = new ArrayList<>(capacity);
    }
    //책 추가
    public void addBook(Book book) {
        if (books.size() >= capacity) {
            //예외 발생
        }
        books.add(book);
    }
    //책 삭제
    public Book removeBook() {
        if (books.size() == 0) {
            //예외 발생
        }
        books.add(book);
    }
}
```

```
public class ExamC3 {
    public static void main(String[] args) {
        Library library = new Library(5);
        // 5개의 Book 객체 초기화
        Book book1 = new Book("자바", "강감찬", 1995, "1234567890");
        Book book2 = new Book("파이썬", "이순신", 2008, "1234567891");
        Book book3 = new Book("자바스크립트", "을지문덕", 2008, "1234567892");
        Book book4 = new Book("자료구조", "연개소문", 1994, "1234567893");
        Book book5 = new Book("리액트", "김춘추", 1999, "1234567894");
        // 책 추가
        library.addBook(book1); library.addBook(book2); library.addBook(book3);
        library.addBook(book4); library.addBook(book5);

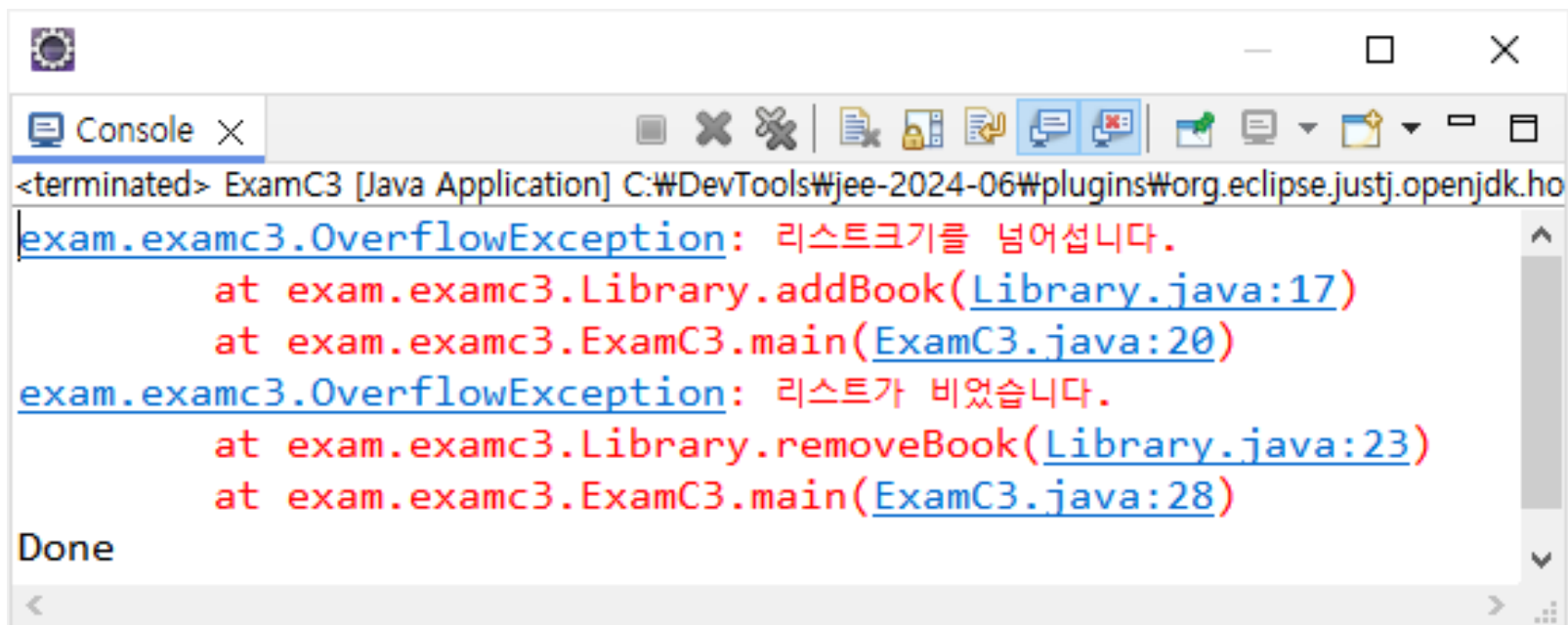
        try {
            library.addBook(new Book("스프링부트", "홍길동", 1999, "1234567895"));
        } catch (Exception e) {
            e.printStackTrace();
        }

        try {
            for (int i = library.getBooks().size() - 1 ; 0 <= 0 ; i--)
                library.removeBook();
            library.removeBook();
        } catch (Exception e) {
            e.printStackTrace();
        }
        System.out.println("Done");
    }
}
```



# ExamC3. 도서 관리 시스템

- 실행 예



```
<terminated> ExamC3 [Java Application] C:\DevTools\jee-2024-06\plugins\org.eclipse.justj.openjdk.hotspot.jre.full\jre\bin\java.exe
exam.examc3.OverflowException: 리스트크기를 넘어섭니다.
    at exam.examc3.Library.addBook(Library.java:17)
    at exam.examc3.ExamC3.main(ExamC3.java:20)
exam.examc3.OverflowException: 리스트가 비었습니다.
    at exam.examc3.Library.removeBook(Library.java:23)
    at exam.examc3.ExamC3.main(ExamC3.java:28)
Done
```

# ExamC4. 도서 관리 시스템

- 과제

- 클래스 library의 타입을 generic으로 구현

```
interface MediaBook {  
    String getMediaType();  
}  
  
class Book implements MediaBook, Comparable<MediaBook> {  
    private String title;  
    private String author;  
    private int publicationYear;  
    private String isbn;  
    @Override  
    public int compareTo(MediaBook mbook) {  
    }  
}
```



# ExamC4. 도서 관리 시스템

```
class CDBook implements MediaBook, Comparable<MediaBook> {
    private String title;
    private String artist;
    private String catalogNumber;

    @Override
    public int compareTo(MediaBook mbook) {
    }
}

class USBBook implements MediaBook, Comparable<MediaBook> {
    private String title;
    private int capacity;
    private String serialNumber;
}
```

```
class Library<T extends MediaBook> {
    private ArrayList<T> items;
    private int capacity;

    // 항목 추가
    public void addItem(T item) {
    }

    // 항목 삭제
    public void removeItem(T item) {
    }
}
```

# ExamC5. 도서 관리 시스템

## ○ 과제

- 람다식을 사용하여 Book은 출판 연도에 따라, CD는 카탈로그 넘버에 따라, USB는 용량에 따라 정렬하는 기능을 구현
  - 1) 람다식 및 스트림 API를 사용한 정렬
  - 2) 스트림 API를 활용하여 출판 연도에 따라 도서를 필터링하고, 제목 또는 저자 이름에 따라 도서를 정렬합니다.
  - 3) filterByYear 메서드에서 람다식을 사용해 Book 객체의 출판 연도를 기준으로 필터링합니다.
  - 4) sortByTitleOrAuthor 메서드에서 스트림 API와 람다식을 사용해 제목 또는 저자 이름에 따라 아이템들을 정렬합니다.

# ExamC5. 도서 관리 시스템

```
interface MediaBook {
    String getMediaType();
}

class Library<T extends MediaBook> {
    private ArrayList<T> items;
    private int capacity;
    public void sortItems() {
        items.sort((a, b) -> {
            if (a instanceof Book && b instanceof Book) {
                return ((Book) a).getPublicationYear() - ((Book) b).getPublicationYear();
            } else if (a instanceof CD && b instanceof CD) {
                return ((CD) a).getCatalogNumber().compareTo(((CD) b).getCatalogNumber());
            } else if (a instanceof USB && b instanceof USB) {
                return Integer.compare(((USB) a).getCapacityGB(), ((USB) b).getCapacityGB());
            }
            return 0;
        });
    }
}
```

# ExamC5. 도서 관리 시스템

```
public List<T> filterByYear(int year) { // 출판 연도에 따른 필터링
    return items.stream()
        .filter(item -> item instanceof Book && ((Book) item).getPublicationYear() == year)
        .collect(Collectors.toList());
}

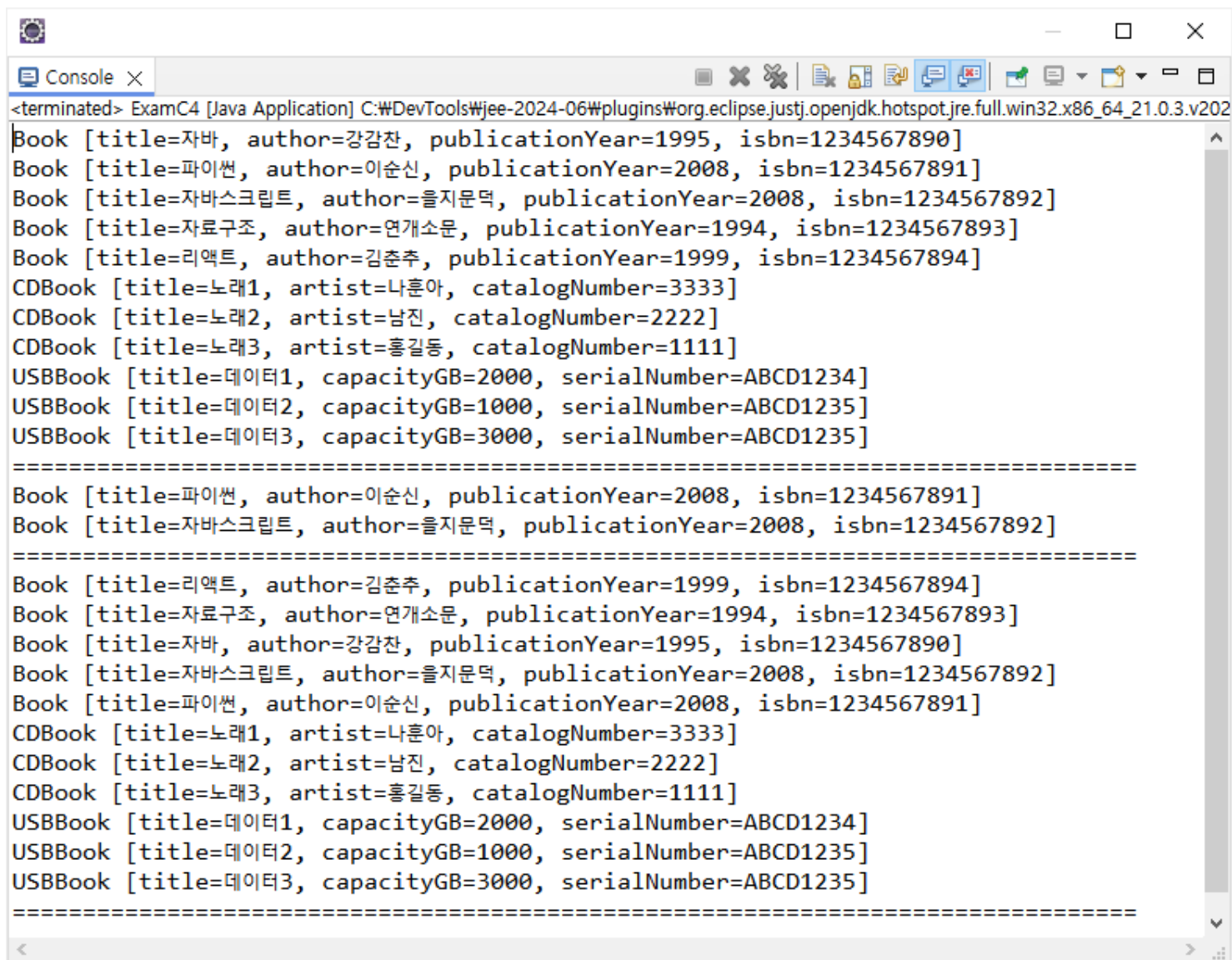
public void sortByTitleOrAuthor() { // 제목 또는 저자 이름에 따른 정렬
    items = items.stream()
        .sorted((a, b) -> {
            if (a instanceof Book && b instanceof Book) {
                return ((Book) a).getTitle().compareTo(((Book) b).getTitle());
            } else if (a instanceof CD && b instanceof CD) {
                return ((CD) a).getArtist().compareTo(((CD) b).getArtist());
            } return 0;
        })
        .collect(Collectors.toList());
}
}
```

# ExamC5. 도서 관리 시스템

```
public static void main(String[] args) {  
    // 출판 연도에 따른 필터링 (예: 2008년도 출판)  
    List<MediaBook> filteredBooks = library.filterByYear(2008);  
    filteredBooks.forEach(System.out::println);  
  
    // 제목 또는 저자 이름에 따른 정렬  
    library.sortByTitleOrAuthor();  
    library.printItems();  
}  
}
```

# ExamC5. 도서 관리 시스템

## ● 실행 예



```
<terminated> ExamC4 [Java Application] C:\DevTools\Wjee-2024-06\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v202
Book [title=자바, author=강감찬, publicationYear=1995, isbn=1234567890]
Book [title=파이썬, author=이순신, publicationYear=2008, isbn=1234567891]
Book [title=자바스크립트, author=을지문덕, publicationYear=2008, isbn=1234567892]
Book [title=자료구조, author=연개소문, publicationYear=1994, isbn=1234567893]
Book [title=리액트, author=김춘추, publicationYear=1999, isbn=1234567894]
CDBook [title=노래1, artist=나훈아, catalogNumber=3333]
CDBook [title=노래2, artist=남진, catalogNumber=2222]
CDBook [title=노래3, artist=홍길동, catalogNumber=1111]
USBBook [title=데이터1, capacityGB=2000, serialNumber=ABCD1234]
USBBook [title=데이터2, capacityGB=1000, serialNumber=ABCD1235]
USBBook [title=데이터3, capacityGB=3000, serialNumber=ABCD1235]
=====
Book [title=파이썬, author=이순신, publicationYear=2008, isbn=1234567891]
Book [title=자바스크립트, author=을지문덕, publicationYear=2008, isbn=1234567892]
=====
Book [title=리액트, author=김춘추, publicationYear=1999, isbn=1234567894]
Book [title=자료구조, author=연개소문, publicationYear=1994, isbn=1234567893]
Book [title=자바, author=강감찬, publicationYear=1995, isbn=1234567890]
Book [title=자바스크립트, author=을지문덕, publicationYear=2008, isbn=1234567892]
Book [title=파이썬, author=이순신, publicationYear=2008, isbn=1234567891]
CDBook [title=노래1, artist=나훈아, catalogNumber=3333]
CDBook [title=노래2, artist=남진, catalogNumber=2222]
CDBook [title=노래3, artist=홍길동, catalogNumber=1111]
USBBook [title=데이터1, capacityGB=2000, serialNumber=ABCD1234]
USBBook [title=데이터2, capacityGB=1000, serialNumber=ABCD1235]
USBBook [title=데이터3, capacityGB=3000, serialNumber=ABCD1235]
=====
```