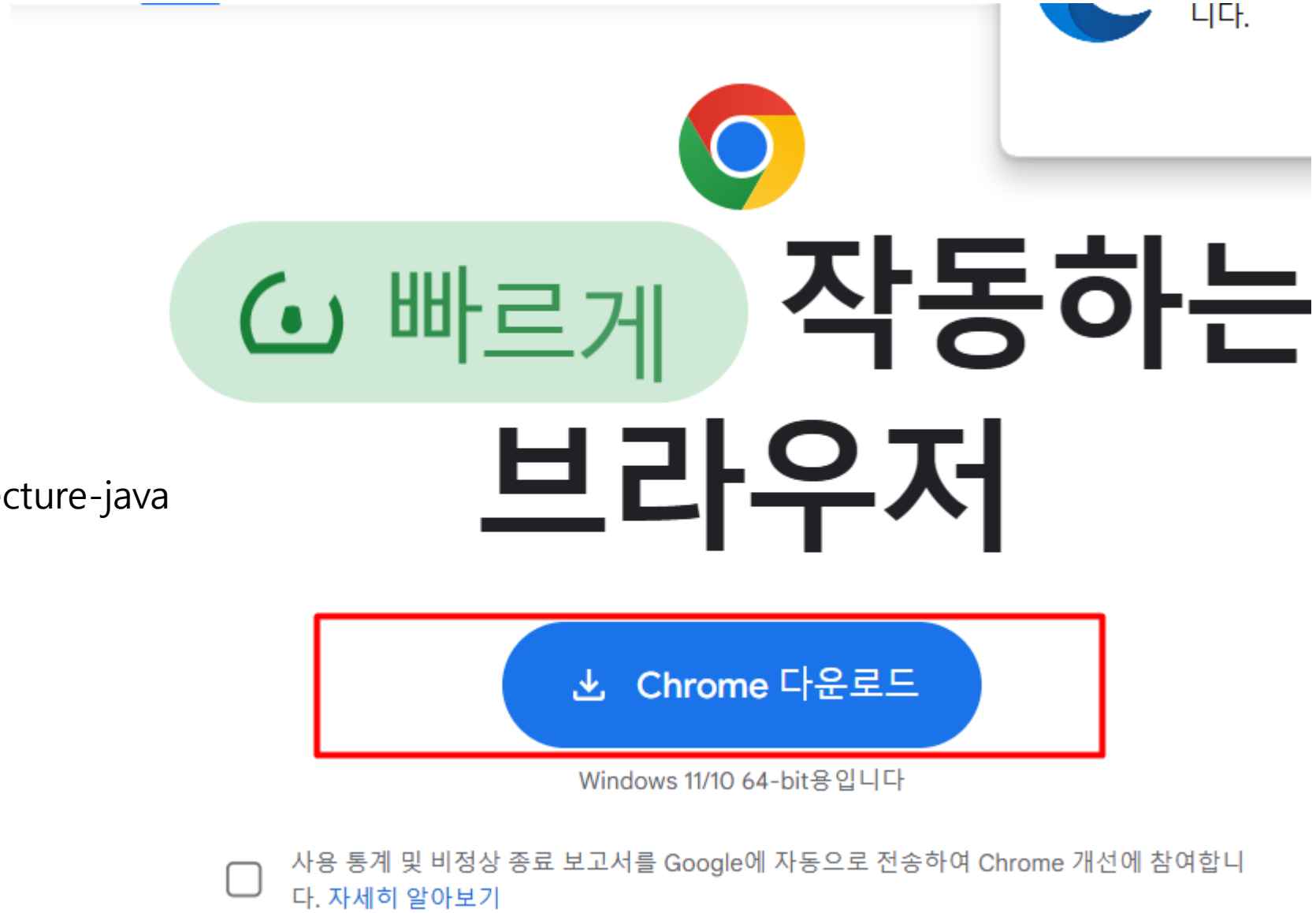
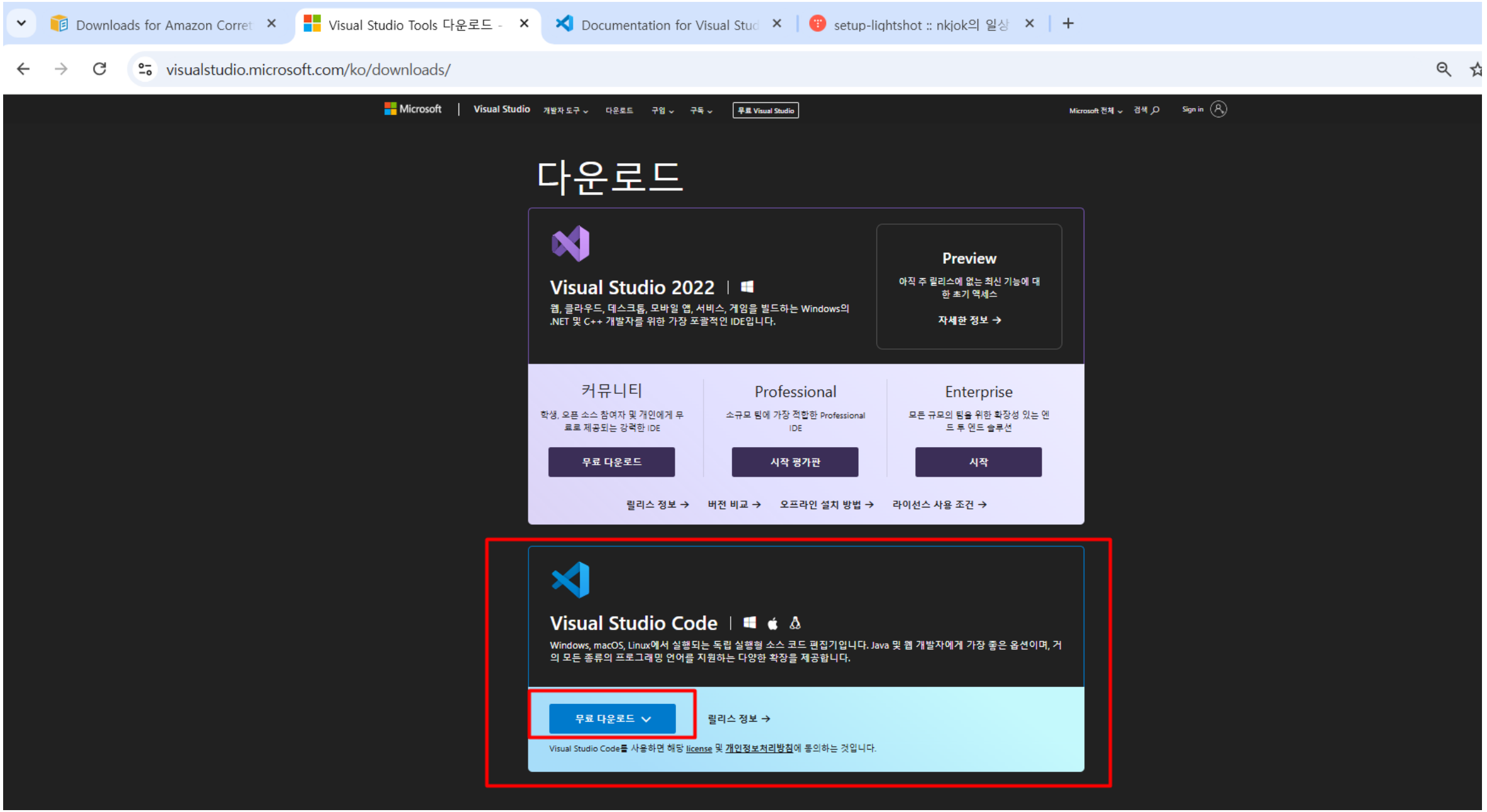
크롬설치

선생님 깃허브

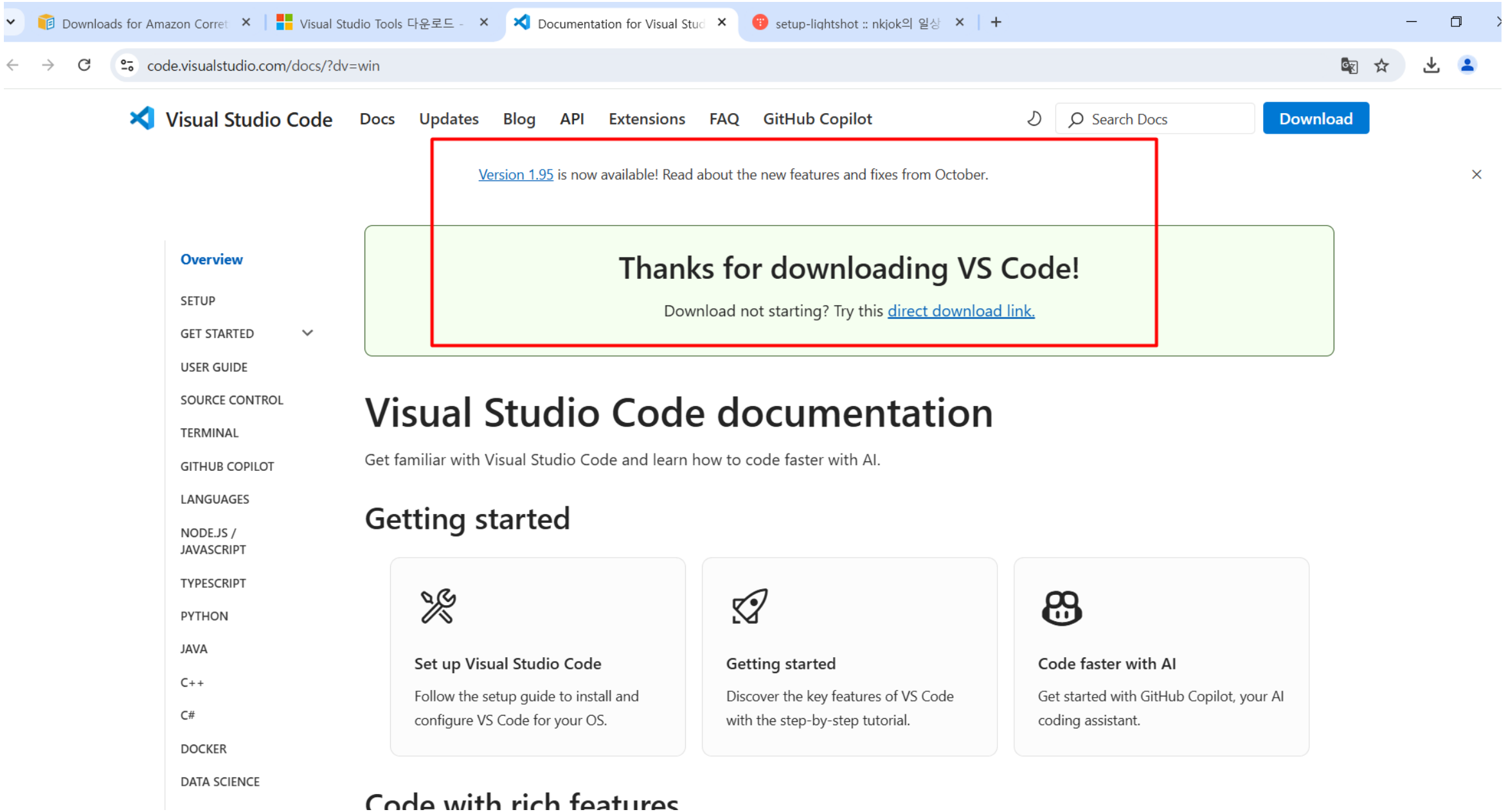
https://github.com/kimsinok/lecture-java



https://visualstudio.microsoft.com/ko/downloads/

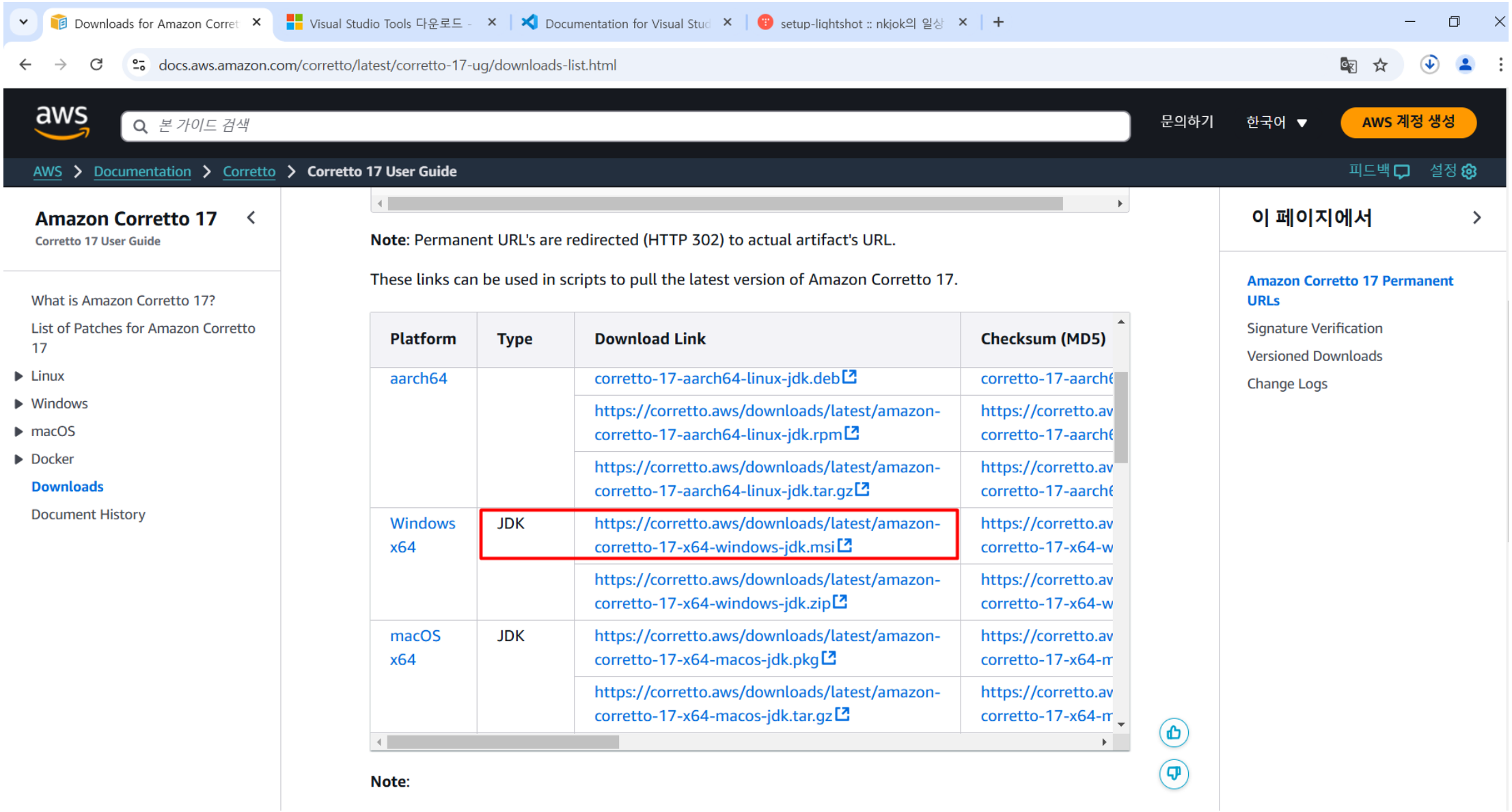


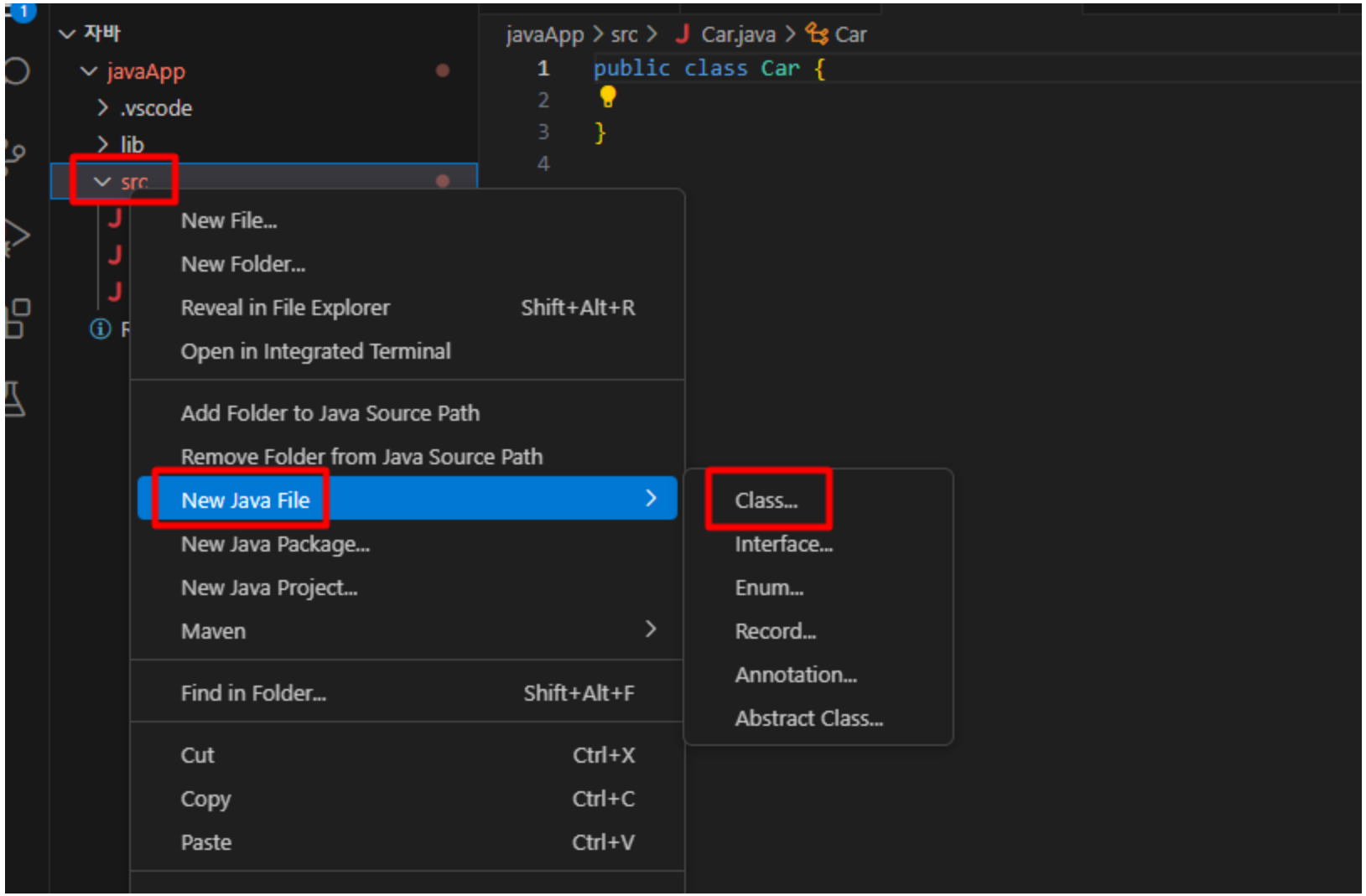
https://code.visualstudio.com/docs/?dv=win VSCodeUserSetup-x64-1.95.3



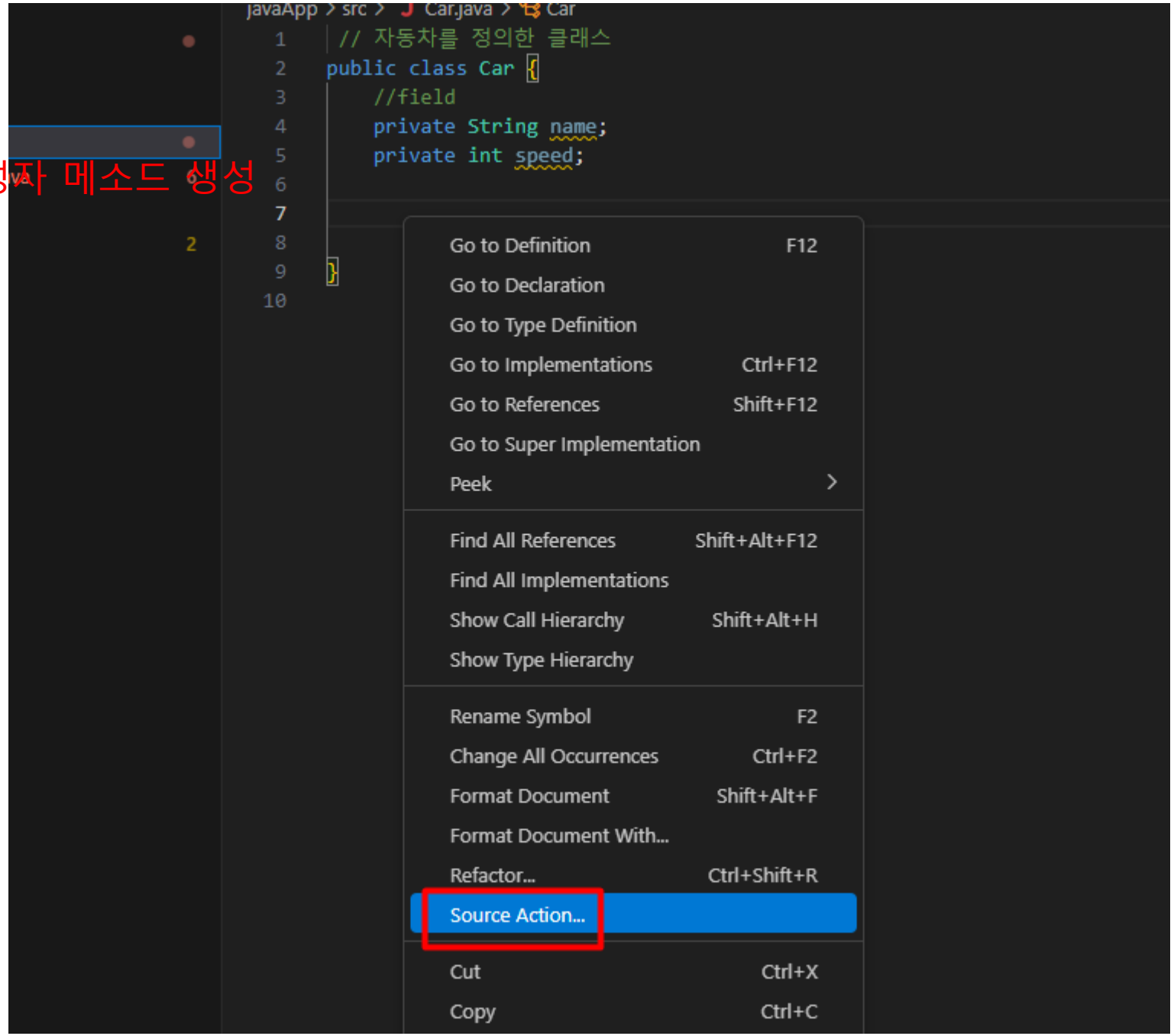
https://docs.aws.amazon.com/corretto/latest/corretto-17-ug/downloads-list.html

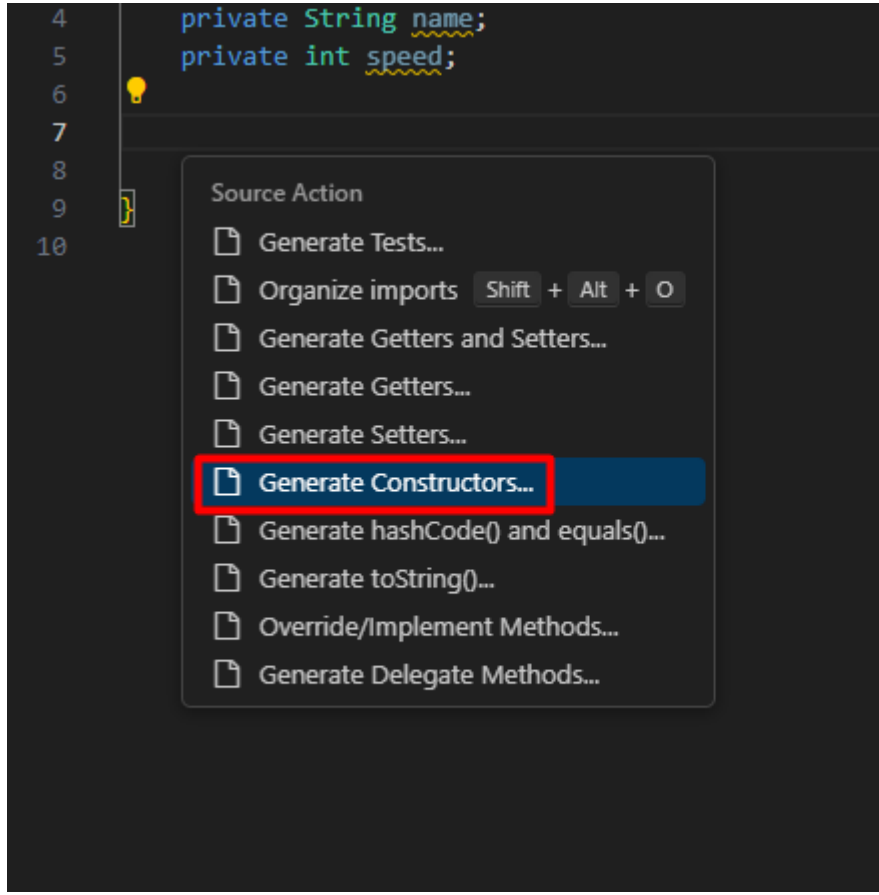
amazon-corretto-17.0.13.11.1-windows-x64

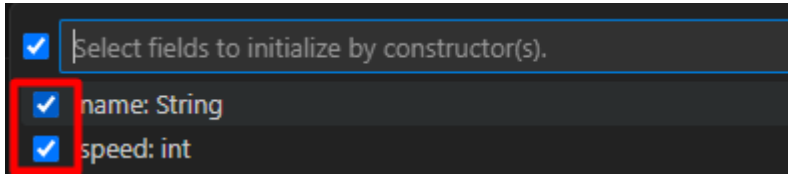



자바
javaApp
.vscode
lib
src
javaApp > src > Car.java > Car
1 public class Car {
2
3 }
4
New File...
New Folder...
Reveal in File Explorer Shift+Alt+R
Open in Integrated Terminal
Add Folder to Java Source Path
Remove Folder from Java Source Path
New Java File
New Java Package...
New Java Project...
Maven
Find in Folder... Shift+Alt+F
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Class...
Interface...
Enum...
Record...
Annotation...
Abstract Class...

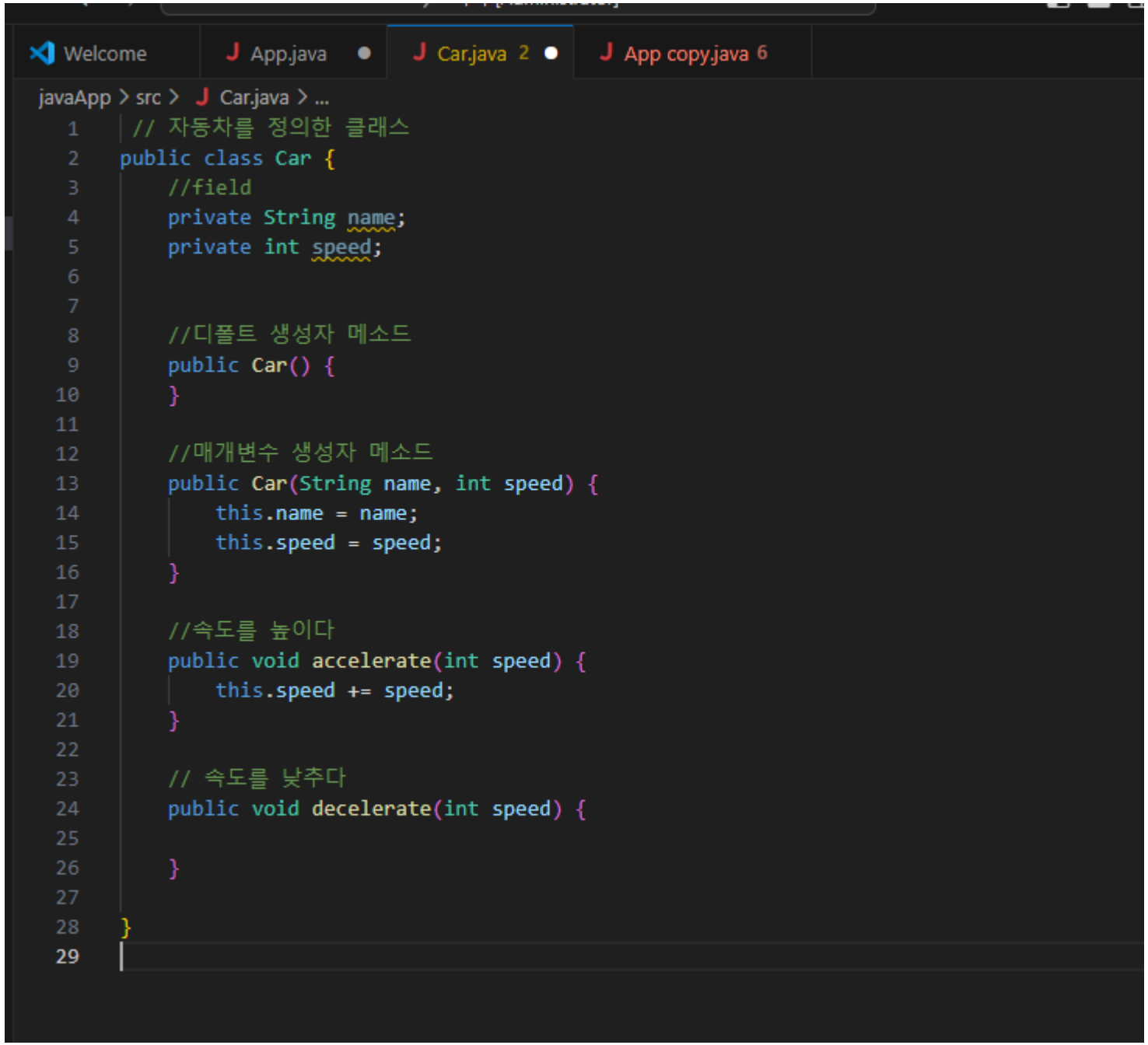
생성자 메소드 생성


JavaApp > src > Car.java > Car
1 // 자동차를 정의한 클래스
2 public class Car {
3 //field
4 private String name;
5 private int speed;
6
7
8
9 }
10
Go to Definition F12
Go to Declaration
Go to Type Definition
Go to Implementations Ctrl+F12
Go to References Shift+F12
Go to Super Implementation
Peek
Find All References Shift+Alt+F12
Find All Implementations
Show Call Hierarchy Shift+Alt+H
Show Type Hierarchy
Rename Symbol F2
Change All Occurrences Ctrl+F2
Format Document Shift+Alt+F
Format Document With...
Refactor... Ctrl+Shift+R
Source Action...
Cut Ctrl+X
Copy Ctrl+C


4 private String name;
5 private int speed;
6
7
8
9 }
10
Source Action
Generate Tests...
Organize imports Shift + Alt + O
Generate Getters and Setters...
Generate Getters...
Generate Setters...
Generate Constructors...
Generate hashCode() and equals()...
Generate toString()...
Override/Implement Methods...
Generate Delegate Methods...


Select fields to initialize by constructor(s).
name: String
speed: int

```
 // 자동차를 정의한 클래스
public class Car {
    //field
    private String name;
    private int speed;


    //디폴트 생성자 메소드
    public Car() {
    }

    //매개변수 생성자 메소드
    public Car(String name, int speed) {
        this.name = name;
        this.speed = speed;
    }

    //속도를 높이다
    public void accelerate(int speed) {
        this.speed += speed;
    }

    // 속도를 낮추다
    public void decelerate(int speed) {
```


Welcome
App.java
Car.java 2
App copy.java 6
javaApp > src > Car.java > ...
// 자동차를 정의한 클래스
public class Car {
//field
private String name;
private int speed;
//디폴트 생성자 메소드
public Car() {
}
//매개변수 생성자 메소드
public Car(String name, int speed) {
this.name = name;
this.speed = speed;
}
//속도를 높이다
public void accelerate(int speed) {
this.speed += speed;
}
// 속도를 낮추다
public void decelerate(int speed) {
}
}

레퍼런스 주소 정보 라고함
레퍼런스정보 = 주소정보

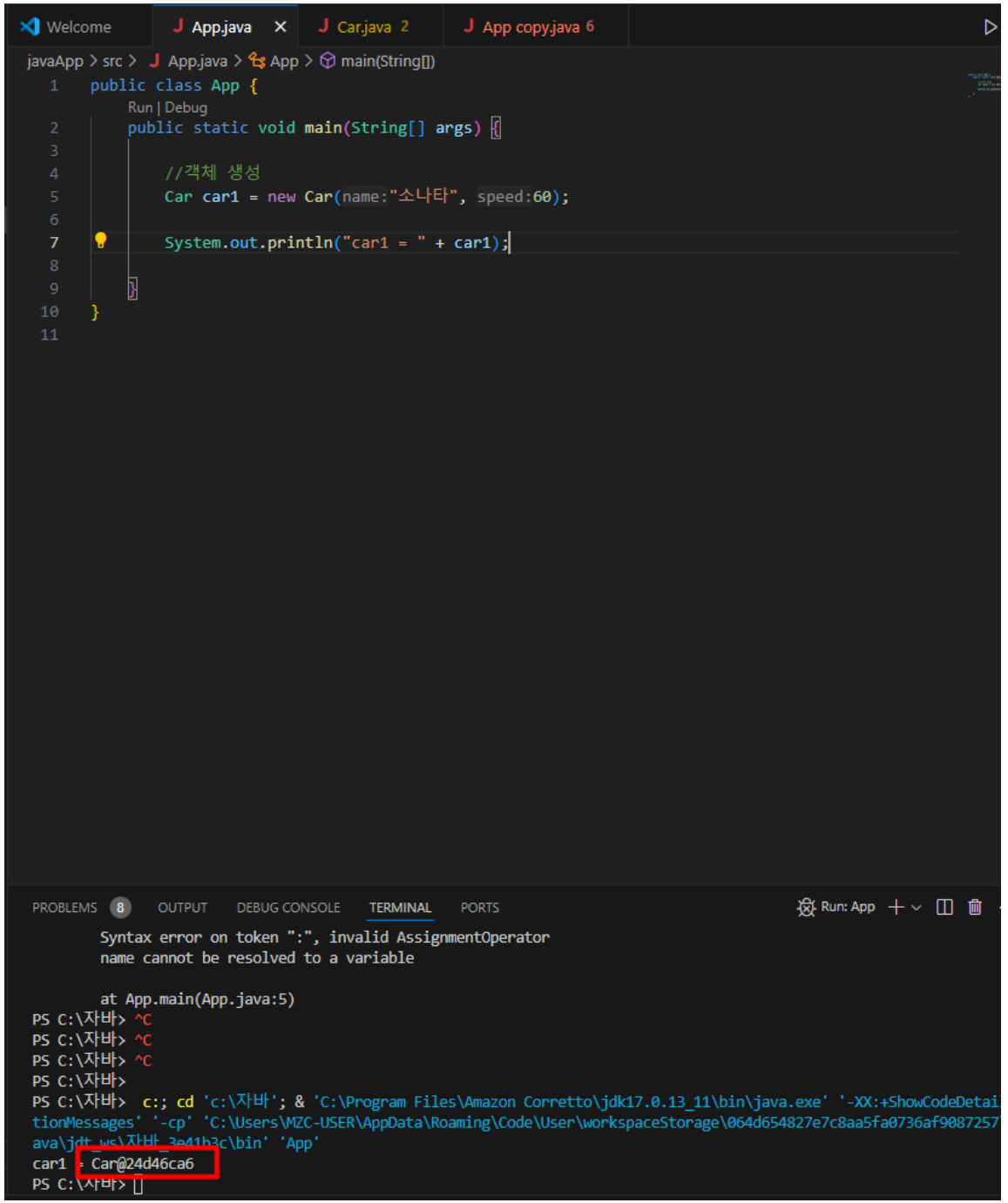
자바변수의 종류 (3가지)
로컬변수
인스턴스변수
클래스변수

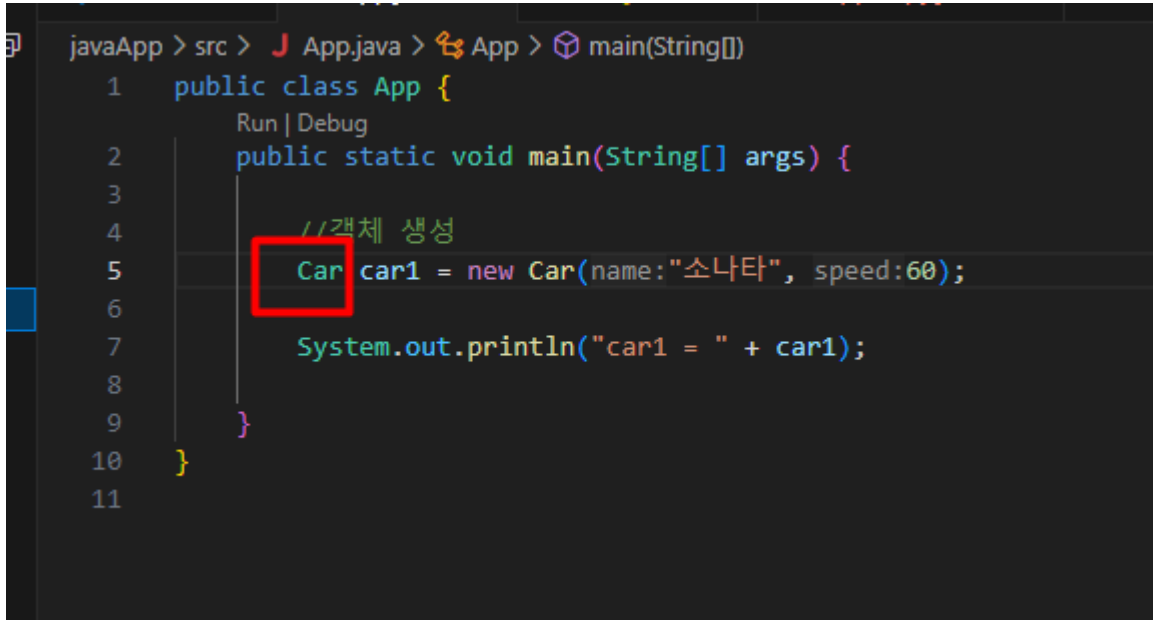
메소드안에 있으면 로컬변수

Static 라고 써져있으면 클래스 변수

//인스턴스 변수 (거의 변수는 인스턴스변수이다, 현대 제조사로 하면 클래스 변수)

컨트롤 + ~(틸드) 하면 콘솔창 on/off 가능





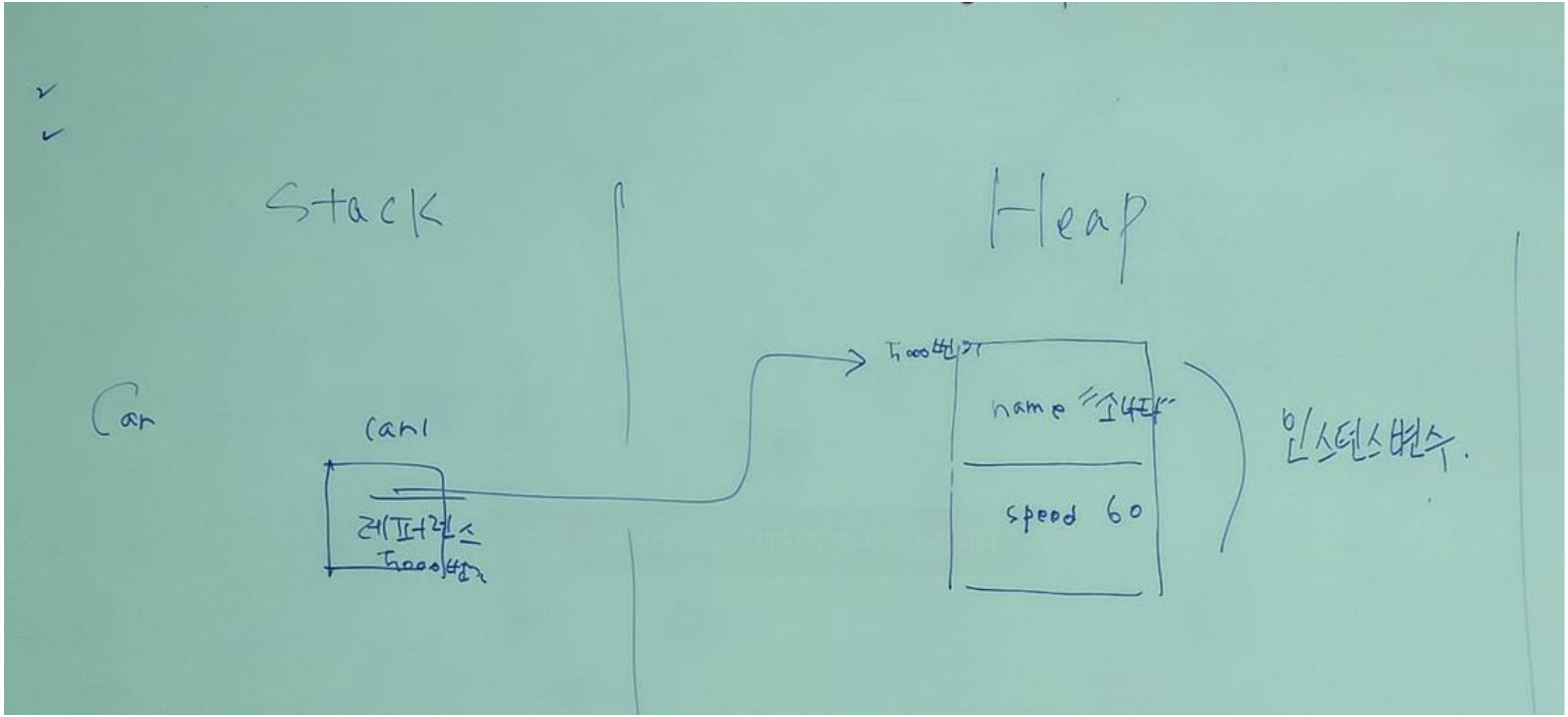
컨트롤 + Car 클릭하면 Car 이라는 메소드로 이동함

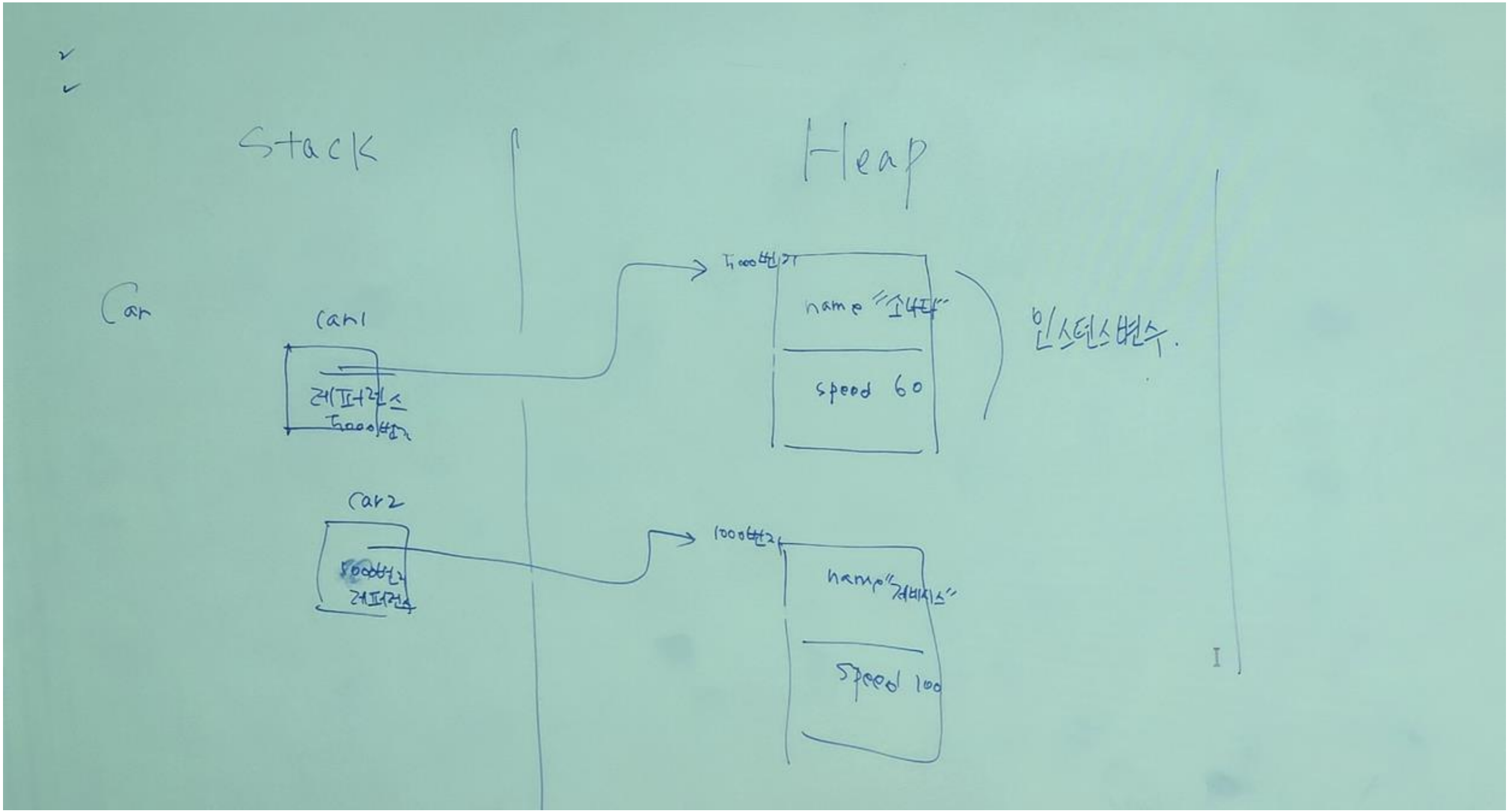
this

· 인스턴스가 자기 자신을 참조하는 데 사용하는 변수이다.

· 모든 인스턴스 메소드에는 this 참조변수를 사용하여 인스턴스 변수에 접근할 수 있다.

· 생성자 메소드에서 매개변수 이름과 인스턴스 변수의 이름이 같을 경우에는 인스턴스 변수 앞에 this 키워드를 붙여 구분해야 한다.

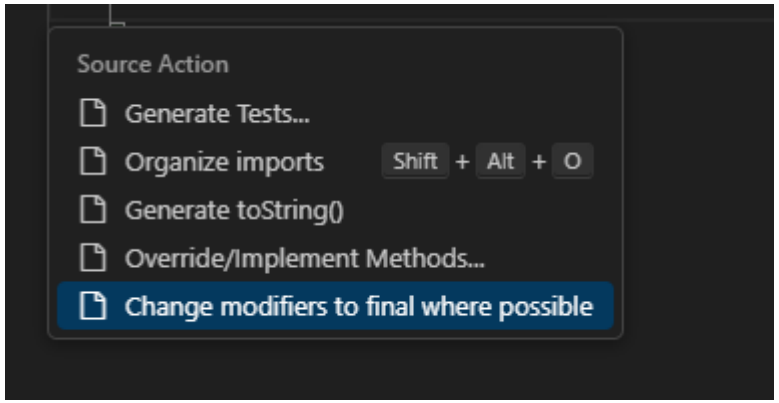

Stack
Heap
Car
car1
레퍼런스
name
speed 60
인스턴스변수.


Stack
Heap
Car
car1
레퍼런스
1000번지
1000번지
name "아반떼"
speed 60
인스턴스변수.
car2
레퍼런스
1000번지
name "제네시스"
Speed 100

Getter 메소드 (조회)
Getter 메소드 (변경)

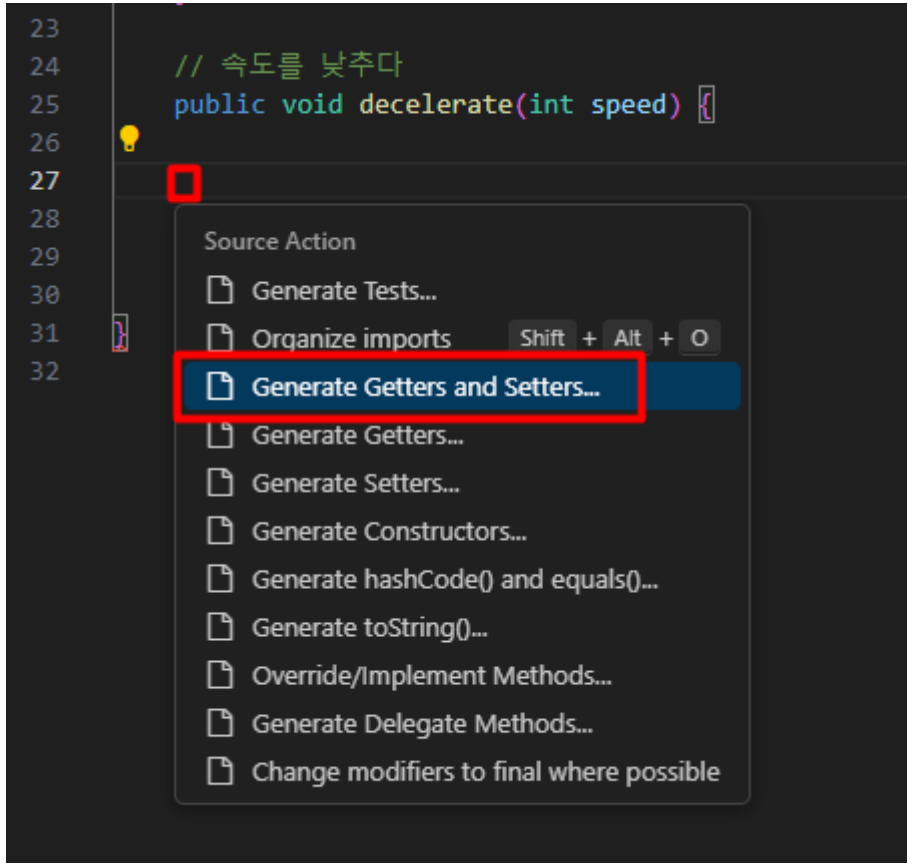
```
public class App {

    public static void main(String[] args) {

        //객체 생성
        Car car1 = new Car("소나타", 60);

        System.out.println("car1 = " + car1);

        Car car2 = new Car("제네시스", 100);

        System.out.println("car2 = " + car2);

        System.out.println("car1.name" +
car2.getName());



    }
}
```

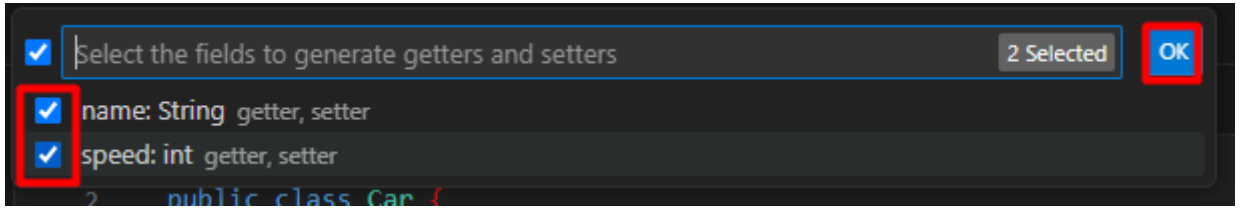
이거 물어봐야함 getname 할때 아래처럼만 나옴




```java
public class App {

    public static void main(String[] args) {

        //객체 생성
        Car car1 = new Car("소나타", 60);

        System.out.println("car1 = " + car1);

        Car car2 = new Car("제네시스", 100);

        System.out.println("car2 = " + car2);

        System.out.println("car1.name" +
car2.getname());



    }
}
```

클래스명 인터페이스명 제외하고는 소문자 지향 (자바는)



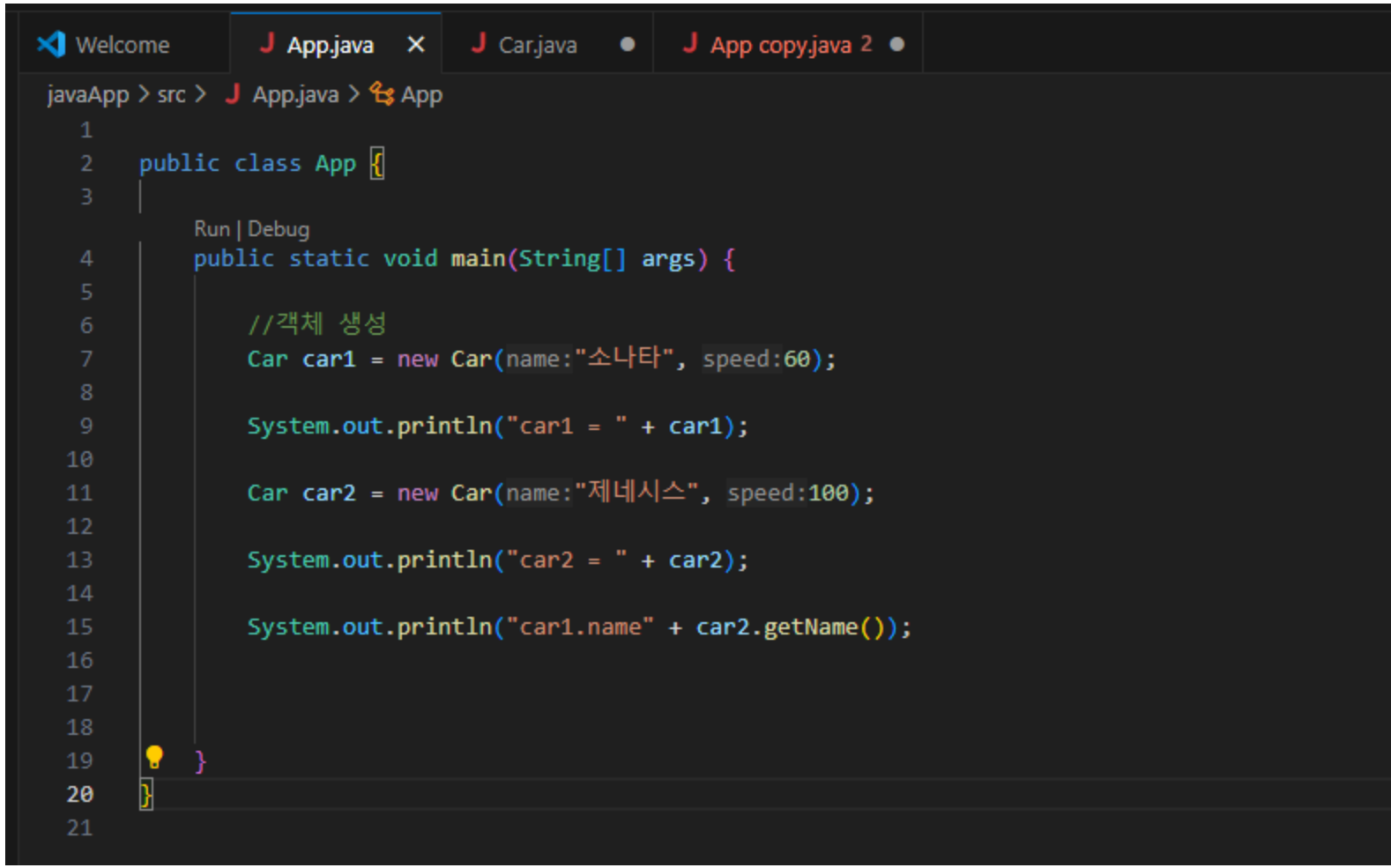


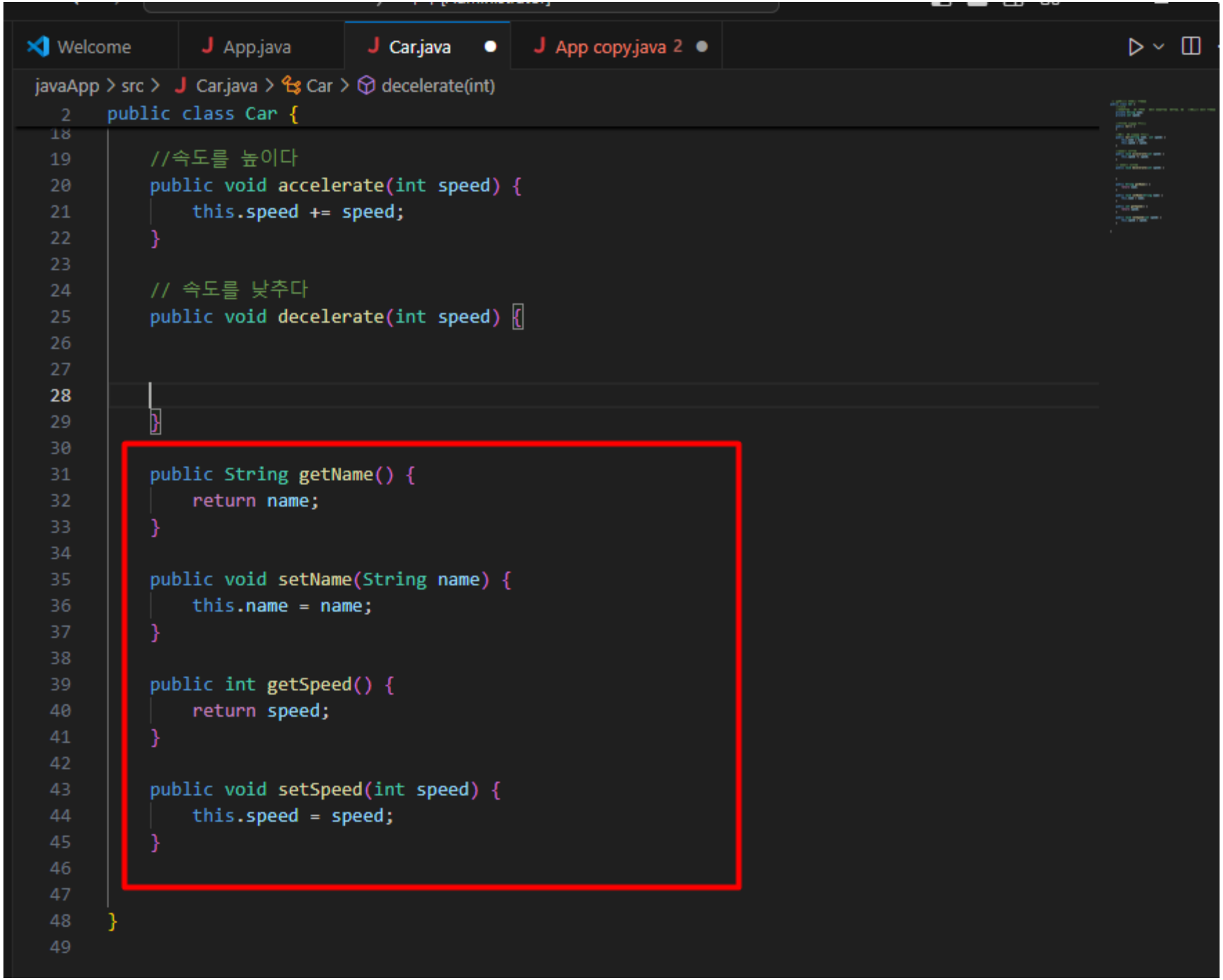
위처럼 하면

This.speed +=This.speed + speed;

자동생성
getName 전부 자동생성 되고
N은 대문자여야함

```
public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public int getSpeed() {
    return speed;
}

public void setSpeed(int speed) {
    this.speed = speed;
}
```


Welcome
App.java
Car.java
App copy.java 2
javaApp > src > App.java > App
Run | Debug
public class App {
public static void main(String[] args) {
//객체 생성
Car car1 = new Car(name:"소나타", speed:60);
System.out.println("car1 = " + car1);
Car car2 = new Car(name:"제네시스", speed:100);
System.out.println("car2 = " + car2);
System.out.println("car1.name" + car2.getName());
}
}


Welcome
App.java
Car.java
App copy.java 2
javaApp > src > Car.java > Car > decelerate(int)
public class Car {
//속도를 높이다
public void accelerate(int speed) {
this.speed += speed;
}
// 속도를 낮추다
public void decelerate(int speed) {
}
public String getName() {
return name;
}
public void setName(String name) {
this.name = name;
}
public int getSpeed() {
return speed;
}
public void setSpeed(int speed) {
this.speed = speed;
}
}

```java
ublic class MethodExam {

    // 두 정수의 합을 구하는 메소드 구현부
    public static int add(int a, int b) {
        return a + b;
    }

    // 두 정수의 곱을 구하는 메소드 구현부
    public static int multiply(int a, int b) {
        return a * b;
    }

    public static void main(String[] args) {
        // 메소드 호출부
        int result1 = add(a:1, b:3);  <- 숫자만 넣으면 됨 (a:, b: 안써야됨)
        System.out.println("result1 : " + result1);

        int result2 = MethodExam.add(a:1, b:3); <- 숫자만 넣으면 됨 (a:, b: 안써야됨)

        System.out.println("result2 : " + result2);

        MethodExam obj = new MethodExam.add();
        System.out.println(obj.multiply(a:1, b:2); <- 숫자만 넣으면 됨 (a:, b: 안써야됨)

    }
```

}

스트링 타입

UML
객체지향 설계 언어
클래스다이어그램이라고 표현한다

클래스이름
속성
오퍼레이션

기본자료형

참조형(레퍼런스)클래스
인터페이스
lnum
하나더있는데 안보였음

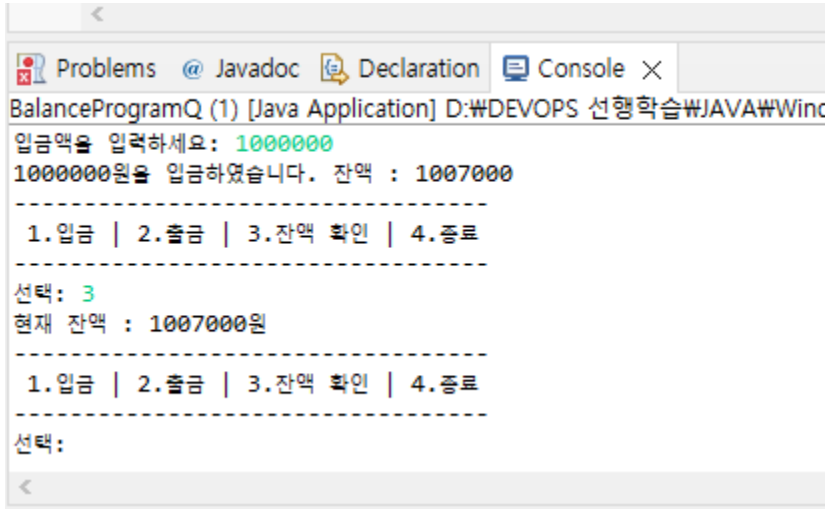
UML표기법 (객체지향모델링 설계 언어)
- 프라이빗
+ 퍼블릭
# 프로텍티드(상속)

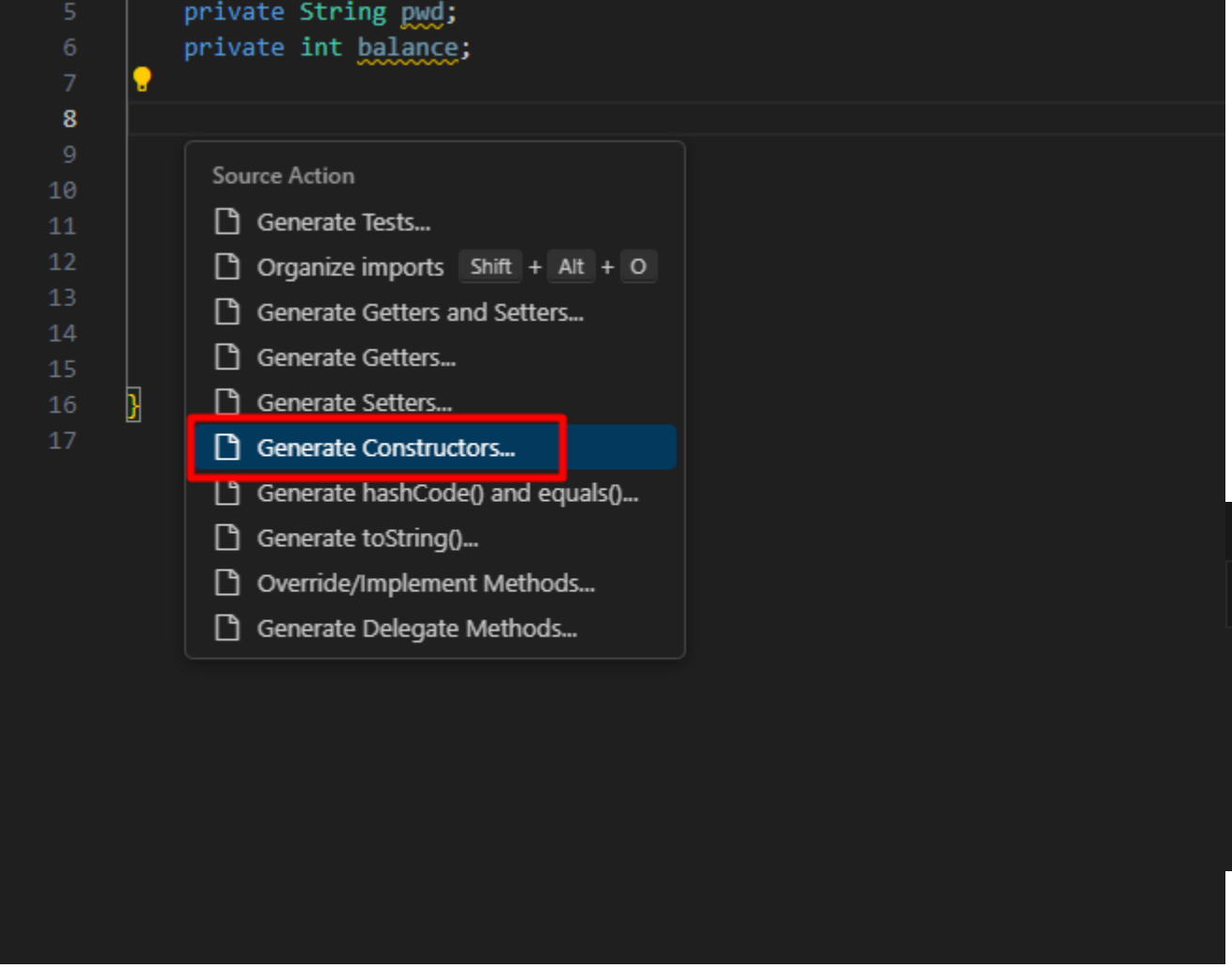
UML표기법 (객체지향모델링 설계 언어)
- 프라이빗
+ 퍼블릭

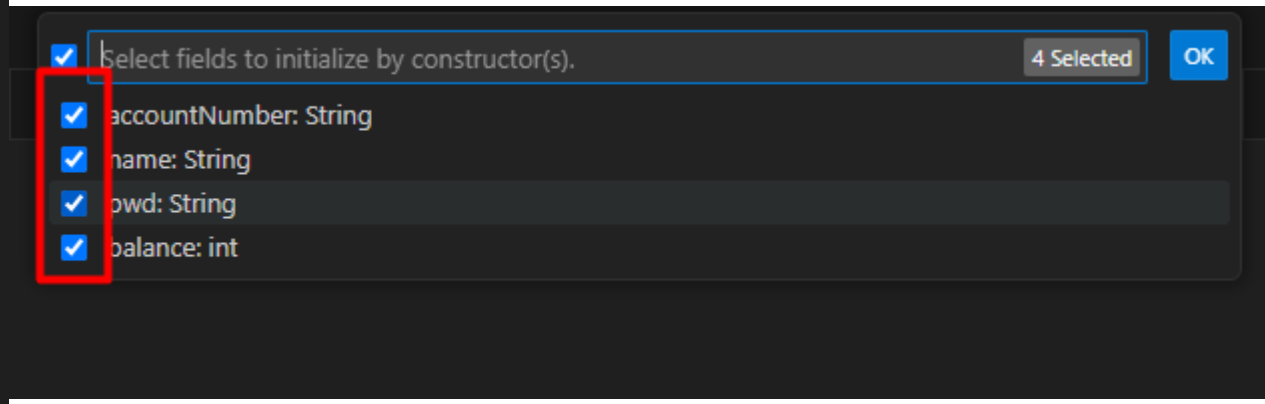
월에 대한 일수를 알려주는 프로그램

```java
package method;

import java.util.Scanner;

public class BalanceProgramQ {
    public static void main(String[] args) {
        int balance = 0;

        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-----------------------------------");
            System.out.println(" 1.입금 | 2.출금 | 3.잔액 확인 | 4.종료");
            System.out.println("-----------------------------------");
            System.out.print("선택: ");

            int choice = scanner.nextInt();
            int amount;

            switch (choice) {
                case 1:
                    System.out.print("입금액을 입력하세요: ");
                    amount = scanner.nextInt();
                    balance = deposit(balance, amount);
                    break;

                case 2:
                    System.out.print("출금액을 입력하세요: ");
                    amount = scanner.nextInt();
                    balance = withdraw(balance, amount);
                    break;

                case 3:
                    System.out.println("현재 잔액 : " + balance + "원");
                    break;

                case 4:
                    System.out.println("시스템을 종료합니다.");
                    return;

                default:
                    System.out.println("올바른 선택이 아닙니다. 다시 선택해주세요.");

            }
        }
    }
    public static int deposit(int balance, int amount){
        balance += amount;
        System.out.println(amount + "원을 입금하였습니다. 잔액 : " + balance);

        return balance;
    }

    public static int withdraw(int balance, int amount){
        if(balance >= amount) {
            balance -= amount;
            System.out.println(amount +"원을 출금하였습니다. 잔액 : " + balance);
        } else {
            System.out.println(amount + "원을 출금하려 했으나 잔액이 부족합니다.");
        }
        return  balance;
    }
}
```

```java
package method;

import java.util.Scanner;

public class BalanceProgramQ {
    public static void main(String[] args) {
        int balance = 0;

        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-----------------------------------");
            System.out.println(" 1.입금 | 2.출금 | 3.잔액 확인 | 4.종료");
            System.out.println("-----------------------------------");
            System.out.print("선택: ");

            int choice = scanner.nextInt();
            int amount;

            switch (choice) {
                case 1:
                    System.out.print("입금액을 입력하세요: ");
                    amount = scanner.nextInt();
                    balance = deposit(balance, amount);
                    break;

                case 2:
                    System.out.print("출금액을 입력하세요: ");
                    amount = scanner.nextInt();
                    balance = withdraw(balance, amount);
                    break;

                case 3:
                    System.out.println("현재 잔액 : " + balance + "원");
                    break;

                case 4:
                    System.out.println("시스템을 종료합니다.");
                    return;

                default:
                    System.out.println("올바른 선택이 아닙니다. 다시 선택해주세요.");

            }
        }
    }
    public static int deposit(int balance, int amount){
        balance += amount;
        System.out.println(amount + "원을 입금하였습니다. 잔액 : " + balance);

        return balance;
    }

    public static int withdraw(int balance, int amount){
        if(balance >= amount) {
            balance -= amount;
            System.out.println(amount +"원을 출금하였습니다. 잔액 : " + balance);
        } else {
            System.out.println(amount + "원을 출금하려 했으나 잔액이 부족합니다.");
        }
        return  balance;
    }
}
```

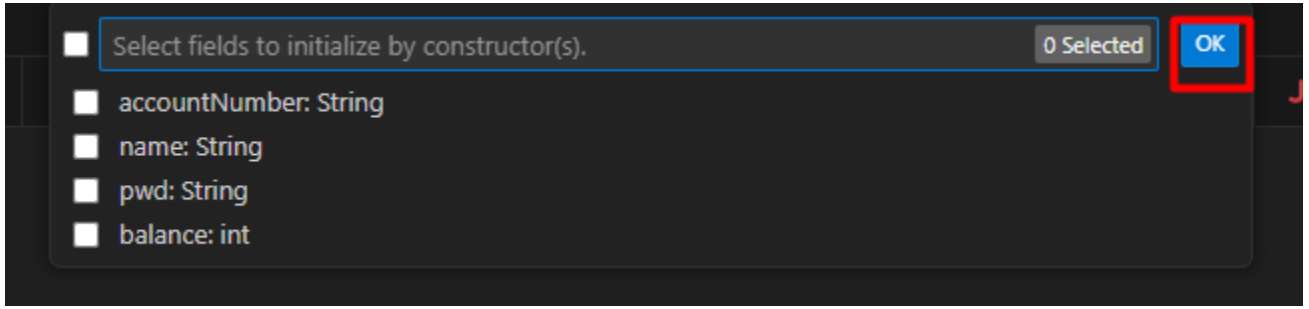


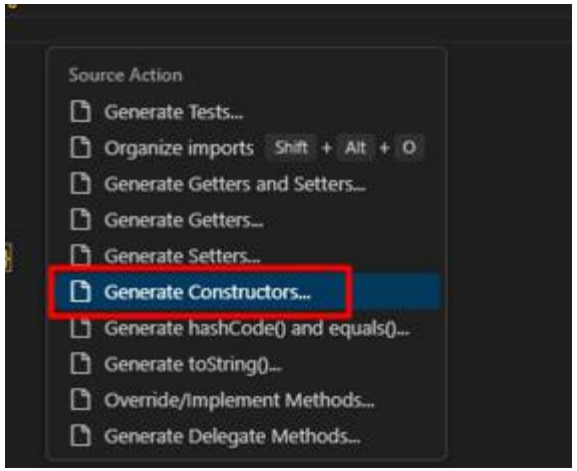





```java
// 생성자 메소드 (Consructor Method)
public Account(String accountNumber, String name, String pwd, int balance) {
    this.accountNumber = accountNumber;
    this.name = name;
    this.pwd = pwd;
    this.balance = balance;
}
```

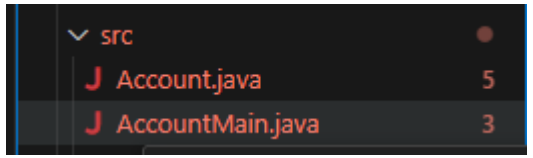
체크안하고 생성해서 디폴트 메소드





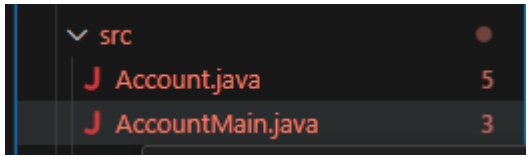

```
public Account() {
}
```

```
        this.balance = balance;
    }

    // instance method
    public int getBalance() {
        return balance;

    // 입금하다   <- 여기서 this를 안넣으면 입금
금액이 변경이 안됨!! 주의!!                    입금부터 오류
    public void deposit(int balance) {
        this.balance += balance;
        //this.balance = this.balance + balance;
    }

     // 출금하다   <-
     public void withdraw(int balance) {
        this.balance -= balance;
        //this.balance = this.balance - balance;


    // 계좌이체하다
    public void transferAccount(Account account,
int balance); {
        this.withdraw(balance);
        account.deposit(balance);
```

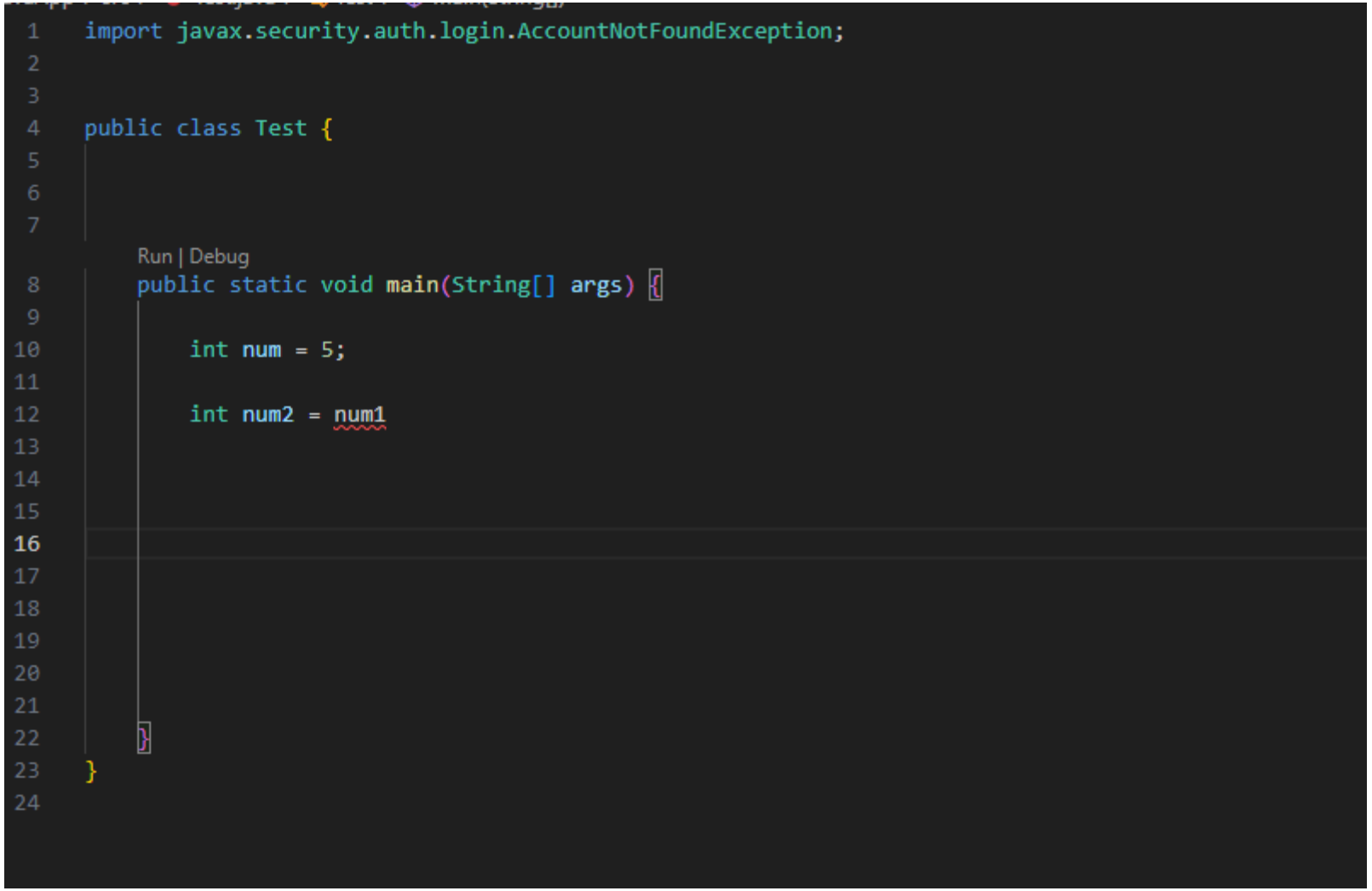


```
정이 됐다고 생각하면 된다)
        Account account2 = new Account("2222",
"이길동", "bbbb", 20000);

        // "1111" 고객 계좌의 잔고를 조회하다
        int balance = account1.getBalance();
        System.out.println(balance);

        // "2222" 고객 계좌의 잔고를 조회하다
        balance = account2.getBalance();
        System.out.println(balance);

        // "1111" 고객 계좌로 10000원을 입금하다
        account1.deposit(10000);
        System.out.println(account1.getBalance(
));

        // "2222" 고객 계좌로 5000원을 출금하다
        account2.withdraw(5000);
        System.out.println(account2.getBalance(
));

        // "1111" 고객 계좌에서 "2222" 고객 계좌
로 1000원을 이체하다
        account1.transferAccount(account2
```

입금부터 오류

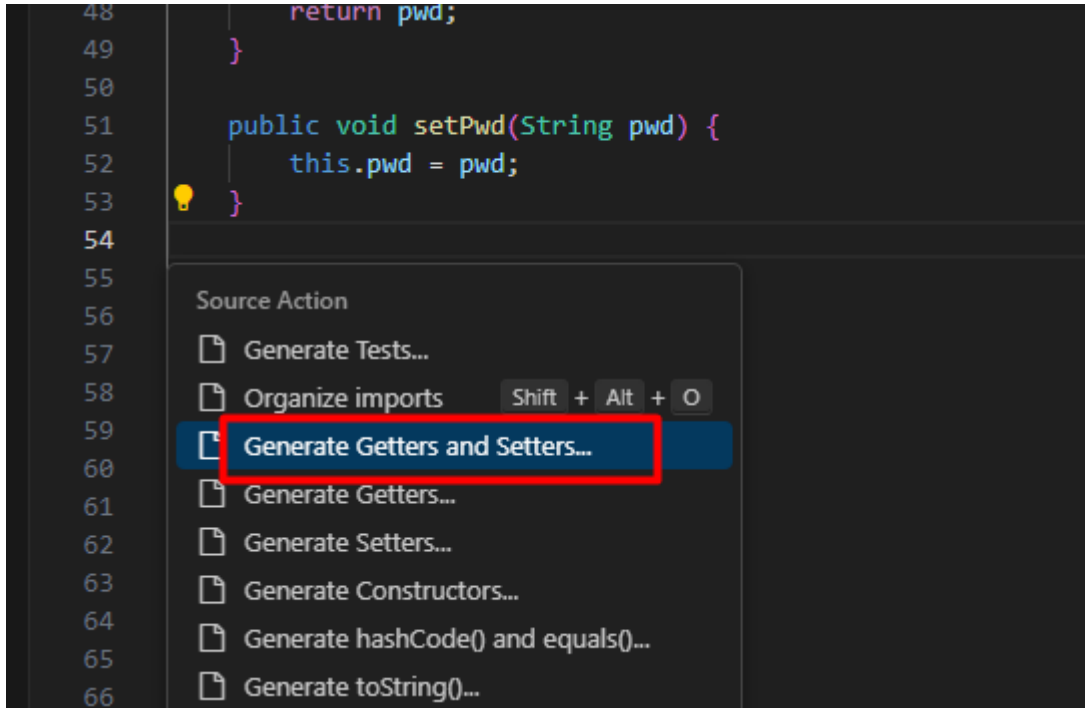


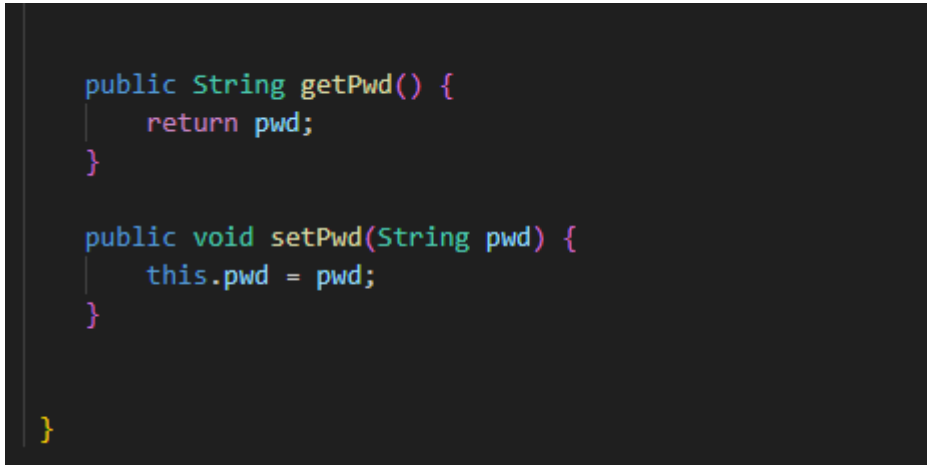
```
    static int num1;
    int num2;

    Car car1; //이걸보면  Car car1 = null; 이렇게 초기화작업이 이루어지는구나 하고 알아야함,  스
택에 할당안됨, 로컬지역변수가 아니기때문에)


    public static void main(String[] args) {
        int num3

        System.out.println(num1);
        Test test = new Test();  //num2는 객체 생성해줘야 쓸수있다
        System.out.println(num2);
        System.out.println(num3); //num3는 int num3 =0; 으로 초기화작업을 해야 쓸수있다

        Car car2 = null



    }
}
```

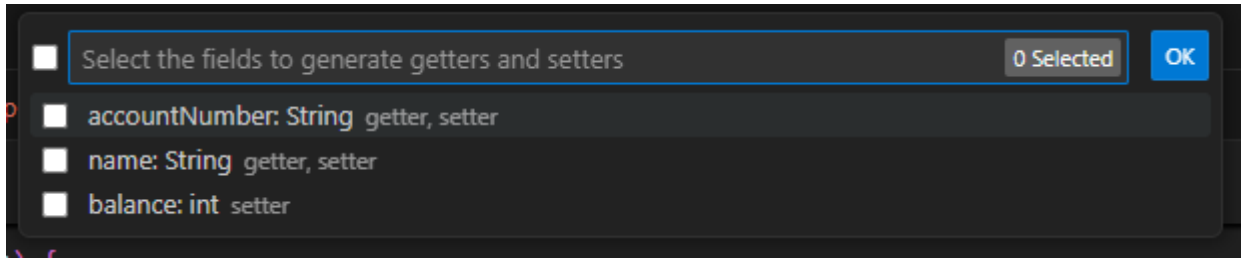

import javax.security.auth.login.AccountNotFoundException;
public class Test {
Run | Debug
public static void main(String[] args) {
int num = 5;
int num2 = num1
}
}

```
import
javax.security.auth.login.AccountNotFoundExcept
ion;


public class Test {



    public static void main(String[] args) {

        Account account1 = new Account("1111",
"일길동", "aaaa", 10000);

        Account copy = src;









    }
}
```

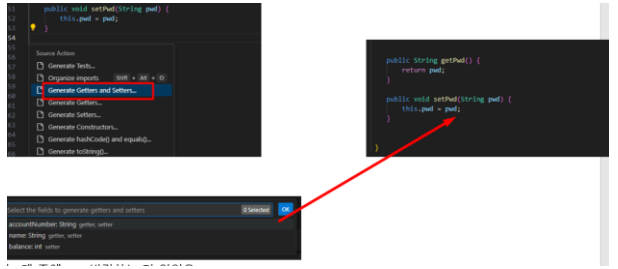
PW변경하는 메소드? 생성할때





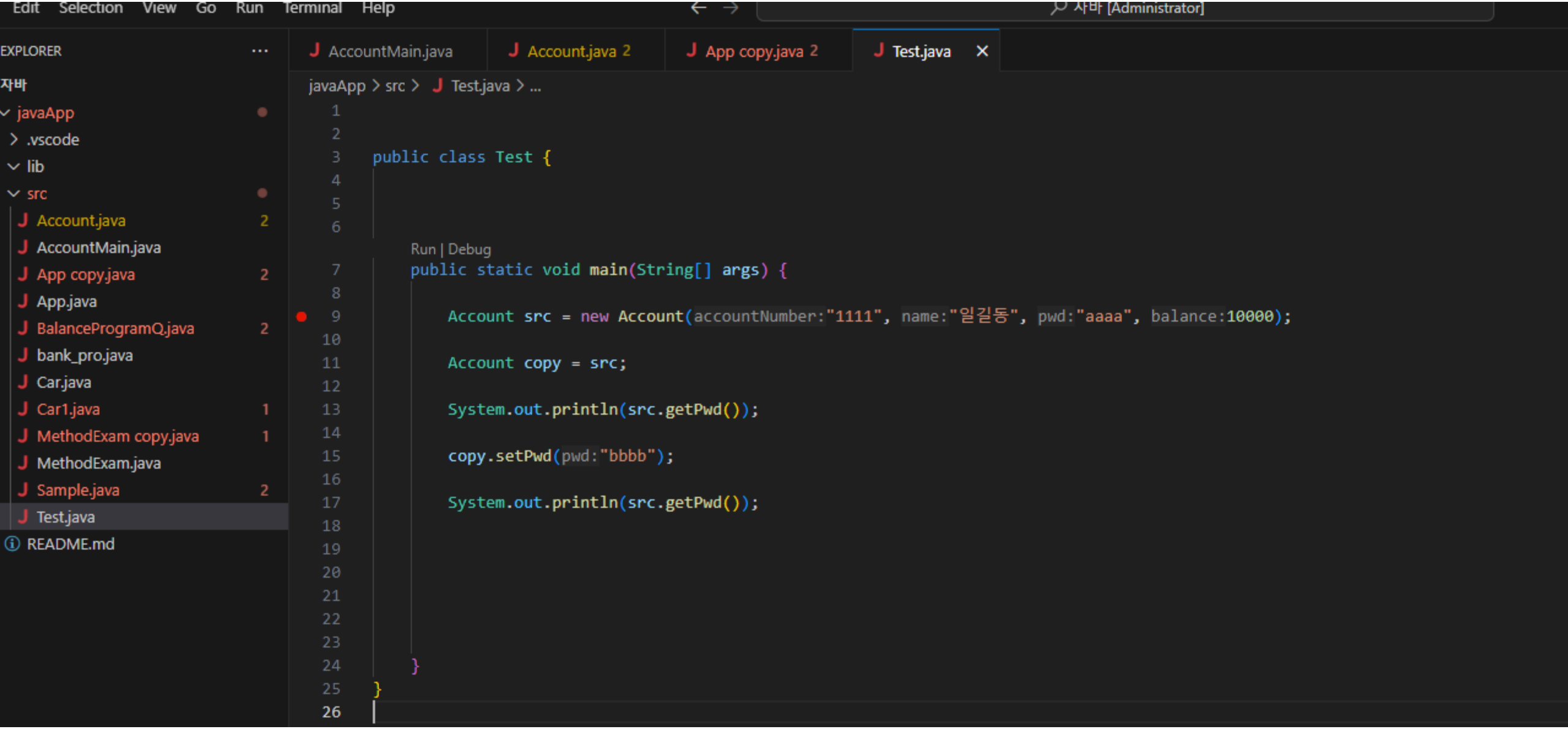


이 3개 중에 PW 변경하는 거 있었음



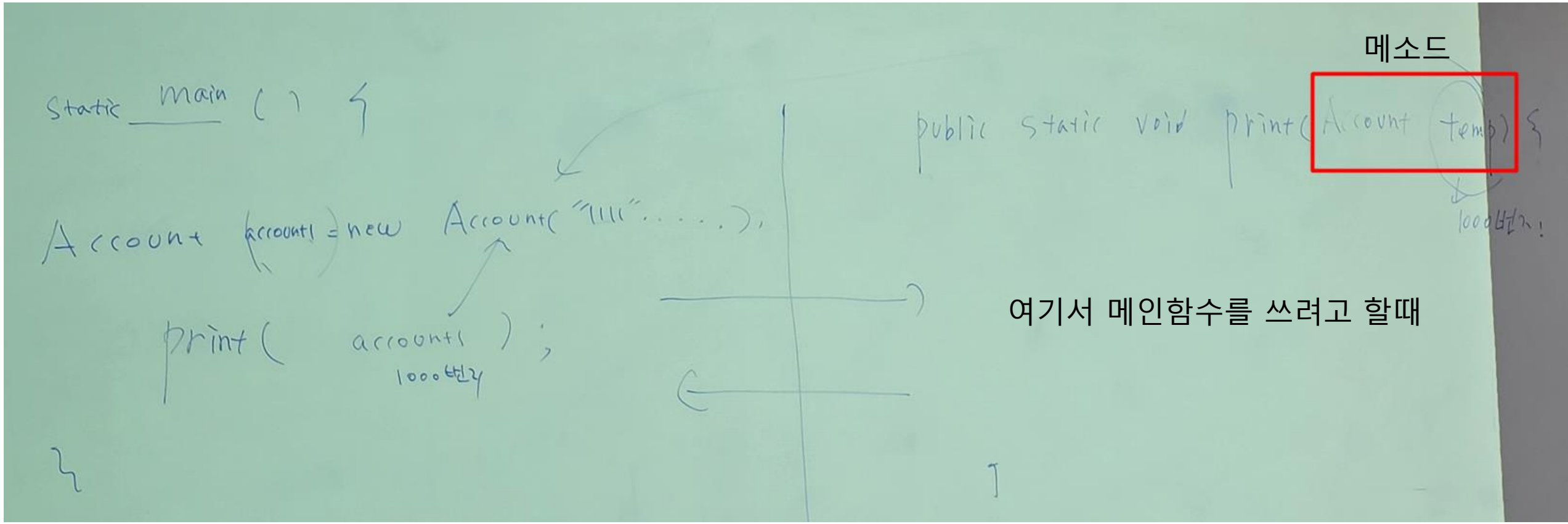
PW변경하는 메소드? 생성할때

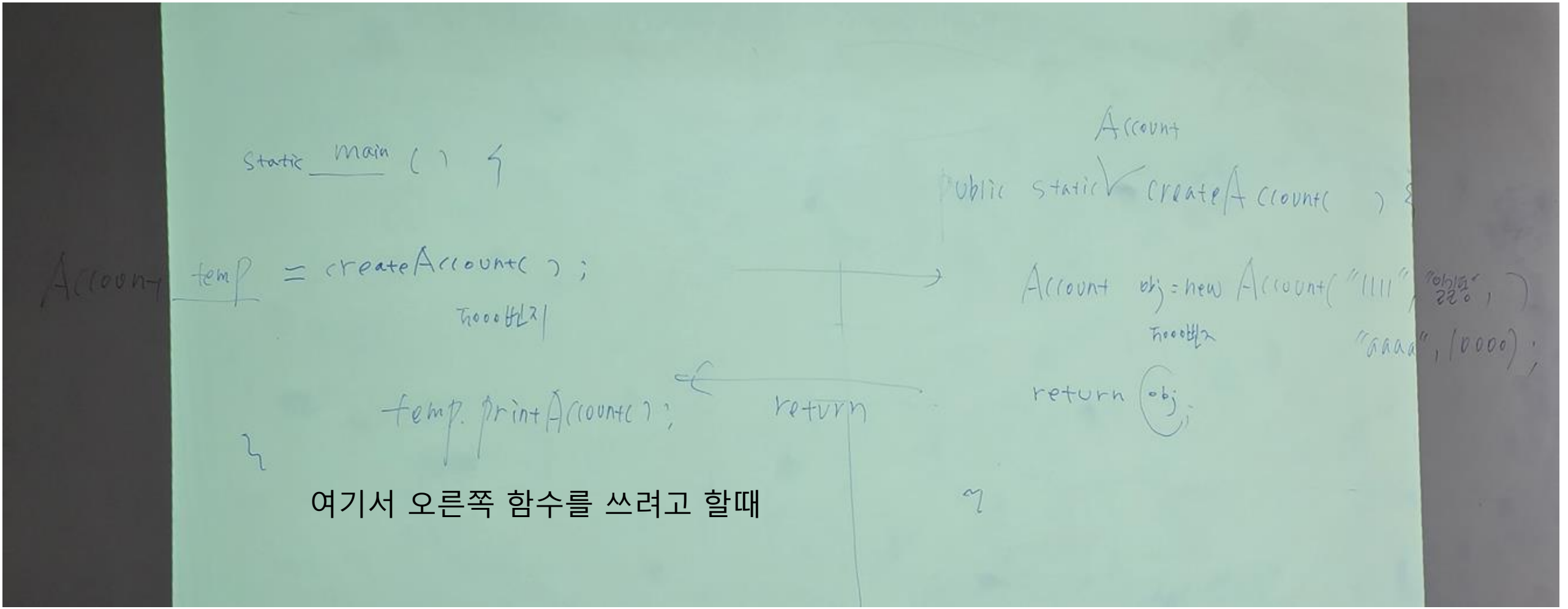
```
public class Test {


    public static void main(String[] args) {
        Account src = new Account("1111", "일길동", "aaaa",
10000);
        Account copy = src;
        System.out.println(src.getPwd());
        copy.setPwd("bbbb");
        System.out.println(src.getPwd());
```


Edit Selection View Go Run Terminal Help
자바 [Administrator]
EXPLORER
자바
javaApp
.vscode
lib
src
Account.java 2
AccountMain.java
App copy.java 2
App.java
BalanceProgramQ.java 2
bank_pro.java
Car.java
Car1.java 1
MethodExam copy.java 1
MethodExam.java
Sample.java 2
Test.java
README.md
AccountMain.java
Account.java 2
App copy.java 2
Test.java
javaApp > src > Test.java > ...
public class Test {
Run | Debug
public static void main(String[] args) {
Account src = new Account(accountNumber:"1111", name:"일길동", pwd:"aaaa", balance:10000);
Account copy = src;
System.out.println(src.getPwd());
copy.setPwd(pwd:"bbbb");
System.out.println(src.getPwd());
}
}

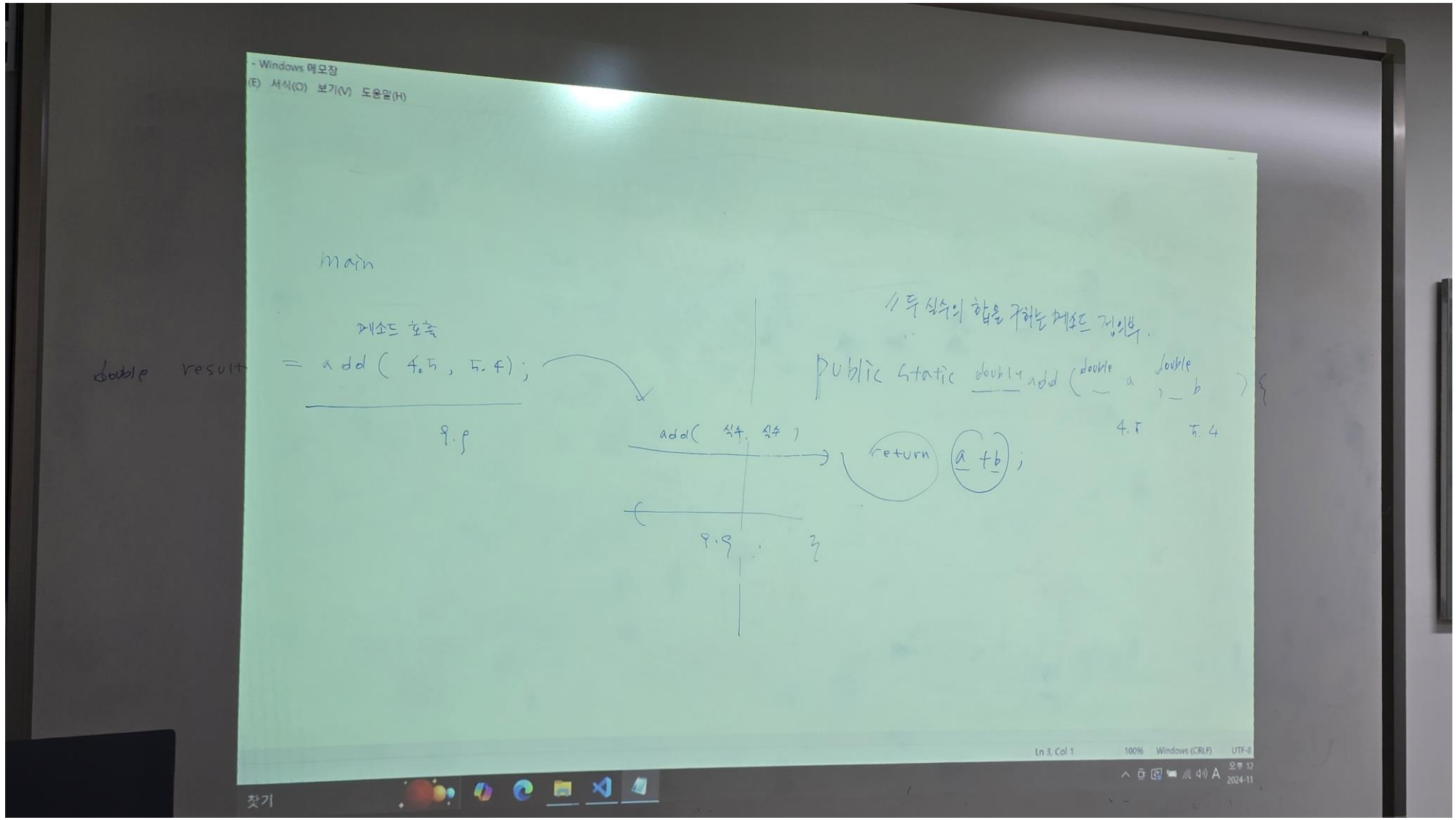
```java
public class Test {



    public static void main(String[] args) {

        // 계좌 객체 생성
        Account account1 = new Account("1111", "일길동", "aaaa",
10000);

        account1.printAccount();


        // 계좌 객체 생성
        Account account2 = new Account("2222", "이길동", "bbbb",
20000);

        account1.printAccount();

    }
}
```

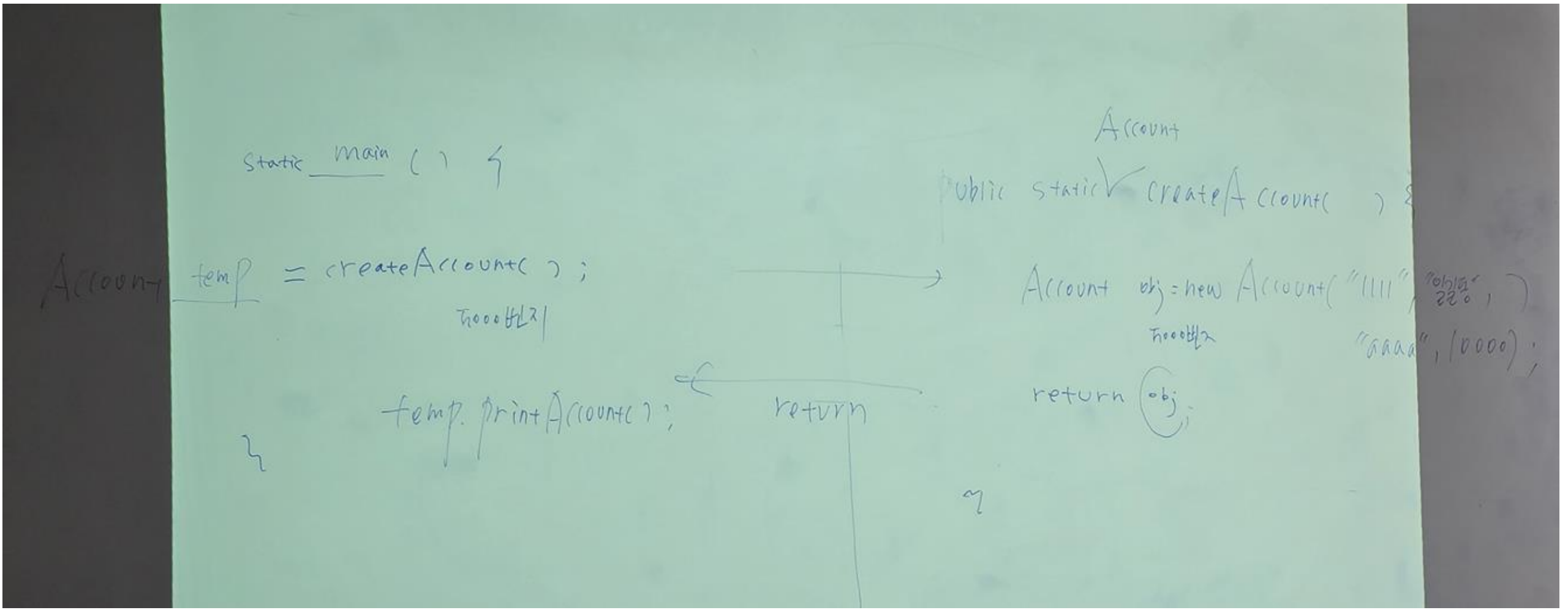
콜바이레퍼런스

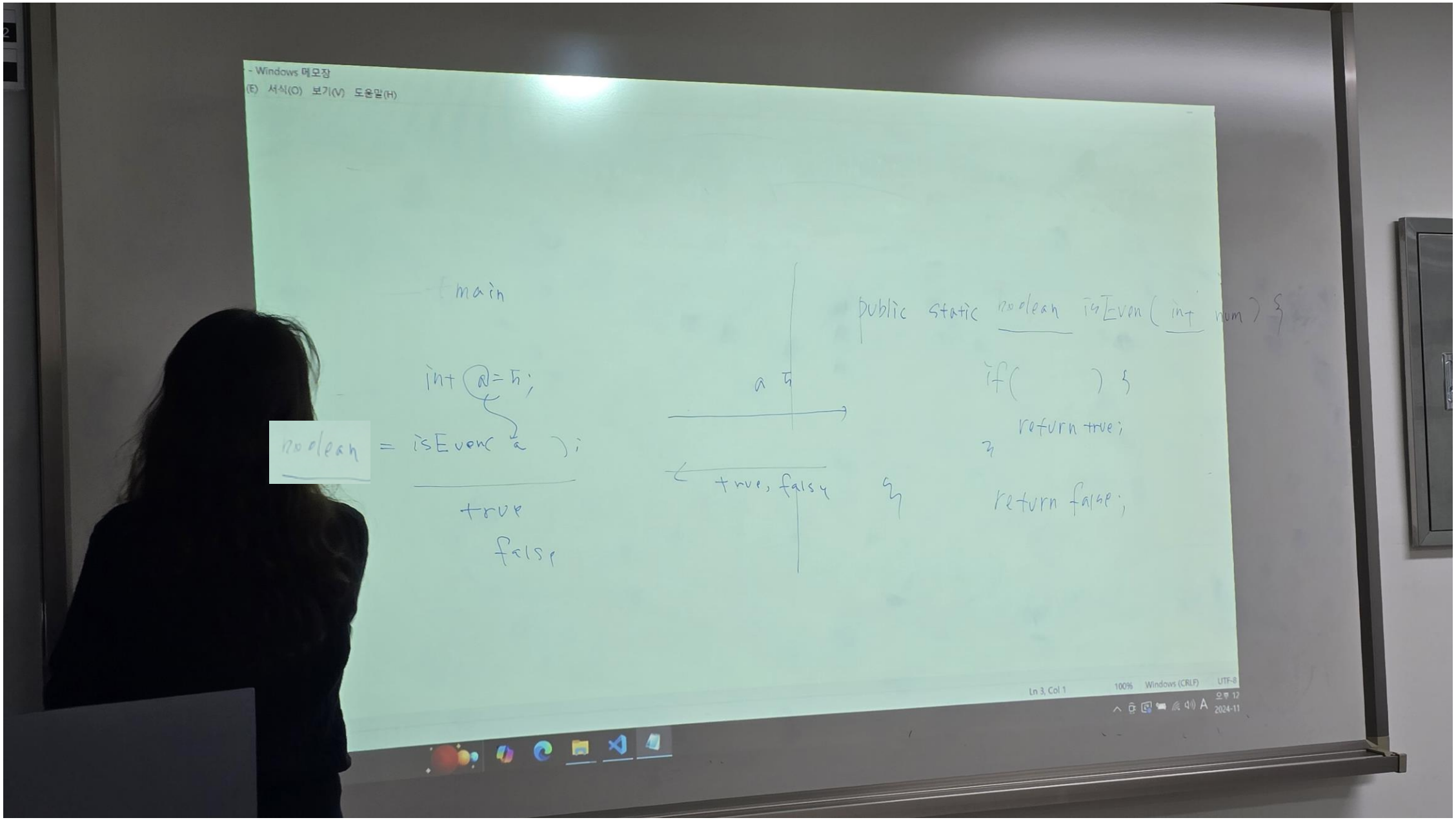

메소드
static main ( ) {
Account account1 = new Account("1111"....);
print(account1);
1000번가
}
public static void print(Account temp) {
1000번가!
여기서 메인함수를 쓰려고 할때
}

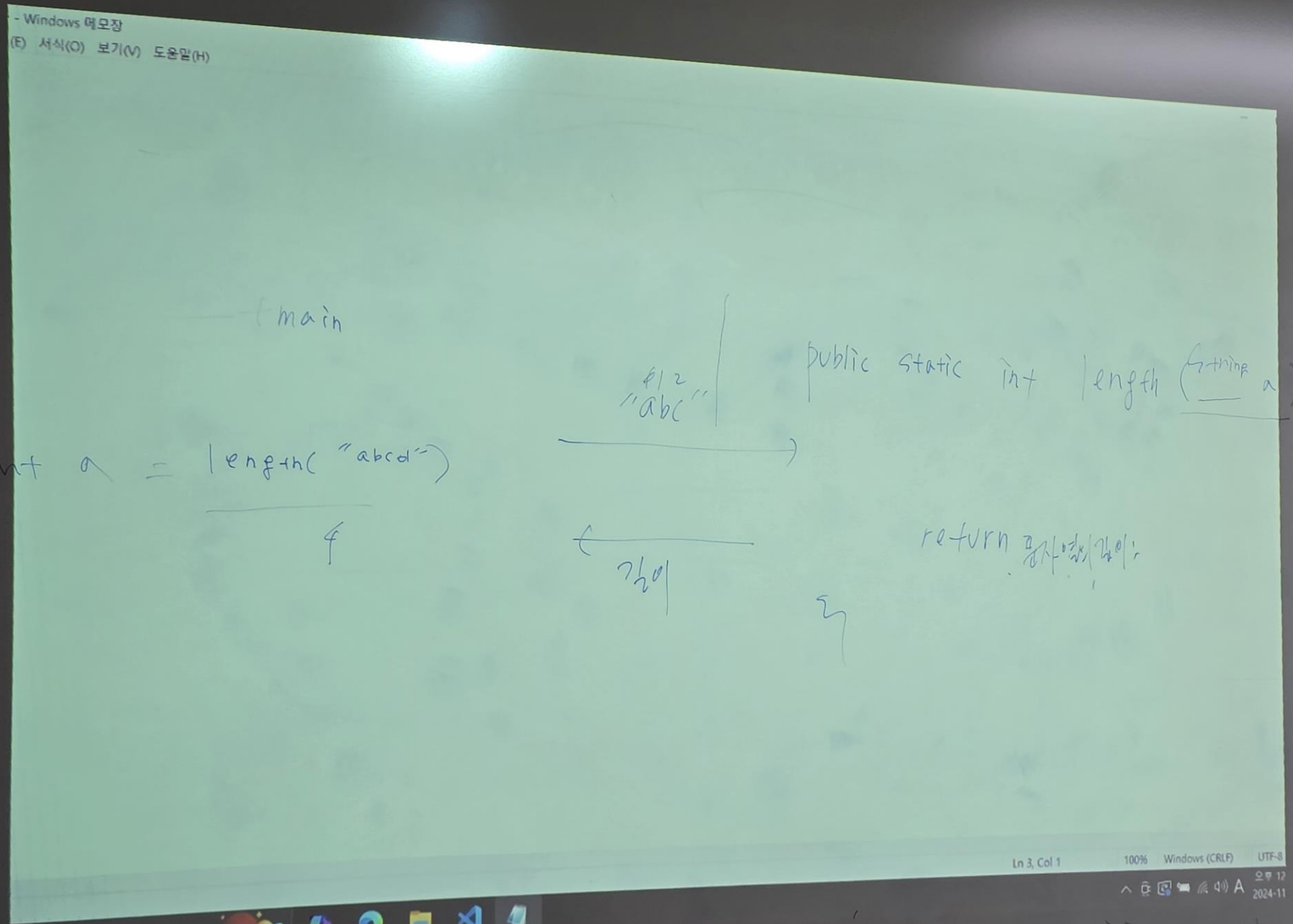


여기서 오른쪽 함수를 쓰려고 할때

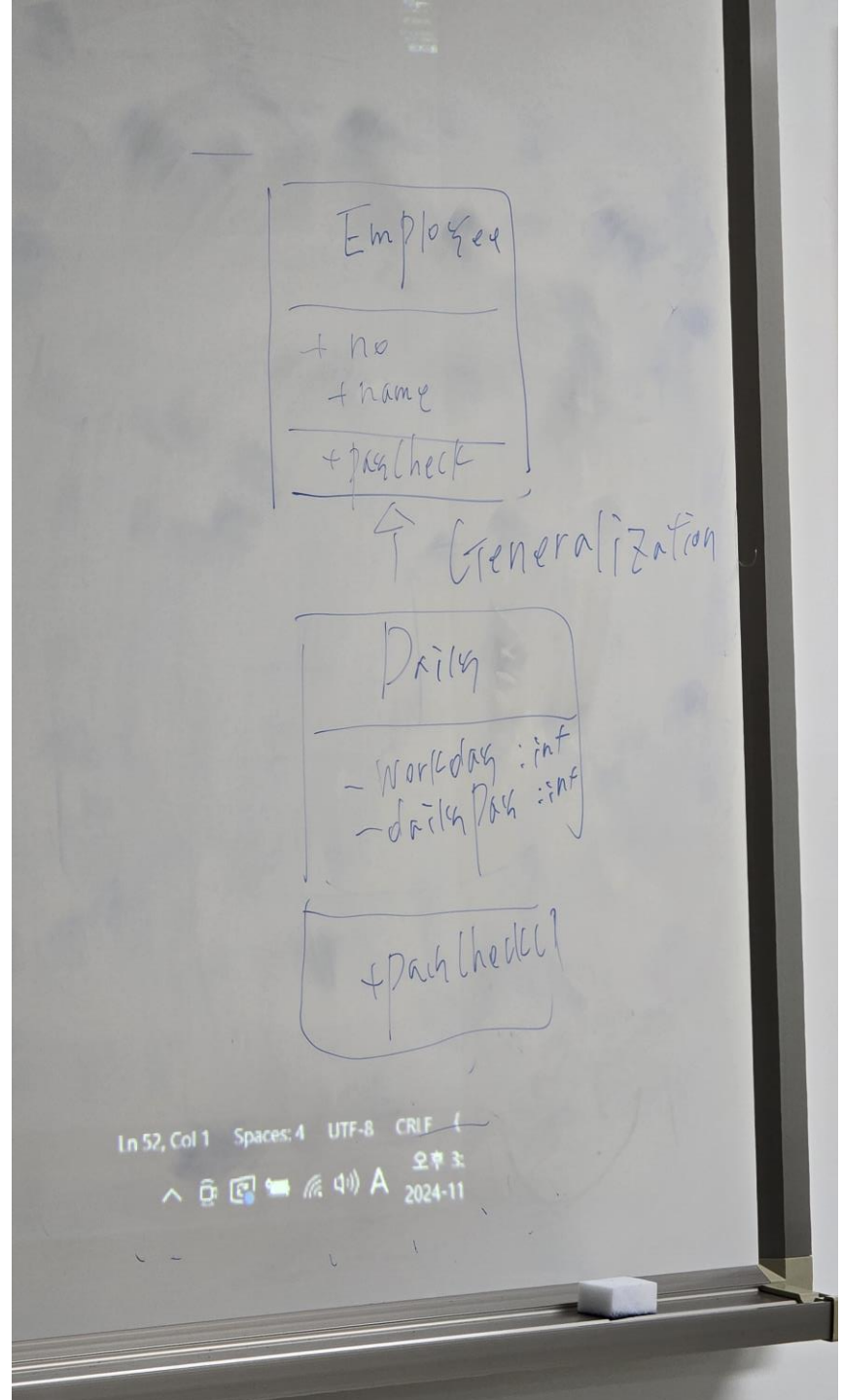

- Windows 메모장
(E) 서식(O) 보기(V) 도움말(H)
main
메소드 호출
double result = add (4.5 , 5.4);
9.9
add(실수, 실수)
// 두 실수의 합을 구하는 메소드 정의부.
public static double add (double a , double b) {
4.5 5.4
return a+b;
9.9
찾기
Ln 3, Col 1
100%
Windows (CRLF)
UTF-8
오후 12
2024-11


static main ( ) {
Account temp = createAccount( );
주소값
temp.printAccount( );
}
Account
public static createAccount( ) {
Account obj = new Account("1111", 아이유, 22, "aaaa", 10000);
주소값
return obj;
}
return


- Windows 메모장
(F) 서식(O) 보기(V) 도움말(H)
main
int a = 5;
boolean = isEven(a);
true
false
a 5
true, false
public static boolean isEven(int num) {
if( ) {
return true;
}
return false;
Ln 3, Col 1
100%
Windows (CRLF)
UTF-8
오후 12
2024-11


- Windows 메모장
(E) 서식(O) 보기(V) 도움말(H)
main
int a = length("abcd");
4
"abc"
public static int length(String a)
return 문자열의 길이;
Ln 3, Col 1
100%
Windows (CRLF)
UTF-8
오후 12
2024-11

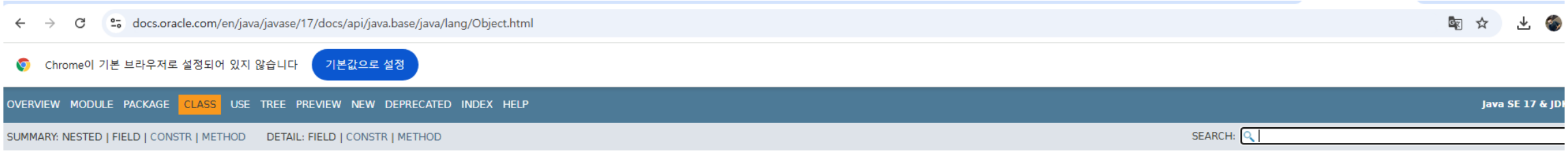
상속


Employee
+ no
+ name
+ payCheck
Generalization
Daily
- workday : int
- dailyPay : int
+ payCheck()
In 52, Col 1 Spaces: 4 UTF-8 CRLF
오후 3:
2024-11


OVERVIEW MODULE PACKAGE CLASS USE TREE PREVIEW NEW DEPRECATED INDEX HELP
Java® Platform, Standard Edition & Java Development Kit
Version 17 API Specification
This document is divided into two sections:
Java SE
The Java Platform, Standard Edition (Java SE) APIs define the core Java platform for general-purpose comp
JDK
The Java Development Kit (JDK) APIs are specific to the JDK and will not necessarily be available in all imp
All Modules
Java SE
JDK
Other Modules
Module
Description
java.base
Defines the foundational APIs of the Java SE Platform.
java.compiler
Defines the Language Model, Annotation Processing, and Java Compiler APIs.
java.datatransfer
Defines the API for transferring data between and within applications.
java.desktop
Defines the AWT and Swing user interface toolkits, plus APIs for accessibility,
java.instrument
Defines services that allow agents to instrument programs running on the JVM
java.logging
Defines the Java Logging API.
java.management
Defines the Java Management Extensions (JMX) API

```java
public class AccountMain1 {
    Run | Debug
    public static void main(String[] args) {

        // 계좌 객체 생성  (한개의 고객계좌가 생성이 됐다고 생각하면 된다)
        Account account1 = new Account(accountNumber:"1111", name:"일길동", pwd:"aaaa", balance:10000);
        System.out.println(account1);

        System.out.println(account1.toString());

        // 계좌 객체 생성  (한개의 고객계좌가 생성이 됐다고 생각하면 된다)
        Account account2 = new Account(accountNumber:"1111", name:"일길동", pwd:"aaaa", balance:10000);

        // 계좌 객체 생성  (한개의 고객계좌가 생성이 됐다고 생각하면 된다)
        Account account3 = new Account(accountNumber:"2222", name:"일길동", pwd:"aaaa", balance:10000);
    }

    if (account2 == account3) {
        System.out.println("레퍼런스가 같다");
    } else {
        System.out.println("레퍼런스가 다르다");


    }
    if (account2.equals(account3)) {    // 객체의 내용 비교
    System.out.println("동일한 계좌입니다");
    } else {
    System.out.println("다른 계좌입니다");
    }
```


```java
// 계좌 객체 생성  (한개의 고객계좌가 생성이 됐다고 생각하면 된다)
Account account1 = new Account("1111", "일길동", "aaaa", 10000);
System.out.println(account1);

System.out.println(account1.toString());
```



Module java.base
Package java.lang

Class Object

java.lang.Object

```
public class Object
```

Class `Object` is the root of the class hierarchy. Every class has `Object` as a superclass. All objects, including arrays, implement the methods of this class.

Since:

1.0

See Also:

`Class`

Constructor Summary

Constructors

Constructor	Description
`Object()`	Constructs a new object.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods** | **Deprecated Methods**

Modifier and Type	Method	Description
`protected Object`	`clone()`	Creates and returns a copy of this object.
`boolean`	`equals(Object obj)`	Indicates whether some other object is "equal to" this one.
`protected void`	`finalize()`	**Deprecated.** The finalization mechanism is inherently problematic.
`final Class<?>`	`getClass()`	Returns the runtime class of this `Object`.
`int`	`hashCode()`	Returns a hash code value for the object.

https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/package-summary.html
Overview> java.base> java.lang> All Classes and Interfaces> Object

2024.12.03(화)

인터페이스 (interface)

· 기능 명세서이다.

· 모든 기능을 추상화로 정의만 하고 구현은 하지 않은것이다.

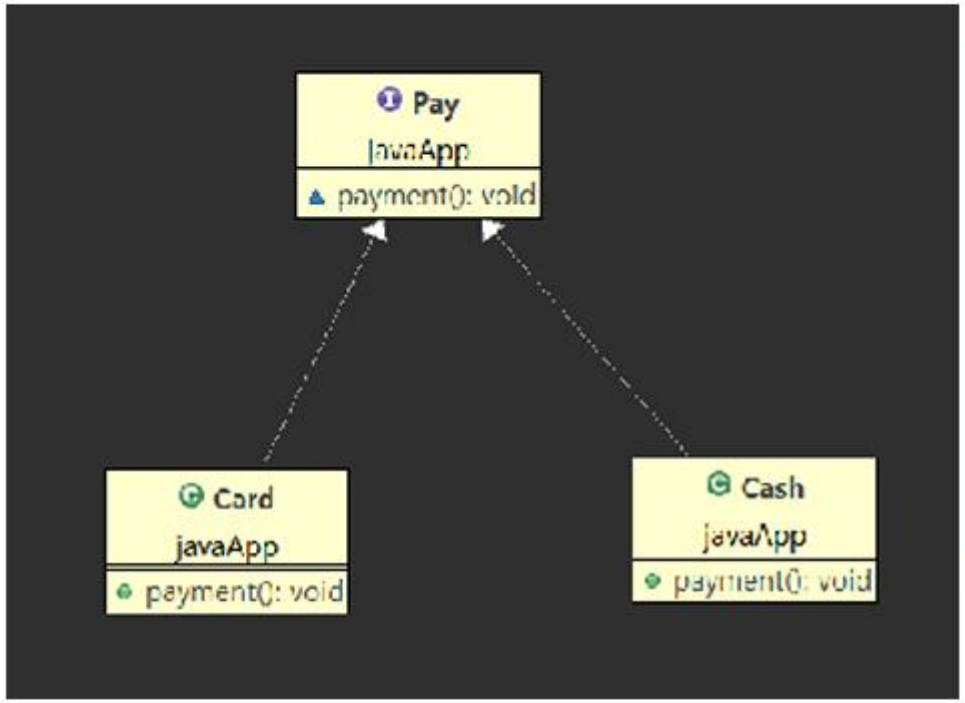
● 인터페이스 (interface)를 사용하는 이유

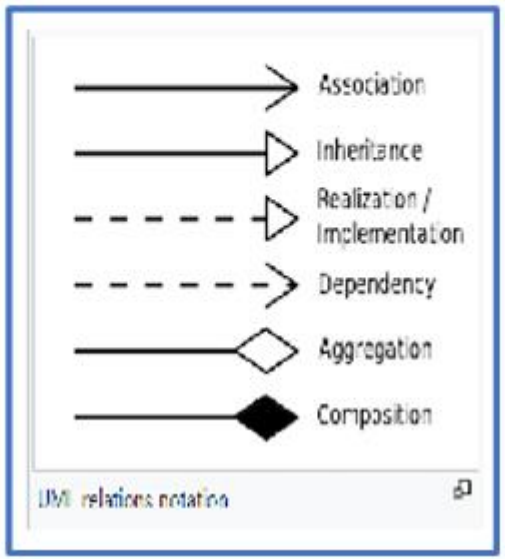
1. 협업을 할때 미리 인터페이스를 작성함으로써 메소드를 정할 수 있다.

2. 클래스간 결합도(코드 종속성)를 낮춘 유연한 방식의 프로그래밍이 가능해진다.

3. 자바에서는 다중 상속을 구현하기 위해서 사용한다.

● UML 클래스 다이어그램





※ Realization 관계

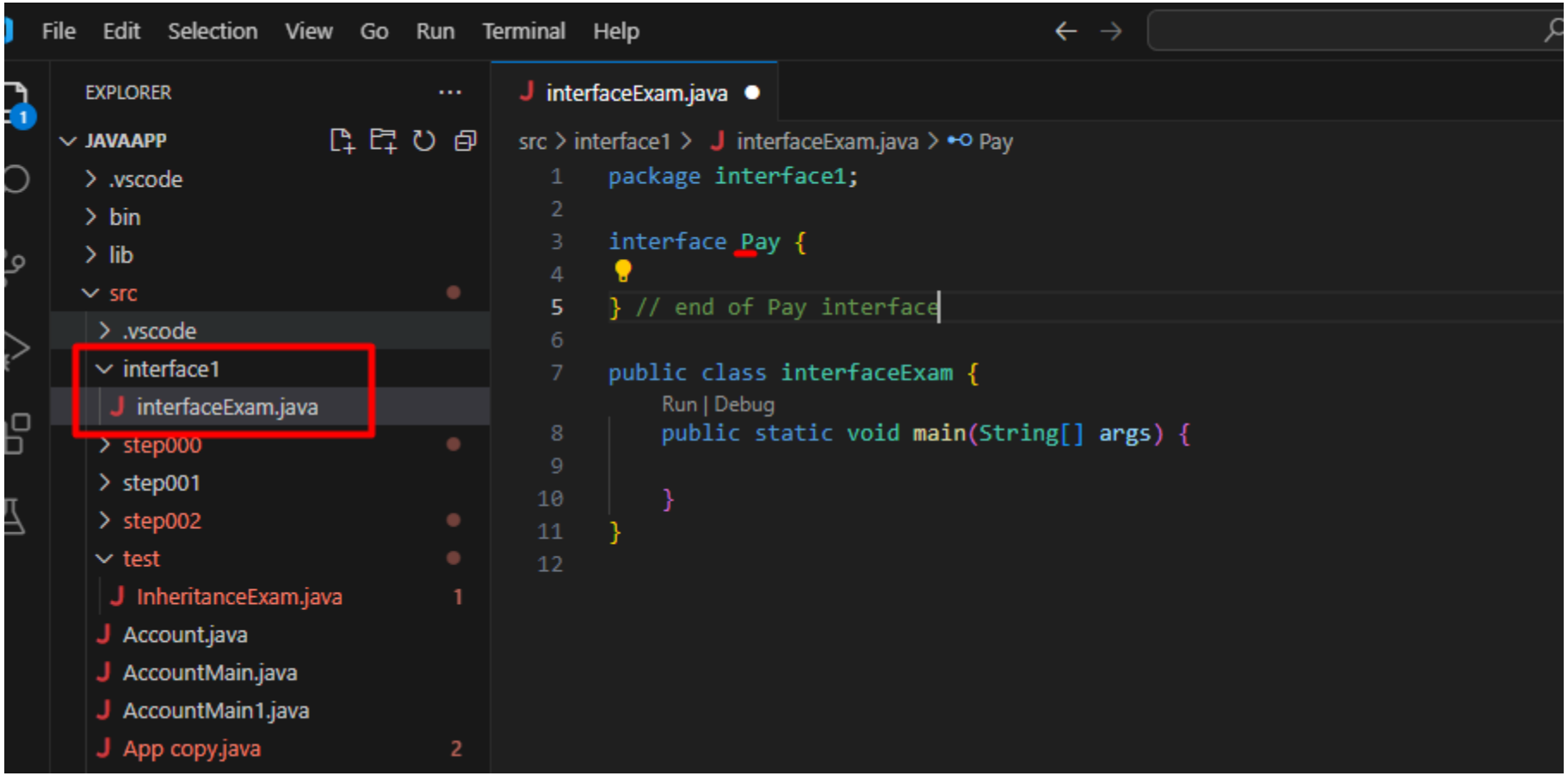
· 인터페이스의 Spec(명세, 정의) 만 있는 메소드를 오버라이딩하여 실제 기능으로 구현하는 것을 말한다.

첫글자 대문자 원칙

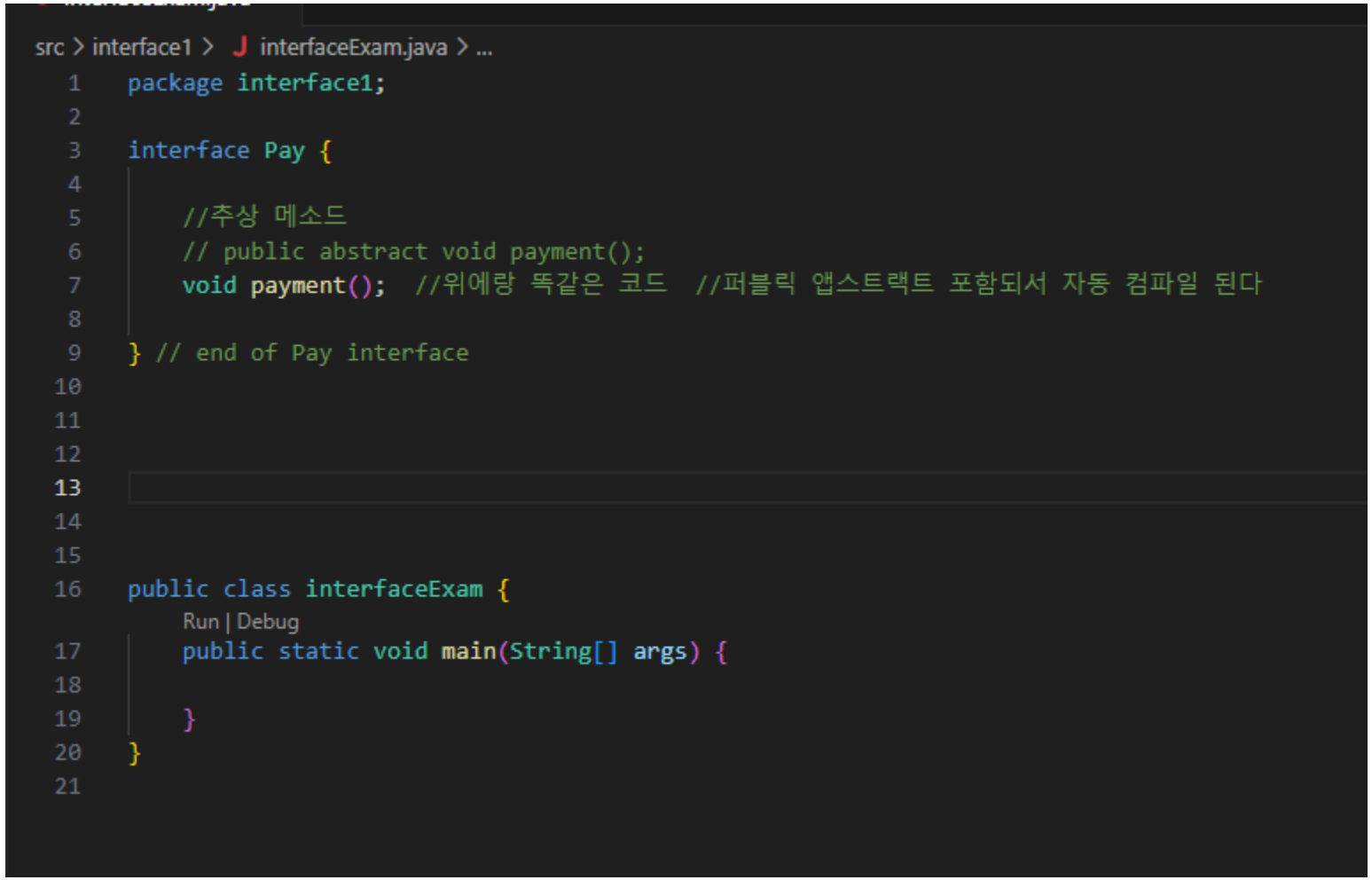
문법

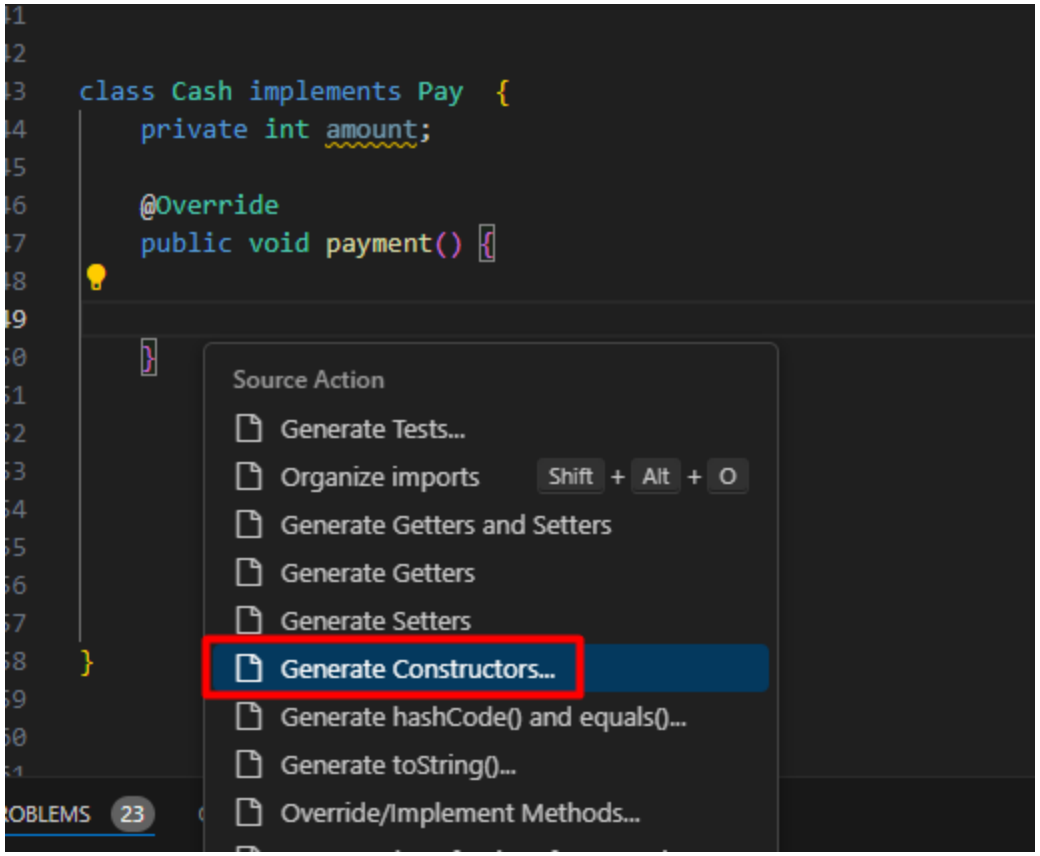
```
접근제어자  interface  인터페이스이름 {

     //상수

    // abstract 메소드

    // default 메소드  (Java8 추가)

    // static  메소드  (Java8 추가)

}
```

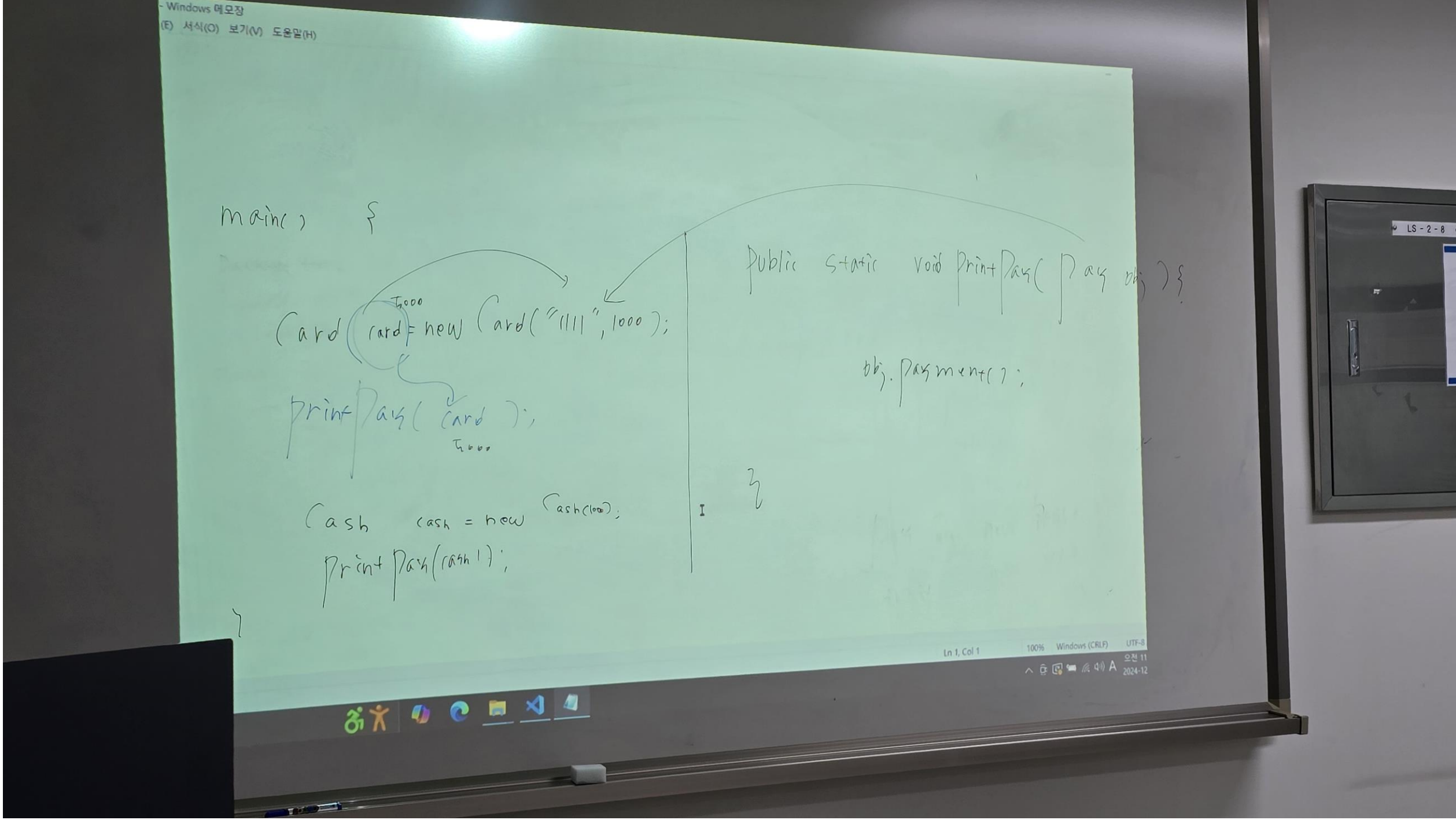
인터페이스 패키지는 소문자로 만드



메인 함수에서 퍼블릭을 썼기때문에 인터페이스 pay 에서 퍼블릭을 쓰면 에러난다

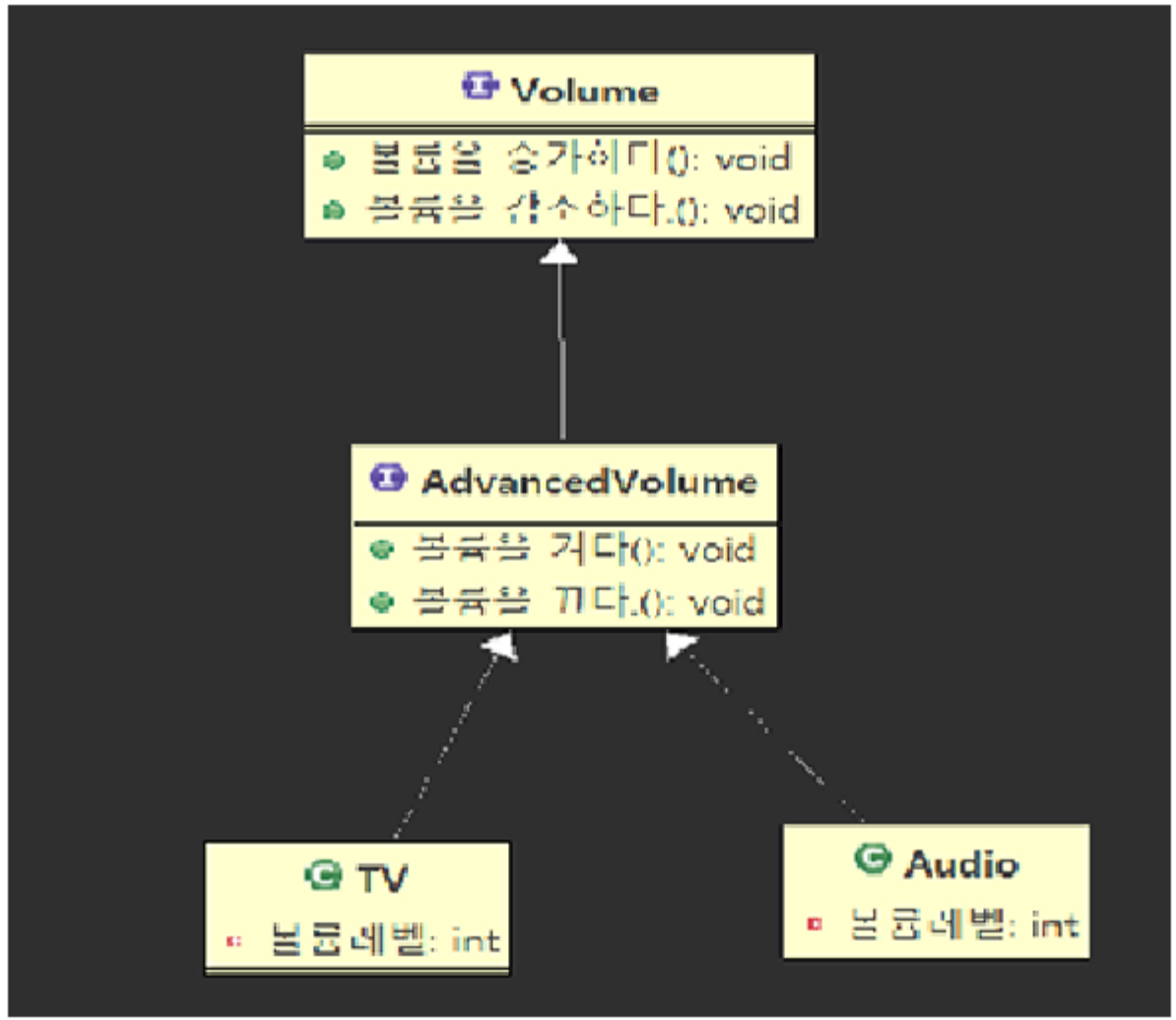

src > interface1 > interfaceExam.java > ...
1 package interface1;
2
3 interface Pay {
4
5 //추상 메소드
6 // public abstract void payment();
7 void payment(); //위에랑 똑같은 코드 //퍼블릭 앱스트랙트 포함되서 자동 컴파일 된다
8
9 } // end of Pay interface
10
11
12
13
14
15
16 public class interfaceExam {
Run | Debug
17 public static void main(String[] args) {
18
19 }
20 }
21


class Cash implements Pay {
private int amount;
@Override
public void payment() {
}
}
Source Action
Generate Tests...
Organize imports Shift + Alt + O
Generate Getters and Setters
Generate Getters
Generate Setters
Generate Constructors...
Generate hashCode() and equals()...
Generate toString()...
Override/Implement Methods...
OBLEMS 23

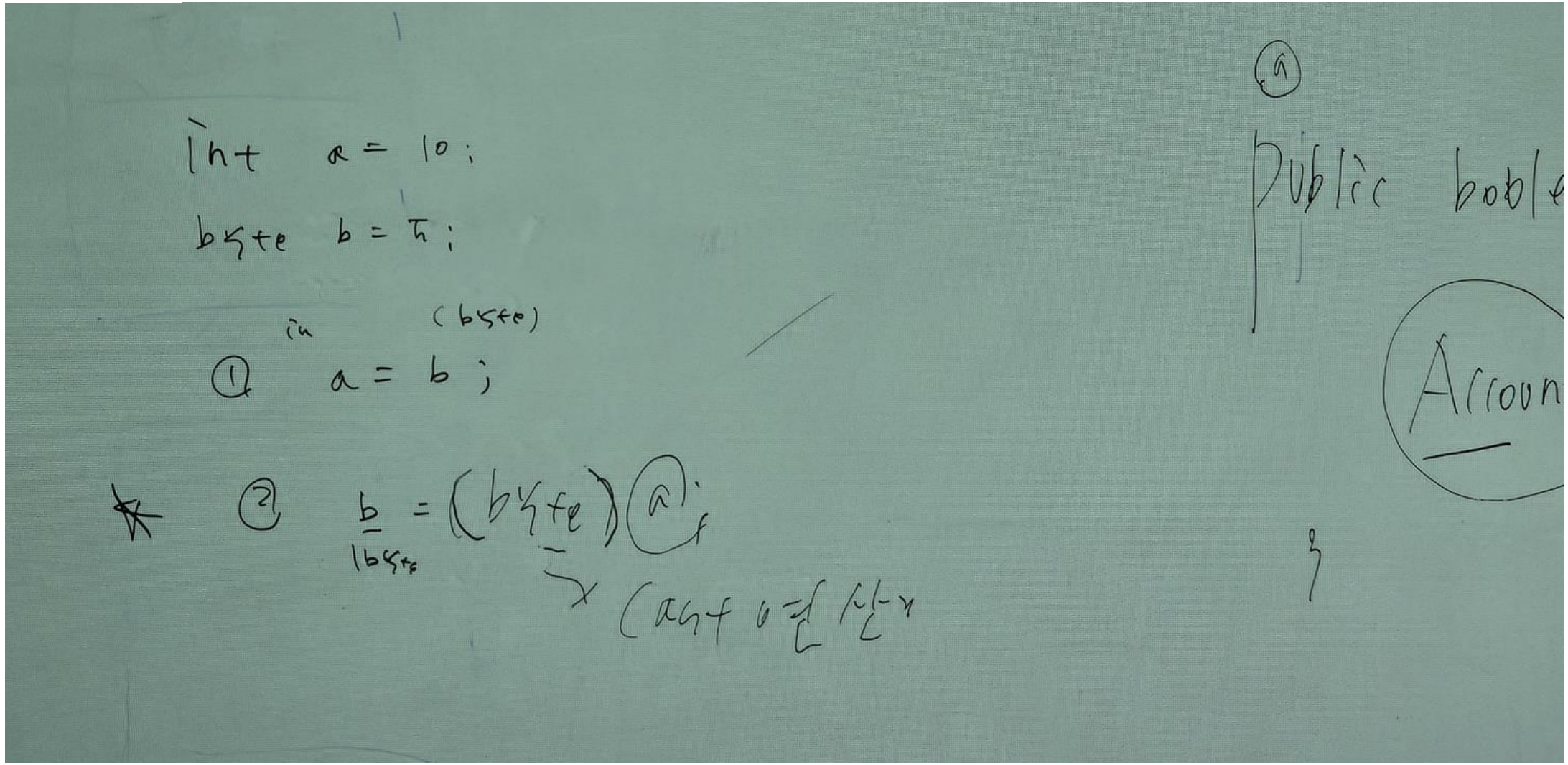

- Windows 메모장
(E) 서식(O) 보기(V) 도움말(H)
main() {
Card card = new Card("1111", 1000);
printPay(card);
Cash cash = new Cash(1000);
printPay(cash);
}
public static void printPay(Pay obj) {
obj.payment();
}
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8
오전 11
2024-12

```
Pay card = new Card("1111",  10000);
card.payment();


Pay cash = new Cash(10000);
cash.payment();
```

● 인터페이스 간의 상속


Volume
볼륨을 증가하다(): void
볼륨을 감소하다(): void
AdvancedVolume
볼륨을 거다(): void
볼륨을 끄다(): void
TV
볼륨레벨: int
Audio
볼륨레벨: int

캐스팅



Object obj = new Car();

④

public boolean equals(Object obj) {

Account a1 = (Account) obj;

↳ 비교하고자 하는 객체

↳ 다운캐스팅

(Car

a1.

}

2. 업캐스팅 UpCasting 이란

업캐스팅이란, 클래스의 상속관계가 아래과 같은 상황에서

```
Parent
  |
Child
```

Parent형의 객체를 생성하고자 할 때, Child형의 정보를 좌변에 제공하는 것이다.

```
Parent p = new Child();
```

상위클래스인 좌변이 요구하는 정보는 그의 하위클래스인 우변은 당연히 전부 가지고있기 때문에 위 문장은 옳은 문장이 된다.

말하자면 업캐스팅이란, 하위 클래스의 정보를 담을 수 있는 객체에 상위클래스의 자료형을 부여해서, 상위클래스처럼 사용하게 하는 것이다.

3. 다운캐스팅 DownCasting 이란

그렇다면 다운캐스팅이란 뭘까? 단순히 업캐스팅의 반대라고 생각하면 될까?

그렇지 않다.

다운캐스팅이란 하위클래스(Child)의 정보를 담을 수 있는 객체의 자료형이 상위클래스(Parent)로 전환되어 있던 것(업캐스팅된 객체)을 다시 되돌리는 것을 의미한다. 업캐스팅 되었던 객체의 자료형을 다시 하위클래스의 정보를 담는 기능을 하도록 자료형을 Child로 바꾸어서 되돌려 놓는 것을 말한다.

```
Parent p = new Child(); //업캐스팅 -p는 Parent형.
Child c = (Child) p; //다운캐스팅! -p는 Child형.
```

원칙적으로 다운캐스팅 혼자만 쓰면 우변이 좌변에서 필요한 정보를 모두 채워주지 못하기 때문에 불가능한 문장이 되고, 꼭 업캐스팅이 선행되어야 한다는 것을 명심해야한다. 아래는 에러가 나는 코드이다.

```
Parent p = new Parent();
Child c = (Child) p;//에러!!
```

예외(Exception)

● 예외(Exception) 란?

· 프로그램을 실행하는 과정에서 발생하는 예상치 못한 에러를 말한다.

· 네트워크 연결에 실패한 경우

· 파일이 존재하지 않는 경우

· 데이터베이스 연결에 실패한 경우

예외(Exception)

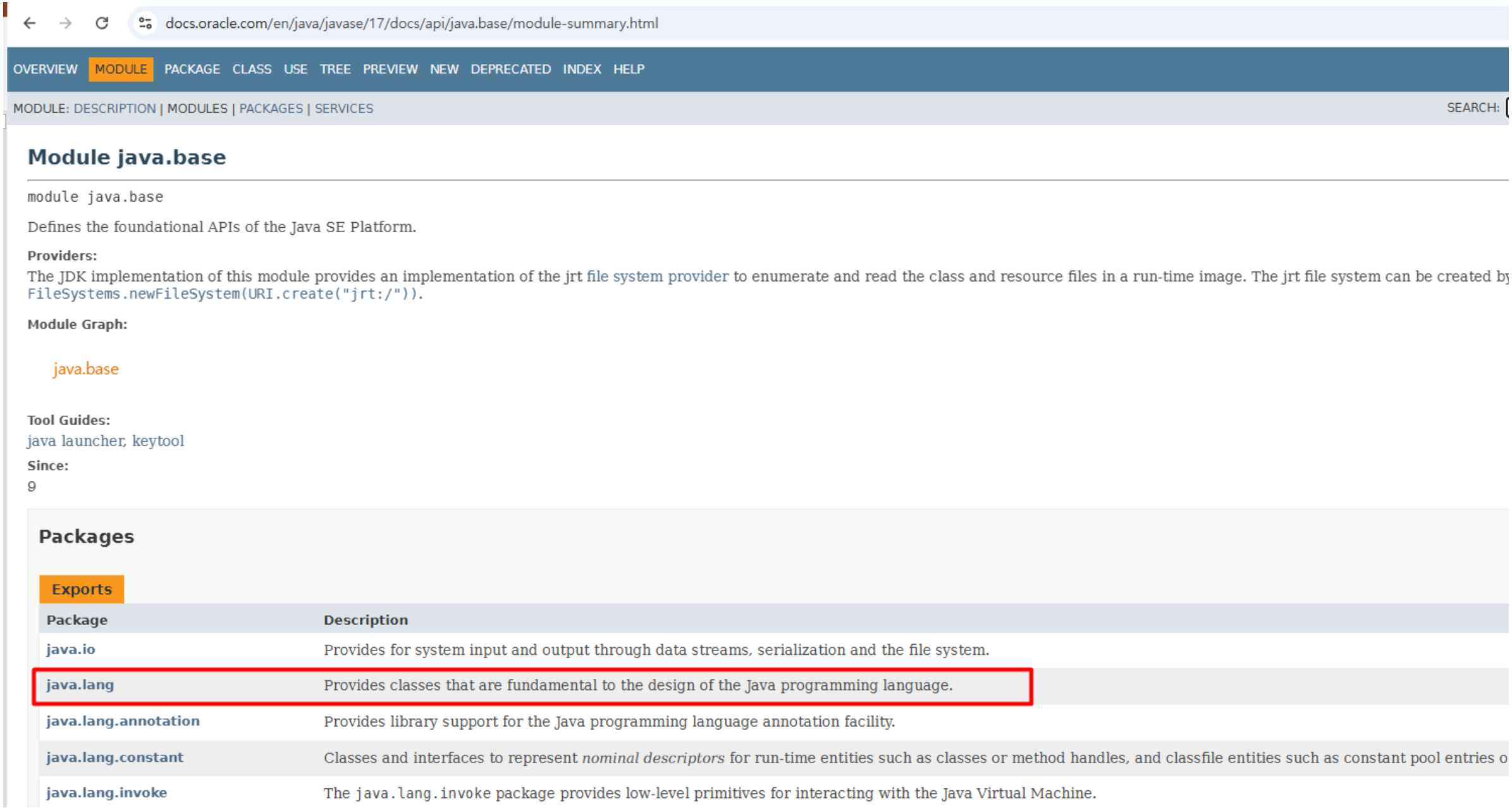
● 예외(Exception) 란?

· 프로그램을 실행하는 과정에서 발생하는 예상치 못한 에러를 말한다.

· 네트워크 연결에 실패한 경우

· 파일이 존재하지 않는 경우

· 데이터베이스 연결에 실패한 경우


docs.oracle.com/en/java/javase/17/docs/api/java.base/module-summary.html
OVERVIEW MODULE PACKAGE CLASS USE TREE PREVIEW NEW DEPRECATED INDEX HELP
MODULE: DESCRIPTION | MODULES | PACKAGES | SERVICES
SEARCH:
Module java.base
module java.base
Defines the foundational APIs of the Java SE Platform.
Providers:
The JDK implementation of this module provides an implementation of the jrt file system provider to enumerate and read the class and resource files in a run-time image. The jrt file system can be created by
FileSystems.newFileSystem(URI.create("jrt:/")).
Module Graph:
java.base
Tool Guides:
java launcher, keytool
Since:
9
Packages
Exports
Package
Description
java.io
Provides for system input and output through data streams, serialization and the file system.
java.lang
Provides classes that are fundamental to the design of the Java programming language.
java.lang.annotation
Provides library support for the Java programming language annotation facility.
java.lang.constant
Classes and interfaces to represent nominal descriptors for run-time entities such as classes or method handles, and classfile entities such as constant pool entries o
java.lang.invoke
The java.lang.invoke package provides low-level primitives for interacting with the Java Virtual Machine.

undsException	Thrown by `String` methods to indicate that an index is either negative or greater than the size of the string.
	Indicates that the named compiler warnings should be suppressed in the annotated element (and in all program elements contained in the annotated element).
	The `System` class contains several useful class fields and methods.
	`System.Logger` instances log messages that will be routed to the underlying logging framework the `LoggerFinder` uses.
l	System loggers levels.
er	The `LoggerFinder` service is responsible for creating, managing, and configuring loggers to the underlying framework it uses.
	A *thread* is a thread of execution in a program.
	A thread state.
ceptionHandler	Interface for handlers invoked when a `Thread` abruptly terminates due to an uncaught exception.
	An instance of `ThreadDeath` is thrown in the victim thread when the (deprecated) `Thread.stop()` method is invoked.
	A thread group represents a set of threads.
	This class provides thread-local variables.
	The `Throwable` class is the superclass of all errors and exceptions in the Java language.
eption	Thrown when an application tries to access a type using a string representing the type's name, but no definition for the type with the specified name can be fou
	Thrown when an unknown but serious exception has occurred in the Java Virtual Machine.
r	Thrown if the Java Virtual Machine cannot find an appropriate native-language definition of a method declared `native`.
ersionError	Thrown when the Java Virtual Machine attempts to read a class file and determines that the major and minor version numbers in the file are not supported.
ionException	Thrown to indicate that the requested operation is not supported.

docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Throwable.html

Module java.base
Package java.lang

Class Throwable

java.lang.Object
    java.lang.Throwable

All Implemented Interfaces:

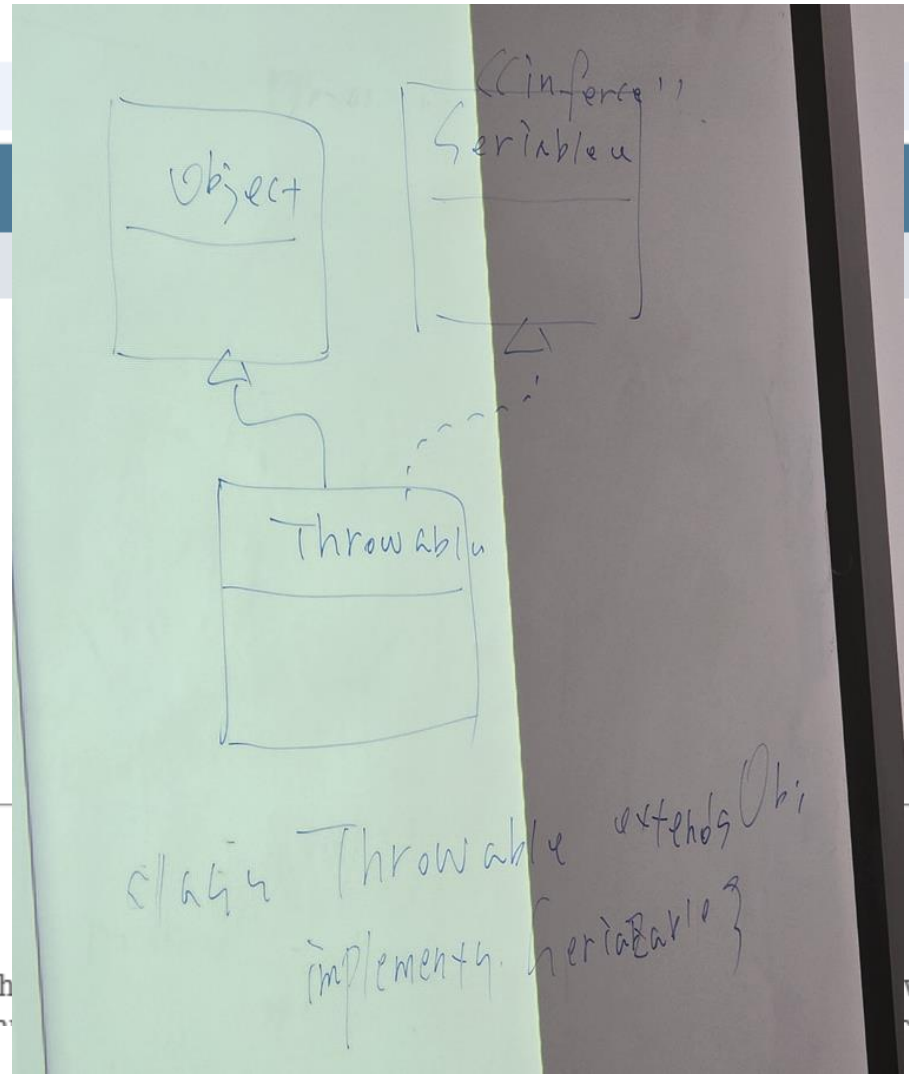
Serializable

Direct Known Subclasses:

Error, Exception

```
public class Throwable
extends Object
implements Serializable
```

The Throwable class is the superclass of all errors and exceptions in the Java language. Only objects th wn by th
this class or one of its subclasses can be the argument type in a catch clause. For the purposes of com of Throw



docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Throwable.html

OVERVIEW MODULE PACKAGE CLASS USE TREE PREVIEW NEW DEPRECATED INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Module java.base
Package java.lang

Class Throwable

java.lang.Object
    java.lang.Throwable

All Implemented Interfaces:

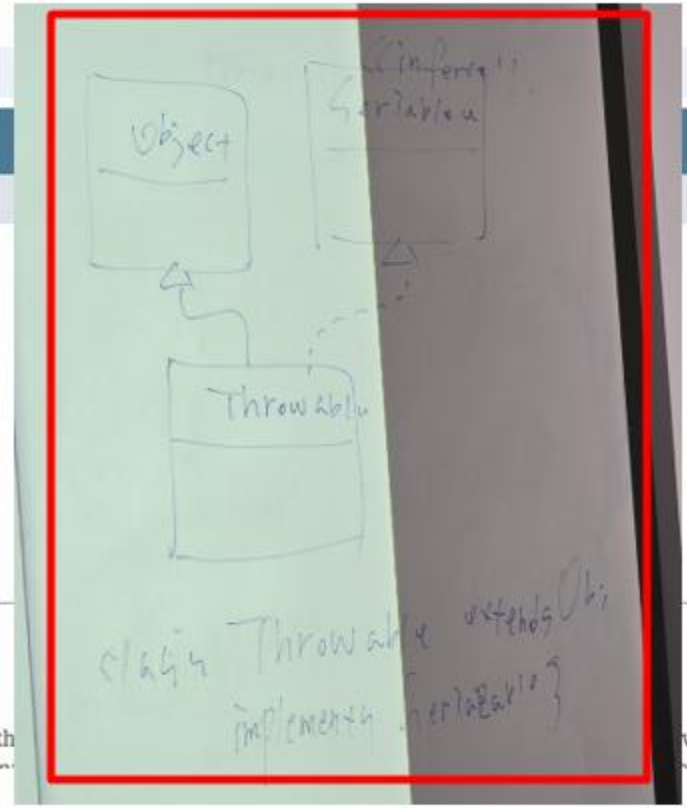
Serializable

Direct Known Subclasses:

Error, Exception

```
public class Throwable
extends Object
implements Serializable
```

The Throwable class is the superclass of all errors and exceptions in the Java language. Only objects th wn by th



Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
final void	addSuppressed(Throwable exception)	Appends the specified exception to the exceptions that were suppressed in order to deliver this exception.
Throwable	fillInStackTrace()	Fills in the execution stack trace.
Throwable	getCause()	Returns the cause of this throwable or null if the cause is nonexistent or unknown.
String	getLocalizedMessage()	Creates a localized description of this throwable.
String	getMessage()	Returns the detail message string of this throwable.
StackTraceElement[]	getStackTrace()	Provides programmatic access to the stack trace information printed by printStackTrace().
final Throwable[]	getSuppressed()	Returns an array containing all of the exceptions that were suppressed, typically by the try-with-resources statement, in order to deliver this exception.
Throwable	initCause(Throwable cause)	Initializes the *cause* of this throwable to the specified value.
void	printStackTrace()	Prints this throwable and its backtrace to the standard error stream.
void	printStackTrace(PrintStream s)	Prints this throwable and its backtrace to the specified print stream.
void	printStackTrace(PrintWriter s)	Prints this throwable and its backtrace to the specified print writer.
void	setStackTrace(StackTraceElement[] stackTrace)	Sets the stack trace elements that will be returned by getStackTrace() and printed by printStackTrace() and related methods.
String	toString()	Returns a short description of this throwable.

언어 감지 영어 한국어 독일어

void
printStackTrace()
Prints this throwable and its backtrace to the standard error stream.
void
printStackTrace(PrintStream s)
Prints this throwable and its backtrace to the specified print stream.
void
printStackTrace(PrintWriter s)
Prints this throwable and its backtrace to the specified print writer.

306 / 5,000

한국어 영어 일본어

void
printStackTrace()
이 throwable과 그 백트레이스를 표준 오류 스트림에 인쇄합니다.
void
printStackTrace(PrintStream s)
이 throwable과 그 백트레이스를 지정된 인쇄 스트림에 인쇄합니다.
void
printStackTrace(PrintWriter s)
이 throwable과 그 백트레이스를 지정된 인쇄 작성기에 인쇄합니다.

void
printStackTrace()
i throwablegwa geu baegteuleiseuleul pyojun olyu seuteulim-e inswaehabnida.
자세히

의견 보내기

entInitializationException, AgentLoadException, AlreadyBoundException, AttachNotSupportedException, AWTException, BackingStoreException, BadAttributeValueExpE
gOperationException, BrokenBarrierException, CardException, CertificateException, ClassNotLoadedException, CloneNotSupportedException, DataFormatException, Dat
ionControl.ExecutionControlException, ExecutionException, ExpandVetoException, FontFormatException, GeneralSecurityException, GSSException, IllegalClassFormat
ption, IncompatibleThreadStateException, InterruptedException, IntrospectionException, InvalidApplicationException, InvalidMidiDataException, InvalidPreferenc
ion, InvalidTypeException, InvocationException, IOException, JMException, JShellException, KeySelectorException, LambdaConversionException, LineUnavailableExcep
TypeParseException, NamingException, NoninvertibleTransformException, NotBoundException, ParseException, ParserConfigurationException, PrinterException, PrintE
iveOperationException, RefreshFailedException, RuntimeException, SAXException, ScriptException, ServerNotActiveException, SQLException, StringConcatException, T
nsformerException, TransformException, UnmodifiableClassException, UnsupportedAudioFileException, UnsupportedCallbackException, UnsupportedFlavorException, Un
axException, VMStartException, XAException, XMLParseException, XMLSignatureException, XMLStreamException, XPathException

---

sses are a form of Throwable that indicates conditions that a reasonable application might want to catch.

lasses that are not also subclasses of RuntimeException are *checked exceptions*. Checked exceptions need to be declared in a method or constructor's throws clause if they can be
the method or constructor boundary.

ceptions

itializationException, AgentLoadException, AlreadyBoundException, AttachNotSupportedException, AWTException, BackingStoreException, BadAttributeValueExpExcept
ationException, BrokenBarrierException, CardException, CertificateException, ClassNotLoadedException, CloneNotSupportedException, DataFormatException, Datatype
ntrol.ExecutionControlException, ExecutionException, ExpandVetoException, FontFormatException, GeneralSecurityException, GSSException, IllegalClassFormatExcep
IncompatibleThreadStateException, InterruptedException, IntrospectionException, InvalidApplicationException, InvalidMidiDataException, InvalidPreferencesFor
nvalidTypeException, InvocationException, IOException, JMException, JShellException, KeySelectorException, LambdaConversionException, LineUnavailableException,
arseException, NamingException, NoninvertibleTransformException, NotBoundException, ParseException, ParserConfigurationException, PrinterException, PrintExcept
erationException, RefreshFailedException, RuntimeException, SAXException, ScriptException, ServerNotActiveException, SQLException, StringConcatException, Timeou
merException, TransformException, UnmodifiableClassException, UnsupportedAudioFileException, UnsupportedCallbackException, UnsupportedFlavorException, Unsuppo
eption, VMStartException, XAException, XMLParseException, XMLSignatureException, XMLStreamException, XPathException

---

re a form of Throwable that indicates conditions that a reasonable application might want to catch.

that are not also subclasses of RuntimeException are *checked exceptions*. Checked exceptions need to be declared in a method or constructor's throws clause if they can be throw
ethod or constructor boundary.

ns

ithmeticException, ArrayStoreException, BufferOverflowException, BufferUnderflowException, CannotRedoException, CannotUndoException, CatalogException, ClassCas
tion, CompletionException, ConcurrentModificationException, DateTimeException, DOMException, DuplicateRequestException, EmptyStackException, EnumConstantNotPre
ileSystemNotFoundException, FindException, IllegalArgumentException, IllegalCallerException, IllegalMonitorStateException, IllegalPathStateException, IllegalS
Exception, InaccessibleObjectException, IncompleteAnnotationException, InconsistentDebugInfoException, IndexOutOfBoundsException, InternalException, InvalidCo
ModuleDescriptorException, InvalidModuleException, InvalidRequestStateException, InvalidStackFrameException, JarSignerException, JMRuntimeException, JSExcepti
TypeException, MalformedParametersException, MirroredTypesException, MissingResourceException, NativeMethodException, NegativeArraySizeException, NoSuchDynami
nismException, NullPointerException, ObjectCollectedException, ProfileDataException, ProviderException, ProviderNotFoundException, RangeException, RasterFormat
ionException, SecurityException, SPIResolutionException, TypeNotPresentException, UncheckedIOException, UndeclaredThrowableException, UnknownEntityException, U
edOperationException, VMDisconnectedException, VMMismatchException, VMOutOfMemoryException, WrongMethodTypeException, XPathException

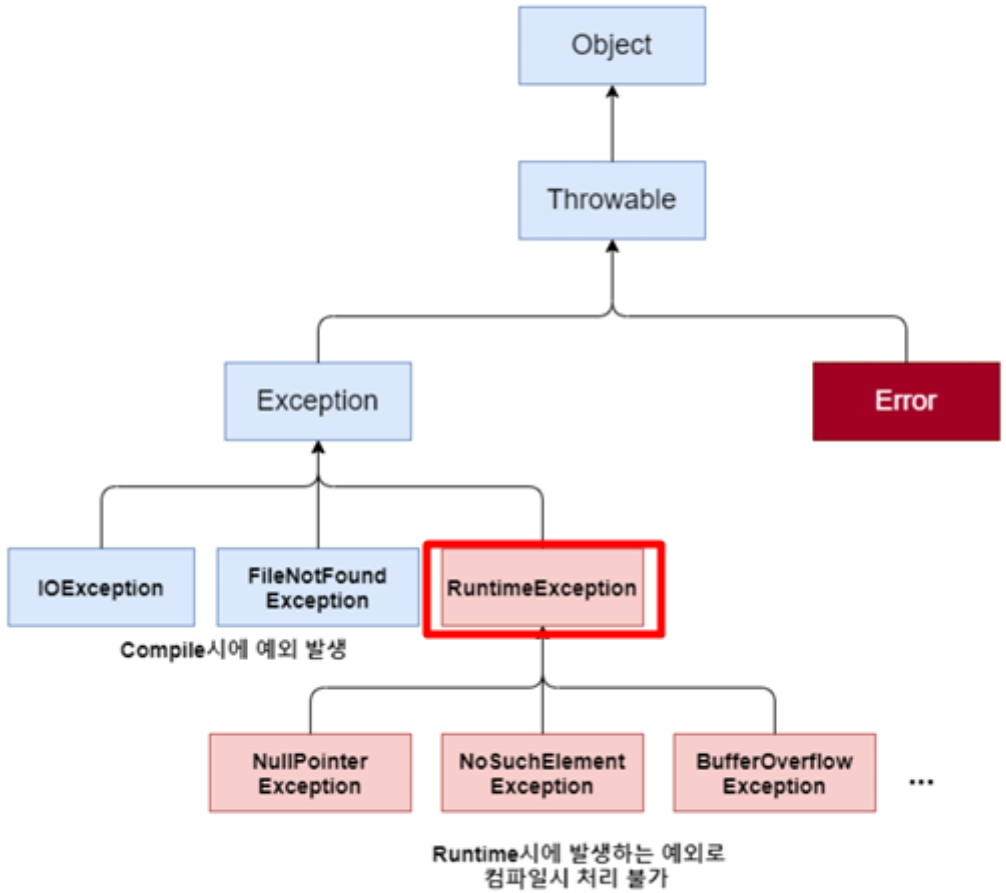
---

hose exceptions that can be thrown during the normal operation of the Java Virtual Machine.

e *unchecked exceptions*. Unchecked exceptions do *not* need to be declared in a method or constructor's `throws` clause if they can be thrown by the execution of the method or cons

ons

● Exception 클래스 계층도



2. throws 구문

```
접근지정자 [static] 리턴타입 메소드명 ( ) throws Exception 클래스이름 {
        throw new Exception 클래스이름( );
}
```

2. throws 구문

public static void main()

```
접근지정자 [static] 리턴타입 메소드명 () throws Exception 클래스이름 {

        throw new Exception 클래스이름();

}
```

IO

Module java.base

`module java.base`

Defines the foundational APIs of the Java SE Platform.

Providers:

The JDK implementation of this module provides an implementation of the jrt file system provider to enumerate and read the class and resource files in a run-time image. The j
`FileSystems.newFileSystem(URI.create("jrt:/"))`.

Module Graph:

java.base

Tool Guides:

java launcher, keytool

Since:

9

Packages

Exports

Package	Description
java.io	Provides for system input and output through data streams, serialization and the file system.
java.lang	Provides classes that are fundamental to the design of the Java programming language.
java.lang.annotation	Provides library support for the Java programming language annotation facility.

Module java.base

Package java.io

`package java.io`

Provides for system input and output through data streams, serialization and the file system. Unless otherwise noted, passing a null argument t
`NullPointerException` to be thrown.

Object Serialization

**Warning: Deserialization of untrusted data is inherently dangerous and should be avoided. Untrusted data should be carefully vali
Secure Coding Guidelines for Java SE.**

- *Java Object Serialization Specification*
- Serial Filtering best practices
- The serialver tool

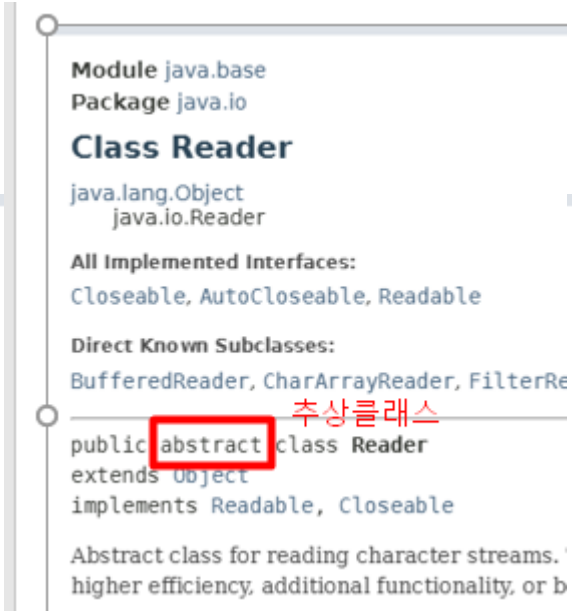
Since:

1.0

All Classes and Interfaces	Interfaces	Classes	Enum Classes	Exceptions	Errors	Annotation Interfaces

Class	Description
BufferedInputStream	A `BufferedInputStream` adds functionality to another input stream-namely, the ability to buffer the input an
BufferedOutputStream	The class implements a buffered output stream.
BufferedReader	Reads text from a character-input stream, buffering characters so as to provide for the efficient reading of cl
BufferedWriter	Writes text to a character-output stream, buffering characters so as to provide for the efficient writing of sin
ByteArrayInputStream	A `ByteArrayInputStream` contains an internal buffer that contains bytes that may be read from the stream.
ByteArrayOutputStream	This class implements an output stream in which the data is written into a byte array.
CharArrayReader	This class implements a character buffer that can be used as a character-input stream.
CharArrayWriter	This class implements a character buffer that can be used as an Writer.
CharConversionException	Base class for character conversion exceptions

OutputStream	This abstract class is the superclass of all classes representing an output stream of bytes.
OutputStreamWriter	An OutputStreamWriter is a bridge from character streams to byte streams: Characters written to it are encoded into bytes using a spe
PipedInputStream	A piped input stream should be connected to a piped output stream; the piped input stream then provides whatever data bytes are writ
PipedOutputStream	A piped output stream can be connected to a piped input stream to create a communications pipe.
PipedReader	Piped character-input streams.
PipedWriter	Piped character-output streams.
PrintStream	A `PrintStream` adds functionality to another output stream, namely the ability to print representations of various data values convenie
PrintWriter	Prints formatted representations of objects to a text-output stream.
PushbackInputStream	A `PushbackInputStream` adds functionality to another input stream, namely the ability to "push back" or "unread" bytes, by storing pus
PushbackReader	A character-stream reader that allows characters to be pushed back into the stream.
RandomAccessFile	Instances of this class support both reading and writing to a random access file.
Reader	Abstract class for reading character streams.
SequenceInputStream	A `SequenceInputStream` represents the logical concatenation of other input streams.
Serial	Indicates that an annotated field or method is part of the serialization mechanism defined by the *Java Object Serialization Specification*



Module java.base
Package java.io

Class Reader

java.lang.Object
    java.io.Reader

All Implemented Interfaces:

Closeable, AutoCloseable, Readable

Direct Known Subclasses:

BufferedReader, CharArrayReader, FilterReader, InputStreamReader, PipedReader, StringReader

---

```
public abstract class Reader
extends Object
implements Readable, Closeable
```

Abstract class for reading character streams. The only methods that a subclass must implement are read(char[], int, int) and close(). Most subclasses, however, will override higher efficiency, additional functionality, or both.

Since:

1.1

See Also:

BufferedReader, LineNumberReader, CharArrayReader, InputStreamReader, FileReader, FilterReader, PushbackReader, PipedReader, StringReader, Writer

mary

	Description
er in)	Creates a buffering character-input stream that uses a default-sized input buffer.
er in, int sz)	Creates a buffering character-input stream that uses an input buffer of the specified size.

y

nce Methods | Concrete Methods

Method	Description
close()	Closes the stream and releases any system resources associated with it.
lines()	Returns a Stream, the elements of which are lines read from this BufferedReader.
mark(int readAheadLimit)	Marks the present position in the stream.
markSupported()	Tells whether this stream supports the mark() operation, which it does.
read()	Reads a single character.
read(char[] cbuf, int off, int len)	Reads characters into a portion of an array.
readLine()	Reads a line of text.
ready()	Tells whether this stream is ready to be read.
reset()	Resets the stream to the most recent mark.

in class java.io.Reader

ead, skip, transferTo

e java.base
ge java.io

s Reader

ng.Object
a.io.Reader

lemented Interfaces:

ble, AutoCloseable, Readable

Known Subclasses:

redReader, CharArrayReader, FilterReader, InputStreamReader, PipedReader, StringReader

---

abstract class **Reader**
ds Object
ments Readable, Closeable

ct class for reading character streams. The only methods that a subclass must implement are read(char[], int, int) and close(). Most subclasses, however, will override some of the m
efficiency, additional functionality, or both.

o:

redReader, LineNumberReader, CharArrayReader, InputStreamReader, FileReader, FilterReader, PushbackReader, PipedReader, StringReader, Writer

d Summary

lds

ifier and Type	Field	Description

ader

StreamReader
ileReader

rfaces:

seable, Readable

Reader
amReader

racter files using a default buffer size. Decoding from bytes to characters uses either a specified charset or the platform's default charset.

meant for reading streams of characters. For reading streams of raw bytes, consider using a FileInputStream.

, FileInputStream

ry

d in class java.io.Reader

nmary

	Description
le)	Creates a new FileReader, given the File to read, using the platform's default charset.
criptor fd)	Creates a new FileReader, given the FileDescriptor to read, using the platform's default charset.
le, Charset charset)	Creates a new FileReader, given the File to read and the charset.
fileName)	Creates a new FileReader, given the name of the file to read, using the platform's default charset.
fileName, Charset charset)	Creates a new FileReader, given the name of the file to read and the charset.

ry

d in class java.io.InputStreamReader

, read, read, ready

d in class java.io.Reader

d, nullReader, read, read, reset, skip, transferTo

d in class java.lang.Object

alize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

ails

Constructor Summary

Constructors

Constructor	Description
`FileReader(File file)`	Creates a new FileReader, given the File to read, using the platform's default charset.
`FileReader(FileDescriptor fd)`	Creates a new FileReader, given the FileDescriptor to read, using the platform's default charset.
`FileReader(File file, Charset charset)`	Creates a new FileReader, given the File to read and the charset.
`FileReader(String fileName)`	Creates a new FileReader, given the name of the file to read, using the platform's default charset.
`FileReader(String fileName, Charset charset)`	Creates a new FileReader, given the name of the file to read and the charset.

Method Summary

Methods declared in class java.io.InputStreamReader

close, getEncoding, read, read, ready

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	close()	Closes the stream and releases any system resources associated with it.
Stream<String>	lines()	Returns a Stream, the elements of which are lines read from this BufferedReader.
void	mark(int readAheadLimit)	Marks the present position in the stream.
boolean	markSupported()	Tells whether this stream supports the mark() operation, which it does.
int	read()	Reads a single character.
int	read(char[] cbuf, int off, int len)	Reads characters into a portion of an array.
String	readLine()	Reads a line of text.
boolean	ready()	Tells whether this stream is ready to be read.
void	reset()	Resets the stream to the most recent mark.

Methods declared in class java.io.Reader

edReader

redReader

rfaces:

seable, Readable

sses:

---

eredReader

aracter-input stream, buffering characters so as to provide for the efficient reading of characters, arrays, and lines.

be specified, or the default size may be used. The default is large enough for most purposes.

d request made of a Reader causes a corresponding read request to be made of the underlying character or byte stream. It is therefore advisable to wrap a BufferedReader around
ostly, such as FileReaders and InputStreamReaders. For example,

```
n
Reader(new FileReader("foo.in"));
```

from the specified file. Without buffering, each invocation of read() or readLine() could cause bytes to be read from the file, converted into characters, and then returned, which ca

ataInputStreams for textual input can be localized by replacing each DataInputStream with an appropriate BufferedReader.

```
Line()
rows IOException
```

line is considered to be terminated by any one of a line feed ('\n'), a carriage return ('\r'), a carriage return followed immediately by a line feed, or by reaching the end-of-file (EOF)

e contents of the line, not including any line-termination characters, or null if the end of the stream has been reached without reading any characters

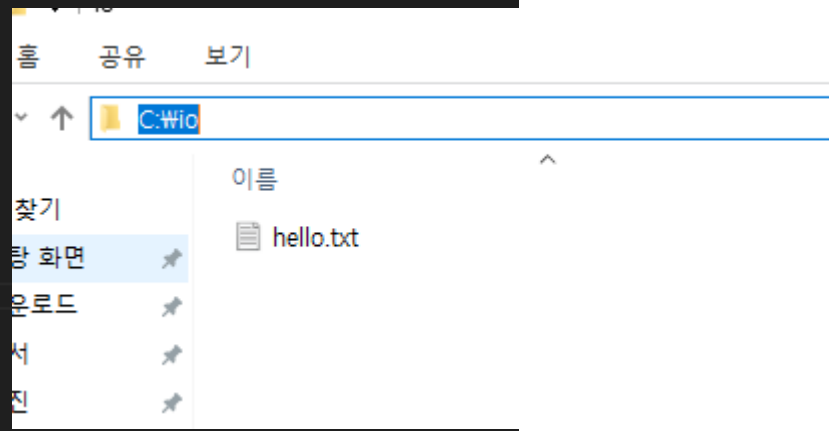
O error occurs

```
(java.nio.file.Path, java.nio.charset.Charset)
```

ExceptionExam1.java 2

src > exception > ExceptionExam1.java > ExceptionExam1 > main(String[])

```java
package exception;

import java.io.FileNotFoundException;
import java.io.FileReader;      //체크드인셉션

public class ExceptionExam1 {

    Run | Debug
    public static void main(String[] args) {

    try {

        // String fileName = "C:\\io\\hello.txt"; , "C:/io/hello.txt"; 둘중에 한가지 방식으로 선택 가능
        String fileName = "C:/a/hello.txt";
        FileReader fr = new FileReader(fileName);
    } catch (FileNotFoundException ex) {
        System.out.println(x:"file not exist!!");
    }

    System.out.println(x:"Exit!!");

    }

}
```

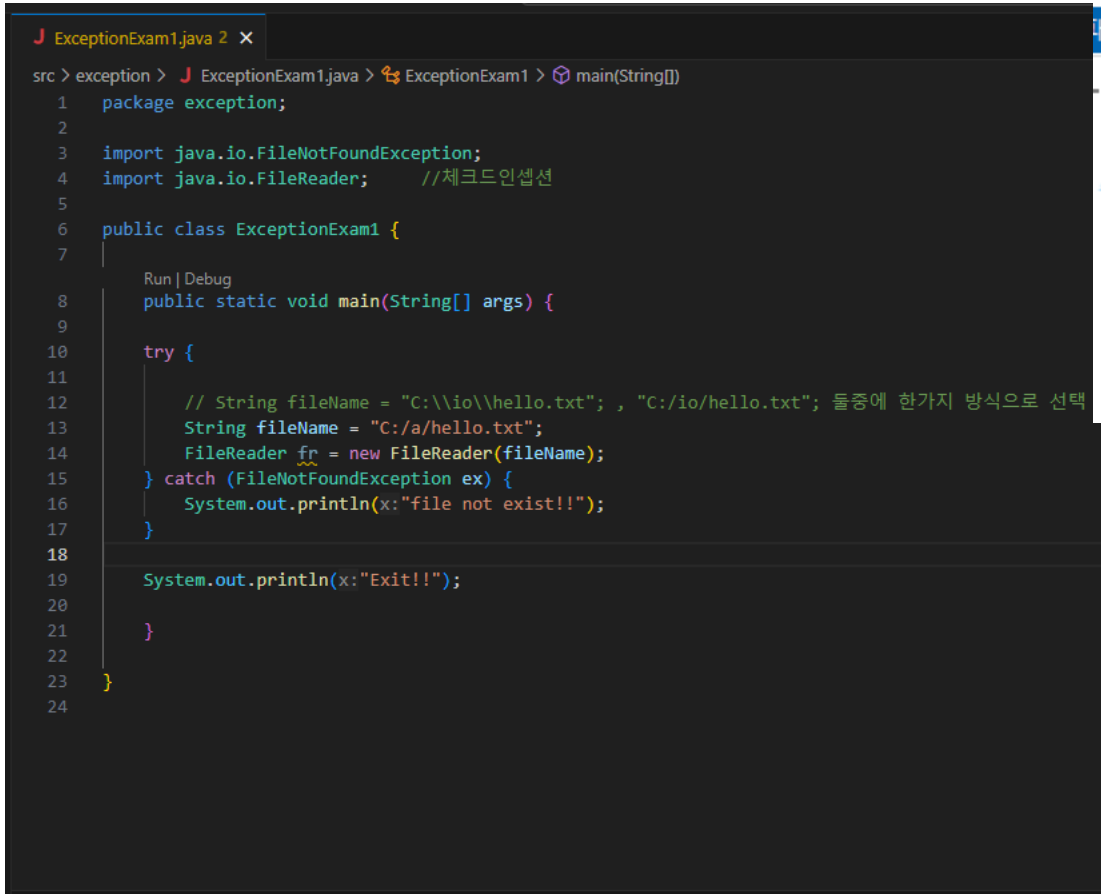


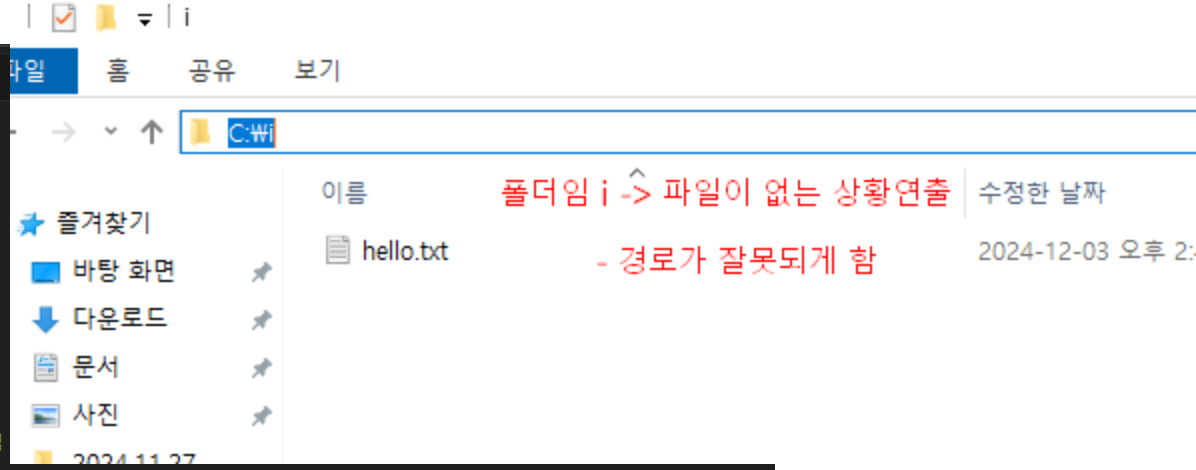
PROBLEMS 21 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\자바\javaApp>  & 'C:\Program Files\Amazon Corretto\jdk17.0.13_11\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:62275' '-XX:+ShowCodeDetai
App\bin' 'exception.ExceptionExam1'
Exit!!
PS C:\자바\javaApp> ^C
PS C:\자바\javaApp>
PS C:\자바\javaApp>  c:; cd 'c:\자바\javaApp'; & 'C:\Program Files\Amazon Corretto\jdk17.0.13_11\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:
sages' '-cp' 'C:\자바\javaApp\bin' 'exception.ExceptionExam1'
Exit!!
PS C:\자바\javaApp>
```

체크드인셉션은 반드시 예외처리가 필요하다

```
package exception;

import java.io.FileNotFoundException;
import java.io.FileReader;      //체크드인셉션

public class ExceptionExam1 {

    public static void main(String[] args) {

    try {

        // String fileName = "C:\\io\\hello.txt"; , "C:/io/hello.txt"; 둘중에 한가지 방
식으로 선택 가능
        String fileName = "C:/a/hello.txt";
        FileReader fr = new FileReader(fileName);
    } catch (FileNotFoundException ex) {
        System.out.println("file not exist!!");
    }

    System.out.println("Exit!!");

    }
```


ExceptionExam1.java 2
src > exception > ExceptionExam1.java > ExceptionExam1 > main(String[])
1 package exception;
2
3 import java.io.FileNotFoundException;
4 import java.io.FileReader; //체크드인셉션
5
6 public class ExceptionExam1 {
7
Run | Debug
8 public static void main(String[] args) {
9
10 try {
11
12 // String fileName = "C:\\io\\hello.txt"; , "C:/io/hello.txt"; 둘중에 한가지 방식으로 선택
13 String fileName = "C:/a/hello.txt";
14 FileReader fr = new FileReader(fileName);
15 } catch (FileNotFoundException ex) {
16 System.out.println(x:"file not exist!!");
17 }
18
19 System.out.println(x:"Exit!!");
20
21 }
22
23 }
24


파일
홈
공유
보기
C:₩i
이름
폴더임 i -> 파일이 없는 상황연출
수정한 날짜
즐겨찾기
바탕 화면
다운로드
문서
사진
hello.txt
- 경로가 잘못되게 함
2024-12-03 오후 2:

PROBLEMS 21
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\자바\javaApp> & 'C:\Program Files\Amazon Corretto\jdk17.0.13_11\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:62275' '-XX:+ShowCodeDetailsInExcepti
App\bin' 'exception.ExceptionExam1'
Exit!!
PS C:\자바\javaApp> ^C
PS C:\자바\javaApp>
PS C:\자바\javaApp> c:; cd 'c:\자바\javaApp'; & 'C:\Program Files\Amazon Corretto\jdk17.0.13_11\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:62290' '-XX
sages' '-cp' 'C:\자바\javaApp\bin' 'exception.ExceptionExam1'
Exit!!
PS C:\자바\javaApp> ^C
PS C:\자바\javaApp>
PS C:\자바\javaApp> c:; cd 'c:\자바\javaApp'; & 'C:\Program Files\Amazon Corretto\jdk17.0.13_11\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:62359' '-XX
sages' '-cp' 'C:\자바\javaApp\bin' 'exception.ExceptionExam1'
file not exist!!
Exit!!
PS C:\자바\javaApp>

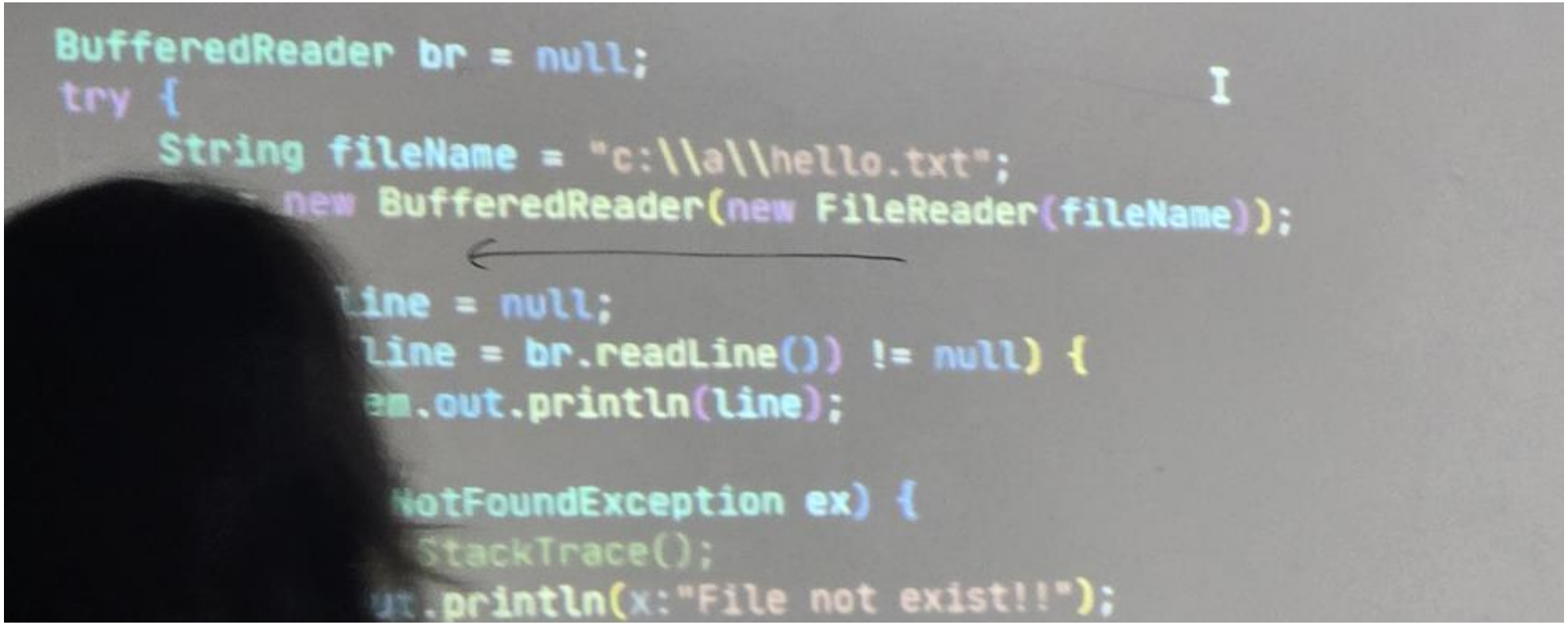
```
package exception;

import java.io.FileNotFoundException;
import java.io.FileReader;      //체크드인셉션

public class ExceptionExam1 {

    public static void main(String[] args) {
        FileReader fr = null;
    try {


        // String fileName = "C:\\io\\hello.txt"; , "C:/io/hello.txt"; 둘중에 한가지 방식으로 선택
능
        String fileName = "C:/a/hello.txt";
            fr = new FileReader(fileName);
    } catch (FileNotFoundException ex) {
        ex.printStackTrace();
        System.out.println("file not exist!!");
    } finally {
        try {
            fr.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
```

ExceptionExam1.java

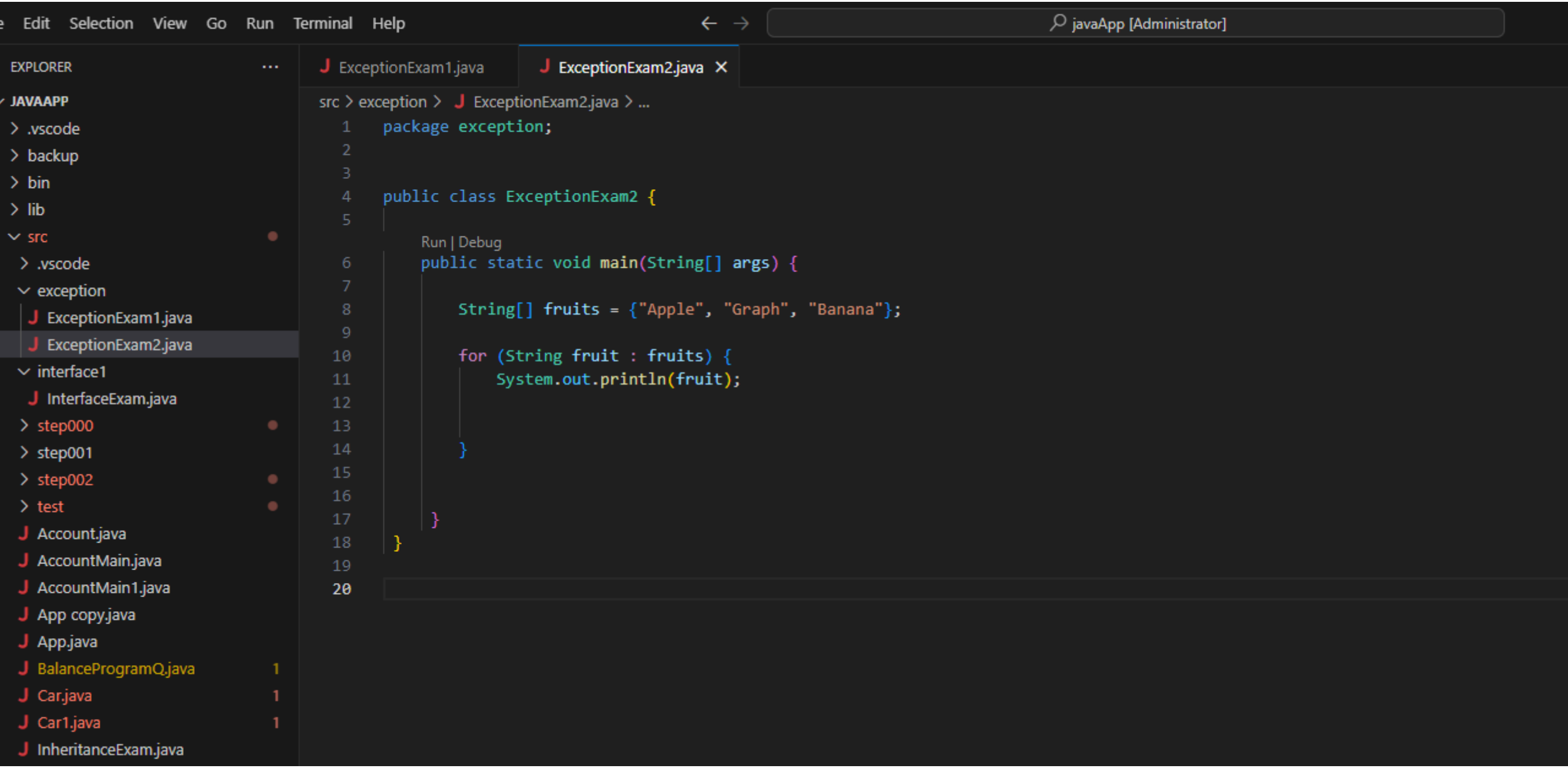
src > exception > ExceptionExam1.java > ...

```java
package exception;

import java.io.FileNotFoundException;
import java.io.FileReader;      //체크드인셉션

public class ExceptionExam1 {

    Run | Debug
    public static void main(String[] args) {
        FileReader fr = null;
    try {


        // String fileName = "C:\\io\\hello.txt"; , "C:/io/hello.txt"; 둘중에 한가지 방식으로 선택 가능
        String fileName = "C:/a/hello.txt";
            fr = new FileReader(fileName);
    } catch (FileNotFoundException ex) {
        ex.printStackTrace();
        System.out.println(x:"file not exist!!");
    } finally {
        try {
            fr.close();
        } catch (Exception e) {
            e.printStackTrace();
        }

    }

    System.out.println(x:"Exit!!");
        }
    }


```

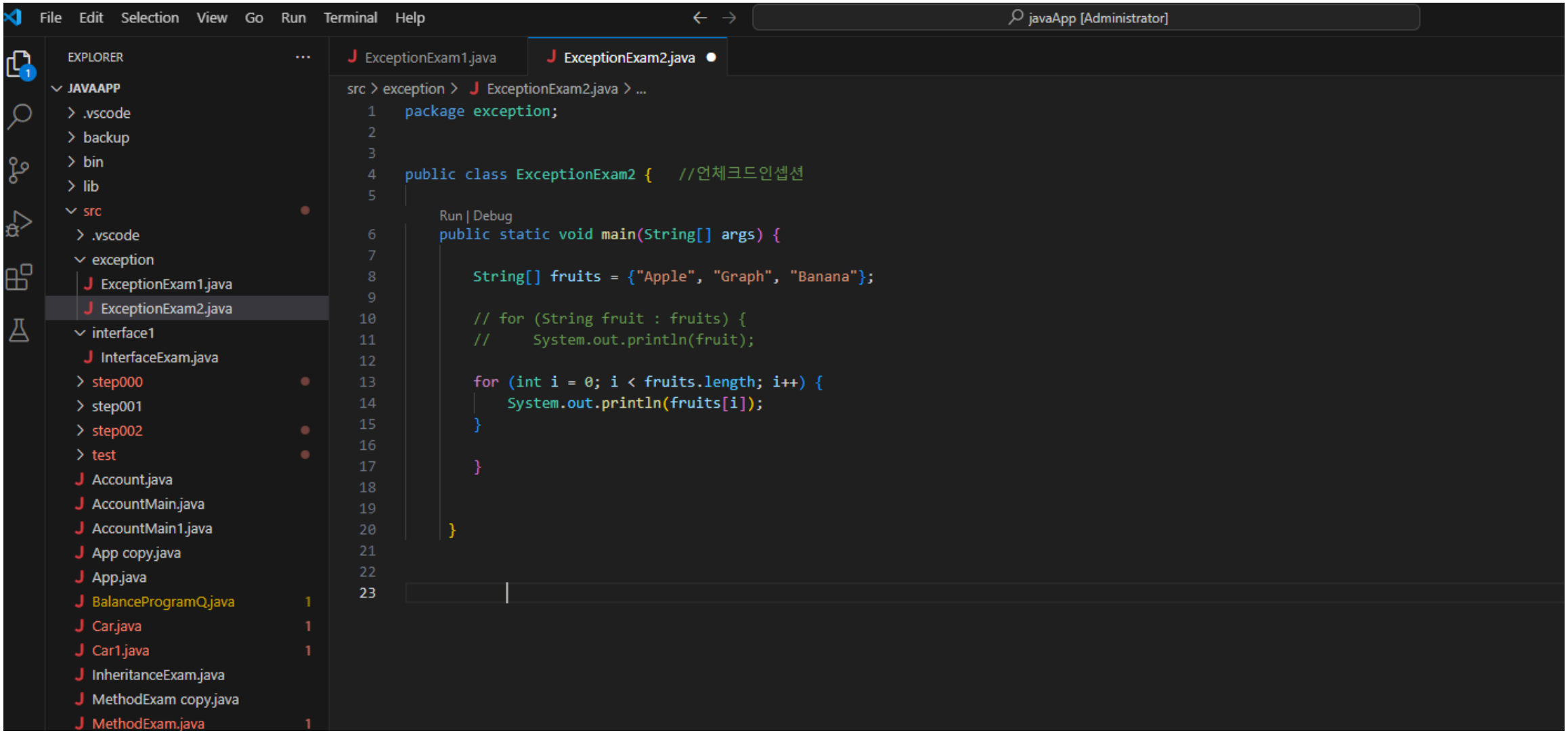

br = new BufferedReader(new FileReader(fileName));


BufferedReader br = null;
try {
String fileName = "c:\\a\\hello.txt";
new BufferedReader(new FileReader(fileName));
line = null;
line = br.readLine()) != null) {
em.out.println(line);
NotFoundException ex) {
StackTrace();
ut.println(x:"File not exist!!");

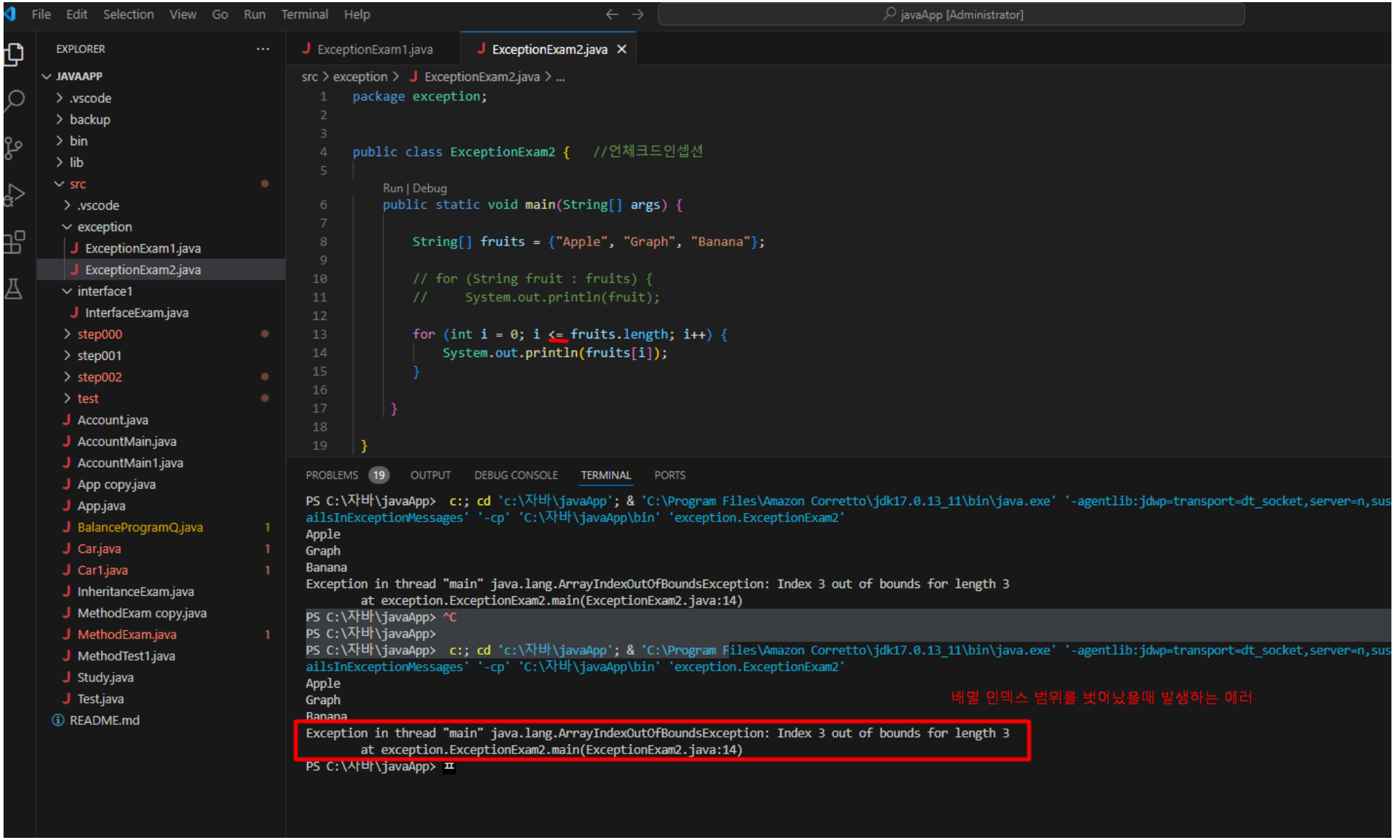
```java
package exception;


public class ExceptionExam2 {

    public static void main(String[] args) {

        String[] fruits = {"Apple", "Graph",
"Banana"};

        for (String fruit : fruits) {
            System.out.println(fruit);


        }


    }
}
```


Edit Selection View Go Run Terminal Help
javaApp [Administrator]
EXPLORER
JAVAAPP
.vscode
backup
bin
lib
src
.vscode
exception
ExceptionExam1.java
ExceptionExam2.java
interface1
InterfaceExam.java
step000
step001
step002
test
Account.java
AccountMain.java
AccountMain1.java
App copy.java
App.java
BalanceProgramQ.java 1
Car.java 1
Car1.java 1
InheritanceExam.java
ExceptionExam1.java
ExceptionExam2.java
src > exception > ExceptionExam2.java > ...
1 package exception;
2
3
4 public class ExceptionExam2 {
5
Run | Debug
6 public static void main(String[] args) {
7
8 String[] fruits = {"Apple", "Graph", "Banana"};
9
10 for (String fruit : fruits) {
11 System.out.println(fruit);
12
13
14 }
15
16
17 }
18 }
19
20

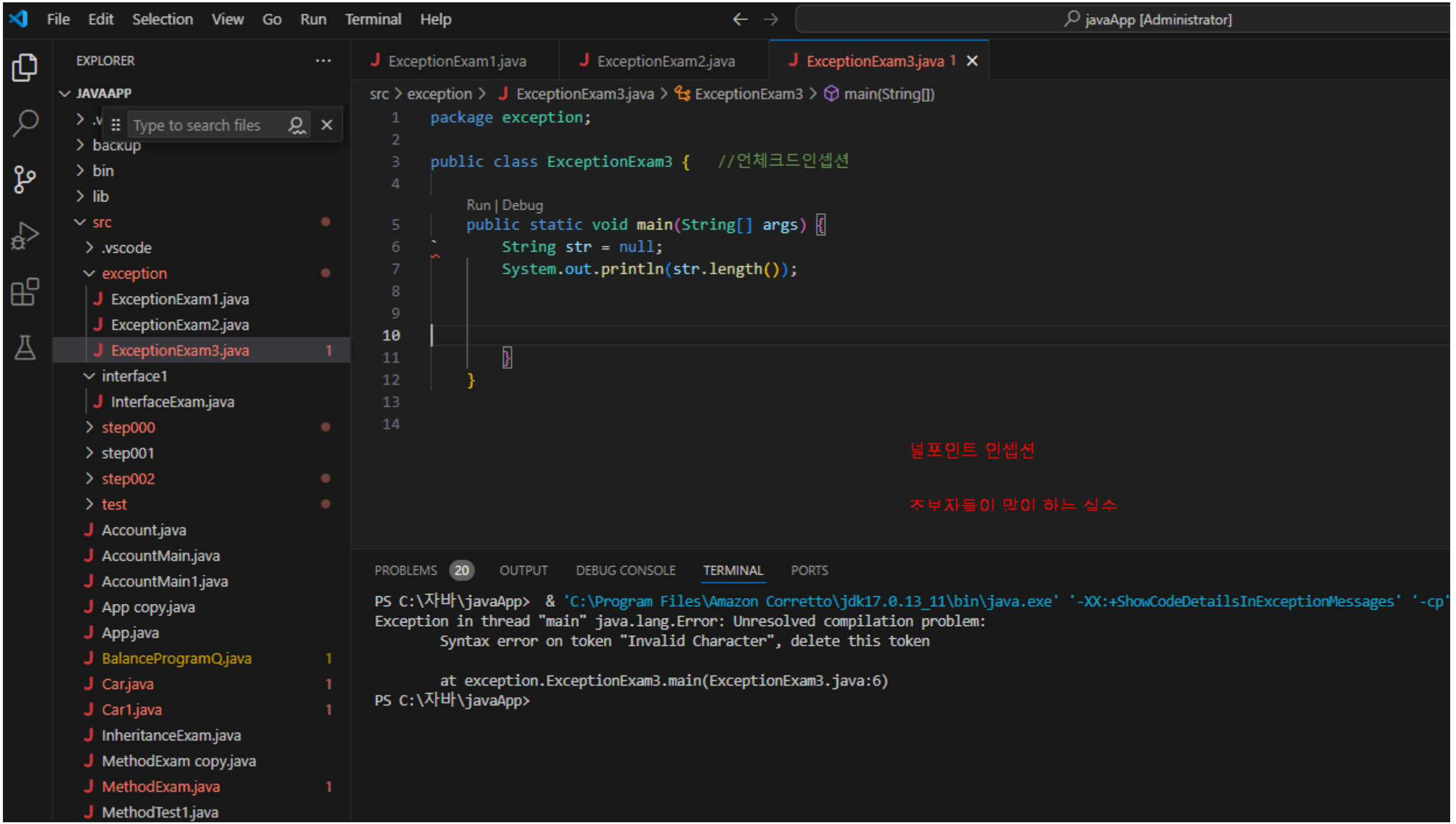
```java
package exception;


public class ExceptionExam2 {   //언체크드인셉션

    public static void main(String[] args) {

        String[] fruits = {"Apple", "Graph",
"Banana"};

        // for (String fruit : fruits) {
        //     System.out.println(fruit);

        for (int i = 0; i < fruits.length; i++)
{
            System.out.println(fruits[i]);
        }

        }


    }
```

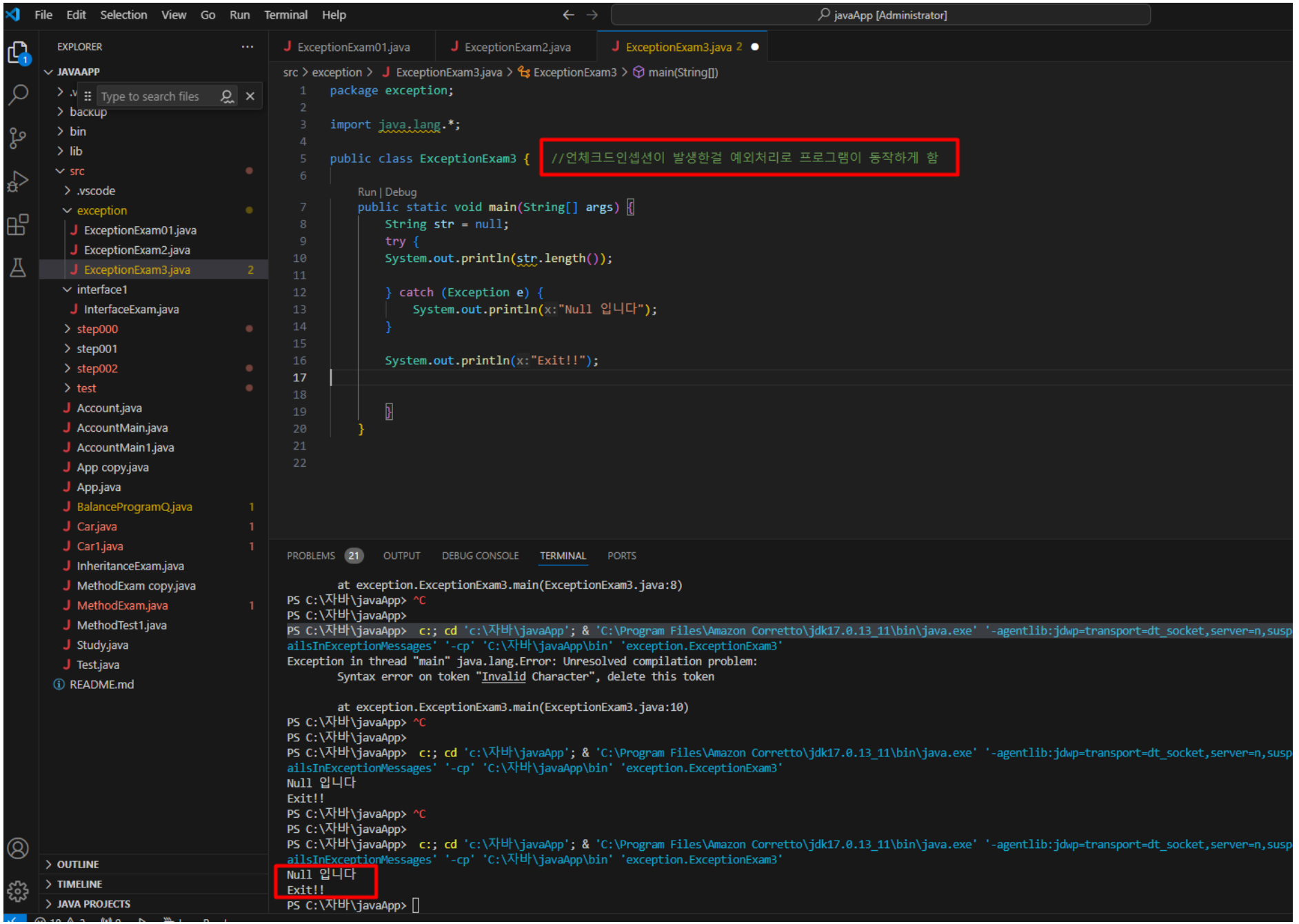

File Edit Selection View Go Run Terminal Help
javaApp [Administrator]
EXPLORER
JAVAAPP
.vscode
backup
bin
lib
src
.vscode
exception
ExceptionExam1.java
ExceptionExam2.java
interface1
InterfaceExam.java
step000
step001
step002
test
Account.java
AccountMain.java
AccountMain1.java
App copy.java
App.java
BalanceProgramQ.java
Car.java
Car1.java
InheritanceExam.java
MethodExam copy.java
MethodExam.java
ExceptionExam1.java
ExceptionExam2.java
src > exception > ExceptionExam2.java > ...
1 package exception;
2
3
4 public class ExceptionExam2 { //언체크드인셉션
5
Run | Debug
6 public static void main(String[] args) {
7
8 String[] fruits = {"Apple", "Graph", "Banana"};
9
10 // for (String fruit : fruits) {
11 // System.out.println(fruit);
12
13 for (int i = 0; i < fruits.length; i++) {
14 System.out.println(fruits[i]);
15 }
16
17 }
18
19
20 }
21
22
23

아 에러가 발생했구나 하고 코드를 수정하면 된다



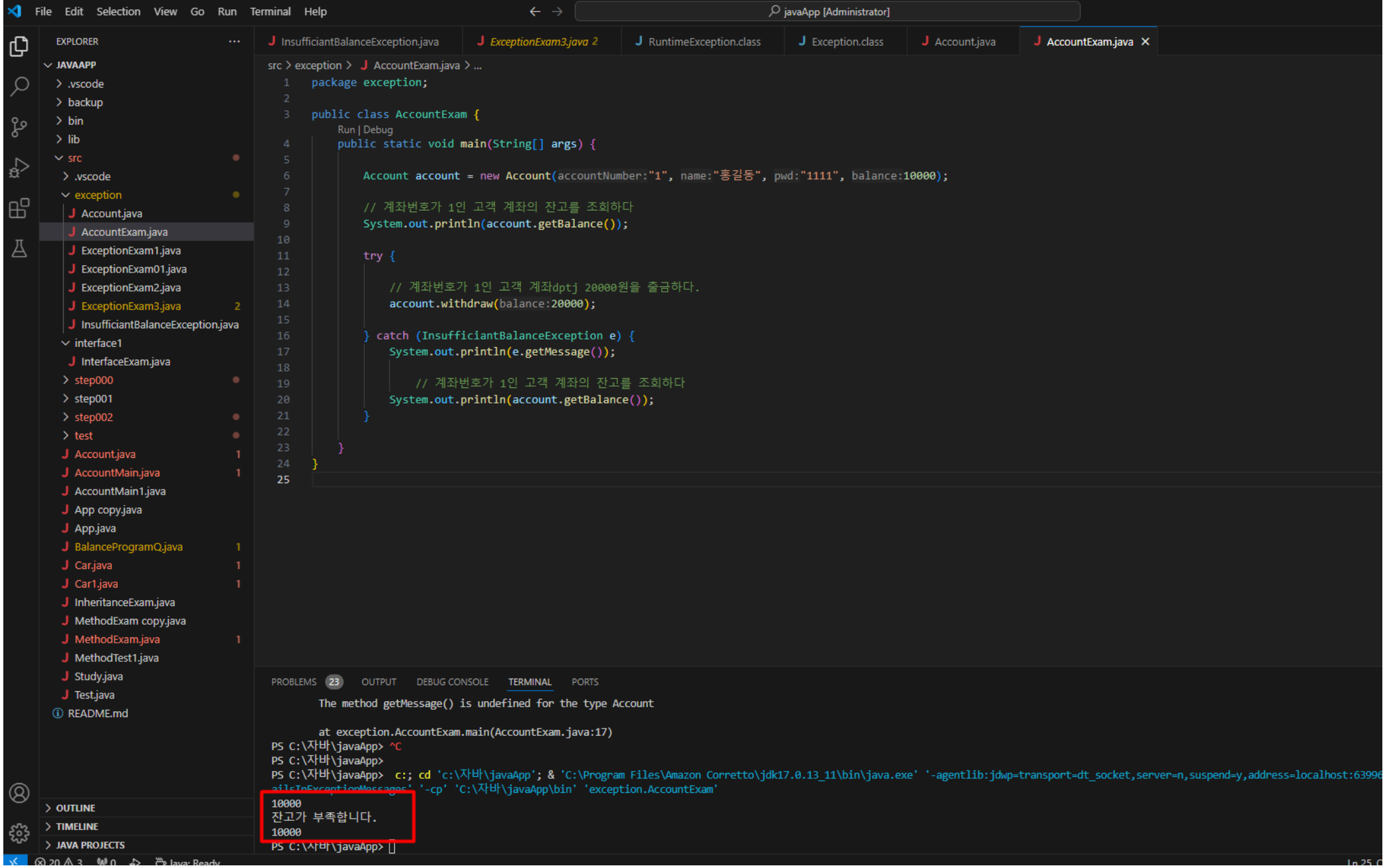
```java
package exception;

import java.lang.*;

public class ExceptionExam3 {    //언체크드인셉션

    public static void main(String[] args) {
        String str = null;
        System.out.println(str.length());



        }
    }
```


File Edit Selection View Go Run Terminal Help
javaApp [Administrator]
EXPLORER
JAVAAPP
Type to search files
backup
bin
lib
src
.vscode
exception
ExceptionExam1.java
ExceptionExam2.java
ExceptionExam3.java 1
interface1
InterfaceExam.java
step000
step001
step002
test
Account.java
AccountMain.java
AccountMain1.java
App copy.java
App.java
BalanceProgramQ.java 1
Car.java 1
Car1.java 1
InheritanceExam.java
MethodExam copy.java
MethodExam.java 1
MethodTest1.java
ExceptionExam1.java
ExceptionExam2.java
ExceptionExam3.java 1
src > exception > ExceptionExam3.java > ExceptionExam3 > main(String[])
1 package exception;
2
3 public class ExceptionExam3 { //언체크드인셉션
4
Run | Debug
5 public static void main(String[] args) {
6 String str = null;
7 System.out.println(str.length());
8
9
10
11 }
12 }
13
14
널포인트 인셉션
주부자들이 많이 하는 실수
PROBLEMS 20 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\자바\javaApp> & 'C:\Program Files\Amazon Corretto\jdk17.0.13_11\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
Syntax error on token "Invalid Character", delete this token
at exception.ExceptionExam3.main(ExceptionExam3.java:6)
PS C:\자바\javaApp>


File Edit Selection View Go Run Terminal Help
javaApp [Administrator]
EXPLORER
JAVAAPP
Type to search files
backup
bin
lib
src
.vscode
exception
ExceptionExam01.java
ExceptionExam2.java
ExceptionExam3.java 2
interface1
InterfaceExam.java
step000
step001
step002
test
Account.java
AccountMain.java
AccountMain1.java
App copy.java
App.java
BalanceProgramQ.java 1
Car.java 1
Car1.java 1
InheritanceExam.java
MethodExam copy.java
MethodExam.java 1
MethodTest1.java
Study.java
Test.java
README.md
OUTLINE
TIMELINE
JAVA PROJECTS
ExceptionExam01.java
ExceptionExam2.java
ExceptionExam3.java 2
src > exception > ExceptionExam3.java > ExceptionExam3 > main(String[])
1 package exception;
2
3 import java.lang.*;
4
5 public class ExceptionExam3 { //언체크드인셉션이 발생한걸 예외처리로 프로그램이 동작하게 함
6
Run | Debug
7 public static void main(String[] args) {
8 String str = null;
9 try {
10 System.out.println(str.length());
11
12 } catch (Exception e) {
13 System.out.println(x:"Null 입니다");
14 }
15
16 System.out.println(x:"Exit!!");
17
18
19 }
20 }
21
22
PROBLEMS 21 OUTPUT DEBUG CONSOLE TERMINAL PORTS
at exception.ExceptionExam3.main(ExceptionExam3.java:8)
PS C:\자바\javaApp> ^C
PS C:\자바\javaApp>
PS C:\자바\javaApp> c:; cd 'c:\자바\javaApp'; & 'C:\Program Files\Amazon Corretto\jdk17.0.13_11\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,susp
ailsInExceptionMessages' '-cp' 'C:\자바\javaApp\bin' 'exception.ExceptionExam3'
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
Syntax error on token "Invalid Character", delete this token
at exception.ExceptionExam3.main(ExceptionExam3.java:10)
PS C:\자바\javaApp> ^C
PS C:\자바\javaApp>
PS C:\자바\javaApp> c:; cd 'c:\자바\javaApp'; & 'C:\Program Files\Amazon Corretto\jdk17.0.13_11\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,susp
ailsInExceptionMessages' '-cp' 'C:\자바\javaApp\bin' 'exception.ExceptionExam3'
Null 입니다
Exit!!
PS C:\자바\javaApp> ^C
PS C:\자바\javaApp>
PS C:\자바\javaApp> c:; cd 'c:\자바\javaApp'; & 'C:\Program Files\Amazon Corretto\jdk17.0.13_11\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,susp
ailsInExceptionMessages' '-cp' 'C:\자바\javaApp\bin' 'exception.ExceptionExam3'
Null 입니다
Exit!!
PS C:\자바\javaApp>

```java
package exception;

import java.lang.*;

public class ExceptionExam3 {    //언체크드인셉션이 발생한걸 예외처리로 프로그램이 동작하게 함

    public static void main(String[] args) {
        String str = null;
        try {
        System.out.println(str.length());

        } catch (Exception e) {
            System.out.println("Null 입니다");
        }

        System.out.println("Exit!!");


        }
    }
```

```java
package exception;

public class AccountExam {
    public static void main(String[] args) {

        Account account = new Account("1", "홍길동", "1111", 10000);

        // 계좌번호가 1인 고객 계좌의 잔고를 조회하다
        System.out.println(account.getBalance());

        try {

            // 계좌번호가 1인 고객 계좌dptj 20000원을 출금하다.
            account.withdraw(20000);

        } catch (InsufficiantBalanceException e) {
            System.out.println(e.getMessage());

                // 계좌번호가 1인 고객 계좌의 잔고를 조회하다
            System.out.println(account.getBalance());
        }

    }
```

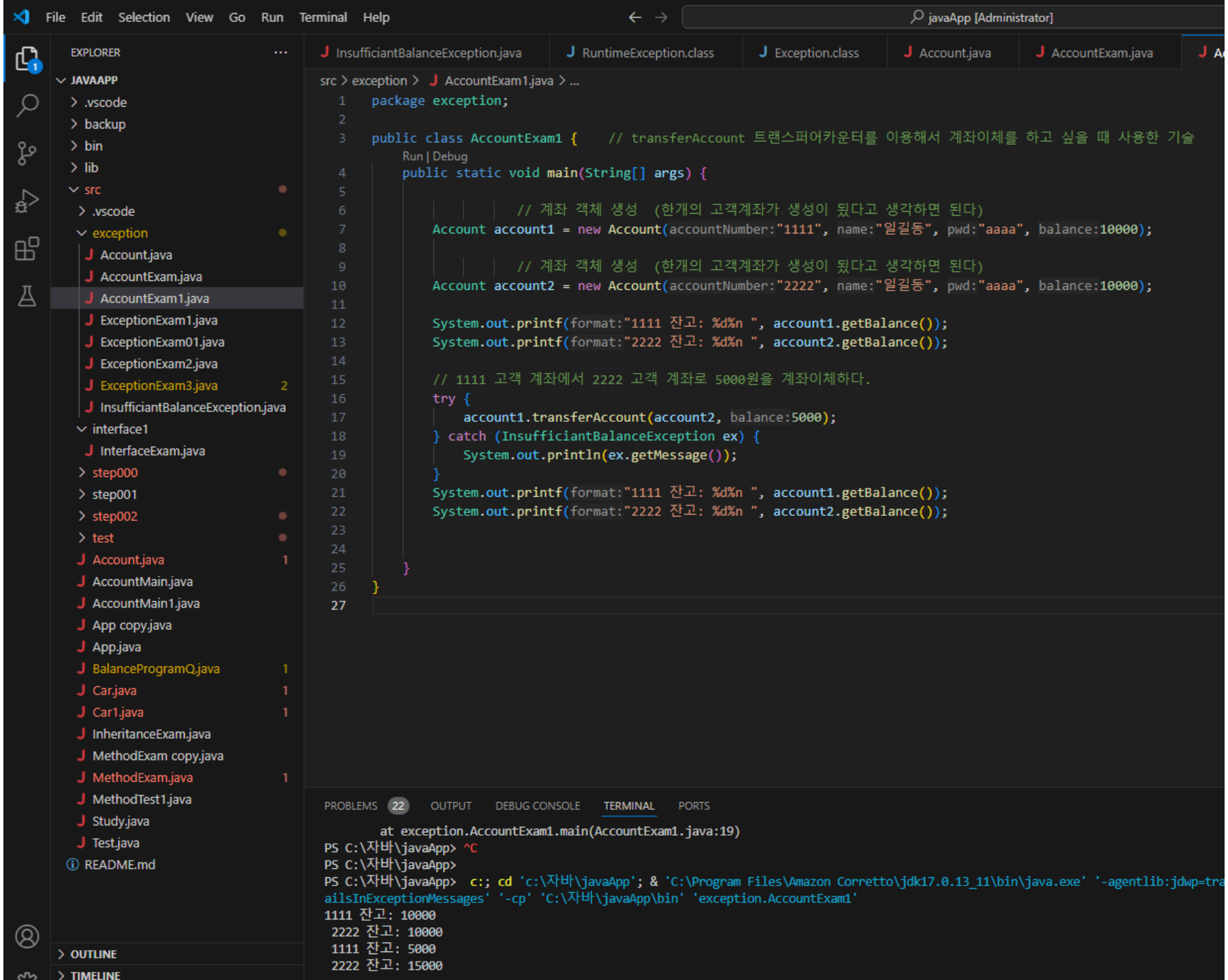

File Edit Selection View Go Run Terminal Help
javaApp [Administrator]
EXPLORER
JAVAAPP
.vscode
backup
bin
lib
src
.vscode
exception
Account.java
AccountExam.java
ExceptionExam1.java
ExceptionExam01.java
ExceptionExam2.java
ExceptionExam3.java 2
InsufficiantBalanceException.java
interface1
InterfaceExam.java
step000
step001
step002
test
Account.java 1
AccountMain.java 1
AccountMain1.java
App copy.java
App.java
BalanceProgramQ.java 1
Car.java 1
Car1.java 1
InheritanceExam.java
MethodExam copy.java
MethodExam.java 1
MethodTest1.java
Study.java
Test.java
README.md
OUTLINE
TIMELINE
JAVA PROJECTS
InsufficiantBalanceException.java
ExceptionExam3.java 2
RuntimeException.class
Exception.class
Account.java
AccountExam.java
src > exception > AccountExam.java > ...
1 package exception;
2
3 public class AccountExam {
Run | Debug
4 public static void main(String[] args) {
5
6 Account account = new Account(accountNumber:"1", name:"홍길동", pwd:"1111", balance:10000);
7
8 // 계좌번호가 1인 고객 계좌의 잔고를 조회하다
9 System.out.println(account.getBalance());
10
11 try {
12
13 // 계좌번호가 1인 고객 계좌dptj 20000원을 출금하다.
14 account.withdraw(balance:20000);
15
16 } catch (InsufficiantBalanceException e) {
17 System.out.println(e.getMessage());
18
19 // 계좌번호가 1인 고객 계좌의 잔고를 조회하다
20 System.out.println(account.getBalance());
21 }
22
23 }
24 }
25
PROBLEMS 23
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
The method getMessage() is undefined for the type Account
at exception.AccountExam.main(AccountExam.java:17)
PS C:\자바\javaApp> ^C
PS C:\자바\javaApp>
PS C:\자바\javaApp> c:; cd 'c:\자바\javaApp'; & 'C:\Program Files\Amazon Corretto\jdk17.0.13_11\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:63996
'-cp' 'C:\자바\javaApp\bin' 'exception.AccountExam'
10000
잔고가 부족합니다.
10000
PS C:\자바\javaApp>

```
2   public class Account {
18      public Account(String accountNumber, String name, String pwd, int balance) {
19          this.accountNumber = accountNumber;
20          this.name = name;
21          this.pwd = pwd;
22          this.balance = balance;
23      }
24
25      // instance method
26      public int getBalance() {
27          return this.balance;
28      }
29
30      // 입금하다   <- 여기서 this를 안넣으면 입금금액이 변경이 안됨!! 주의!!
31      public void deposit(int balance) {
32          this.balance += balance;
33          //this.balance = this.balance + balance;
34      }
35
36      // 출금하다   <-
37      public void withdraw(int balance) throws InsufficiantBalanceException {
38          if (balance > this.balance) {
39              throw new InsufficiantBalanceException();
40
41          }
42          this.balance -= balance;
43          //this.balance = this.balance - balance;
44      }
45
46      // // 계좌이체하다
47      // public void transferAccount(Account account, int balance) {
48      //     this.withdraw(balance);
49      //     account.deposit(balance);
50      // }
51
52
```

```java
package exception;

public class AccountExam {
    Run | Debug
    public static void main(String[] args) {

        Account account = new Account(accountNumber:"1", name:"홍길동", pwd:"1111", balance:10000);

        // 계좌번호가 1인 고객 계좌의 잔고를 조회하다
        System.out.println(account.getBalance());

        try {

            // 계좌번호가 1인 고객 계좌dptj 20000원을 출금하다.
            account.withdraw(balance:20000);

        } catch (InsufficiantBalanceException e) {
            System.out.println(e.getMessage());

                // 계좌번호가 1인 고객 계좌의 잔고를 조회하다
            System.out.println(account.getBalance());
        }

    }
}

```

```
    //this.balance = this.balance - balance;
 }

// // 계좌이체하다
// public void transferAccount(Account account, int balance) {
//     this.withdraw(balance);
//     account.deposit(balance);
// }


public String getPwd() {
    return pwd;
}

public void setPwd(String pwd) {
    this.pwd = pwd;
}

public String printAccount() {
    return String.format("계좌번호: %s, 이름: %s, 비밀번호: %s, 잔고: %d\n",
        this.accountNumber, this.name, this.pwd, this.balance);
}

@Override
public String toString() {
```

```
        Account account = new Account("1", "홍길동", "1111", 10000);

        // 계좌번호가 1인 고객 계좌의 잔고를 조회하다
        System.out.println(account.getBalance());

        try {

            // 계좌번호가 1인 고객 계좌dptj 20000원을 출금하다.
            account.withdraw(20000);

        } catch (InsufficiantBalanceException e) {
            System.out.println(e.getMessage());

                // 계좌번호가 1인 고객 계좌의 잔고를 조회하다
            System.out.println(account.getBalance());
        }

    }
}
```


File Edit Selection View Go Run Terminal Help
javaApp [Administrator]
EXPLORER
JAVAAPP
.vscode
backup
bin
lib
src
.vscode
exception
Account.java
AccountExam.java
AccountExam1.java
ExceptionExam1.java
ExceptionExam01.java
ExceptionExam2.java
ExceptionExam3.java
InsufficiantBalanceException.java
interface1
InterfaceExam.java
step000
step001
step002
test
Account.java
AccountMain.java
AccountMain1.java
App copy.java
App.java
BalanceProgramQ.java
Car.java
Car1.java
InheritanceExam.java
MethodExam copy.java
MethodExam.java
MethodTest1.java
Study.java
Test.java
README.md
OUTLINE
TIMELINE
InsufficiantBalanceException.java
RuntimeException.class
Exception.class
Account.java
AccountExam.java
src > exception > AccountExam1.java > ...
1 package exception;
2
3 public class AccountExam1 { // transferAccount 트랜스퍼어카운터를 이용해서 계좌이체를 하고 싶을 때 사용한 기술
Run | Debug
4 public static void main(String[] args) {
5
6 // 계좌 객체 생성 (한개의 고객계좌가 생성이 됐다고 생각하면 된다)
7 Account account1 = new Account(accountNumber:"1111", name:"일길동", pwd:"aaaa", balance:10000);
8
9 // 계좌 객체 생성 (한개의 고객계좌가 생성이 됐다고 생각하면 된다)
10 Account account2 = new Account(accountNumber:"2222", name:"일길동", pwd:"aaaa", balance:10000);
11
12 System.out.printf(format:"1111 잔고: %d%n ", account1.getBalance());
13 System.out.printf(format:"2222 잔고: %d%n ", account2.getBalance());
14
15 // 1111 고객 계좌에서 2222 고객 계좌로 5000원을 계좌이체하다.
16 try {
17 account1.transferAccount(account2, balance:5000);
18 } catch (InsufficiantBalanceException ex) {
19 System.out.println(ex.getMessage());
20 }
21 System.out.printf(format:"1111 잔고: %d%n ", account1.getBalance());
22 System.out.printf(format:"2222 잔고: %d%n ", account2.getBalance());
23
24
25 }
26 }
27
PROBLEMS 22 OUTPUT DEBUG CONSOLE TERMINAL PORTS
at exception.AccountExam1.main(AccountExam1.java:19)
PS C:\자바\javaApp> ^C
PS C:\자바\javaApp>
PS C:\자바\javaApp> c:; cd 'c:\자바\javaApp'; & 'C:\Program Files\Amazon Corretto\jdk17.0.13_11\bin\java.exe' '-agentlib:jdwp=tra
ailsInExceptionMessages' '-cp' 'C:\자바\javaApp\bin' 'exception.AccountExam1'
1111 잔고: 10000
2222 잔고: 10000
1111 잔고: 5000
2222 잔고: 15000

```java
package exception;

public class AccountExam1 {     // transferAccount 트랜스퍼어카운터를 이용해서 계좌이체를 하고 싶을 때
    public static void main(String[] args) {

                    // 계좌 객체 생성  (한개의 고객계좌가 생성이 됐다고 생각하면 된다)
        Account account1 = new Account("1111", "일길동", "aaaa", 10000);

                    // 계좌 객체 생성  (한개의 고객계좌가 생성이 됐다고 생각하면 된다)
        Account account2 = new Account("2222", "일길동", "aaaa", 10000);

        System.out.printf("1111 잔고: %d%n ", account1.getBalance());
        System.out.printf("2222 잔고: %d%n ", account2.getBalance());

        // 1111 고객 계좌에서 2222 고객 계좌로 5000원을 계좌이체하다.
        try {
            account1.transferAccount(account2, 5000);
        } catch (InsufficiantBalanceException ex) {
            System.out.println(ex.getMessage());
        }
        System.out.printf("1111 잔고: %d%n ", account1.getBalance());
        System.out.printf("2222 잔고: %d%n ", account2.getBalance());


    }
```

Generic (제네릭)

● Generic(제네릭)이란?

· 데이터 타입(data type)을 일반화하는(generalize) 것을 의미한다.

· 제네릭은 클래스나 메소드를 선언할 때 데이터 타입을 정하는 것이 아니라,

인스턴스를 생성할 때나 메소드를 호출할 때 데이터 타입을 정한다.

● Generic(제네릭)의 장점

· 제네릭을 사용하면 잘못된 데이터 타입이 들어올 수 있는 것을 컴파일 단계에서 방지할 수 있다.

· 클래스 외부에서 타입을 지정해주기 때문에 따로 데이터 타입을 체크하고 변환해줄 필요가 없다.

il

framework, some internationalization support classes, a service loader, properties, random number generation, string parsing and scanning classes, base64 encoding and decoding, lity classes. This package also contains legacy collection classes and legacy date and time classes.

ramework

ine, and design rationale, please see:

ework Documentation

mming guide with examples of use of the collections framework, please see:

ework Tutorial

ackage	Description
ava.util.concurrent	Utility classes commonly useful in concurrent programming.
ava.util.function	*Functional interfaces* provide target types for lambda expressions and method references.
ava.util.jar	Provides classes for reading and writing the JAR (Java ARchive) file format, which is based on the standard ZIP file format with an optional manife
ava.util.logging	Provides the classes and interfaces of the Java 2 platform's core logging facilities.
ava.util.prefs	This package allows applications to store and retrieve user and system preference and configuration data.
ava.util.random	This package contains classes and interfaces that support a generic API for random number generation.

base
util

ayList<E>

ct
ostractCollection<E>
il.AbstractList<E>
va.util.ArrayList<E>

rs:

elements in this list

d Interfaces:

, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess

ubclasses:

t, RoleList, RoleUnresolvedList

ArrayList<E>
ractList<E>
ist<E>, RandomAccess, Cloneable, Serializable

y implementation of the List interface. Implements all optional list operations, and permits all elements, including null. In addition to implementing the List interface, this class p
array that is used internally to store the list. (This class is roughly equivalent to Vector, except that it is unsynchronized.)

mpty, get, set, iterator, and listIterator operations run in constant time. The add operation runs in *amortized constant time*, that is, adding n elements requires O(n) time. All o
ughly speaking). The constant factor is low compared to that for the LinkedList implementation.

st instance has a *capacity*. The capacity is the size of the array used to store the elements in the list. It is always at least as large as the list size. As elements are added to an ArrayL
The details of the growth policy are not specified beyond the fact that adding an element has constant amortized time cost.

`true` if this list contains no elements

contains

```
public boolean contains(Object o)
```

Returns `true` if this list contains the specified element. More formally, returns `true` if and only if this list contains at least one element e

Specified by:

`contains` in interface `Collection<E>`

Specified by:

`contains` in interface `List<E>`

Overrides:

`contains` in class `AbstractCollection<E>`

Parameters:

`o` - element whose presence in this list is to be tested

Returns:

<E>

n this list

e<E>

lasses:

SequentialList, ArrayList, AttributeList, CopyOnWriteArrayList, LinkedList, RoleList, RoleUnresolvedList, Stack, Vector

---

<E>

so known as a *sequence*). The user of this interface has precise control over where in the list each element is inserted. The user can access elements by their integer index (position
e list.

y allow duplicate elements. More formally, lists typically allow pairs of elements e1 and e2 such that `e1.equals(e2)`, and they typically allow multiple null elements if they allow nul
meone might wish to implement a list that prohibits duplicates, by throwing runtime exceptions when the user attempts to insert them, but we expect this usage to be rare.

s additional stipulations, beyond those specified in the `Collection` interface, on the contracts of the `iterator`, `add`, `remove`, `equals`, and `hashCode` methods. Declarations for other
nvenience.

des four methods for positional (indexed) access to list elements. Lists (like Java arrays) are zero based. Note that these operations may execute in time proportional to the index va
nkedList class, for example). Thus, iterating over the elements in a list is typically preferable to indexing through it if the caller does not know the implementation.

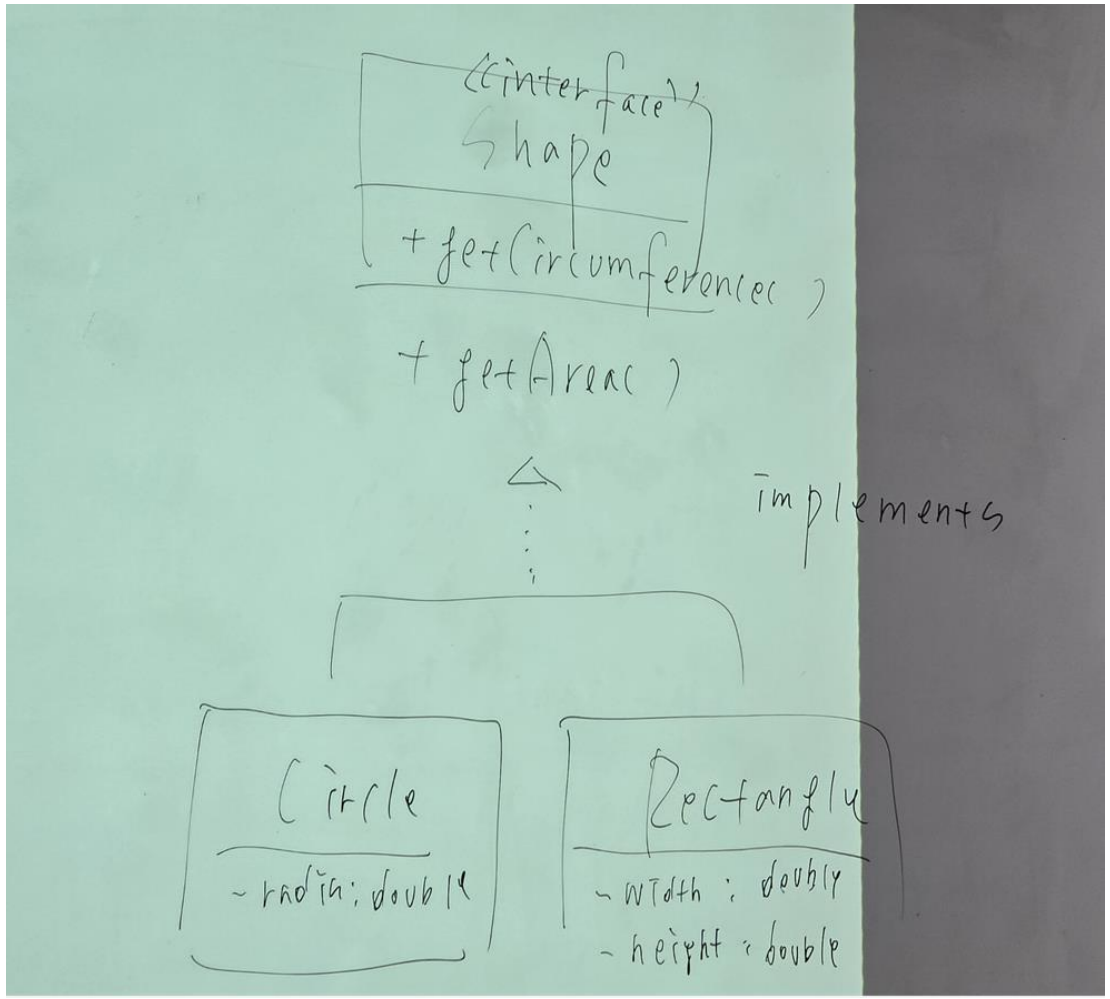
des a special iterator, called a `ListIterator`, that allows element insertion and replacement, and bidirectional access in addition to the normal operations that the `Iterator` interfa
st iterator that starts at a specified position in the list.

des two methods to search for a specified object. From a performance standpoint, these methods should be used with caution. In many implementations they will perform costly line

des two methods to efficiently insert and remove multiple elements at an arbitrary point in the list.

ible for lists to contain themselves as elements, extreme caution is advised: the `equals` and `hashCode` methods are no longer well defined on such a list.

숙제



UML 클래스 다이어그램 (위에 그림이 UML 다이어그램으로 보임

객체지향 설계

```java
@Override
public double getArea() {

    return Math.PI * this.radius * this.radius;
}
```

SS USE TREE PREVIEW NEW DEPRECATED INDEX HELP

METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH:

---

ls for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

d Summary

ds

ifier and Type	Field	Description
ic final double	E	The double value that is closer than any other to *e*, the base of the natural logarithms.
ic final double	PI	The double value that is closer than any other to *pi*, the ratio of the circumference of a circle to its diameter.

● Generic(제네릭)이란?

· 데이터 타입(data type)을 일반화하는(generalize) 것을 의미한다.

· 제네릭은 클래스나 메소드를 선언할 때 데이터 타입을 정하는 것이 아니라,

인스턴스를 생성할 때나 메소드를 호출할 때 데이터 타입을 정한다.

● Generic(제네릭)의 장점

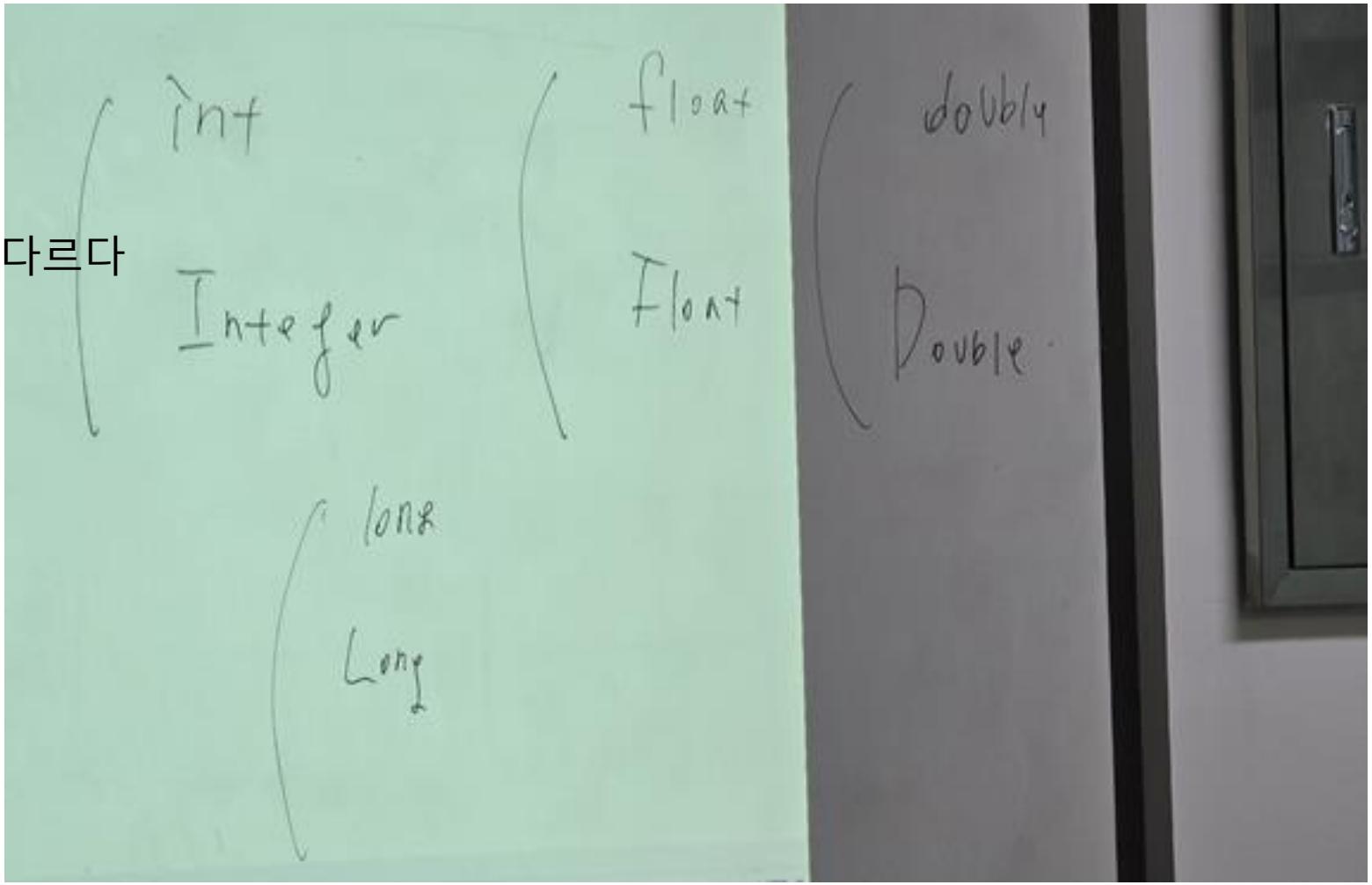
· 제네릭을 사용하면 잘못된 데이터 타입이 들어올 수 있는 것을 컴파일 단계에서 방지할 수 있다.

· 클래스 외부에서 타입을 지정해주기 때문에 따로 데이터 타입을 체크하고 변환

All Classes and Interfaces | Interfaces | Classes | Enum Classes | Exceptions | Errors

Class	Description
AbstractCollection<E>	This class provides a skeletal implementation of the Collection interfa interface.
AbstractList<E>	This class provides a skeletal implementation of the List interface to m backed by a "random access" data store (such as an array).
AbstractMap<K,V>	This class provides a skeletal implementation of the Map interface, to m
AbstractMap.SimpleEntry<K,V>	An Entry maintaining a key and a value.
AbstractMap.SimpleImmutableEntry<K,V>	An unmodifiable Entry maintaining a key and a value.
AbstractQueue<E>	This class provides skeletal implementations of some Queue operations.
AbstractSequentialList<E>	This class provides a skeletal implementation of the List interface to m backed by a "sequential access" data store (such as a linked list).
AbstractSet<E>	This class provides a skeletal implementation of the Set interface to mi
ArrayDeque<E>	Resizable-array implementation of the Deque interface.
ArrayList<E>	Resizable-array implementation of the List interface.
Arrays	This class contains various methods for manipulating arrays (such as so
Base64	This class consists exclusively of static methods for obtaining encoders
Base64.Decoder	This class implements a decoder for decoding byte data using the Base6 2045.

서로 다르다



int는 기본자료형이고 Intergar 는 레퍼런스형(참조형) 이다 제네릭은 레퍼런스형만 올수있다 ?

(E) 서식(O) 보기(V) 도움말(H)

```
blic class Box<T> {

rivate T item;
```

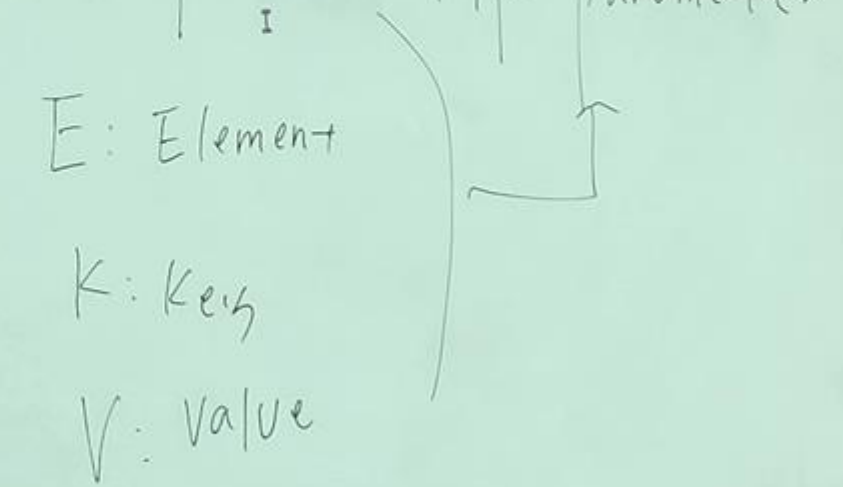
T : Type :

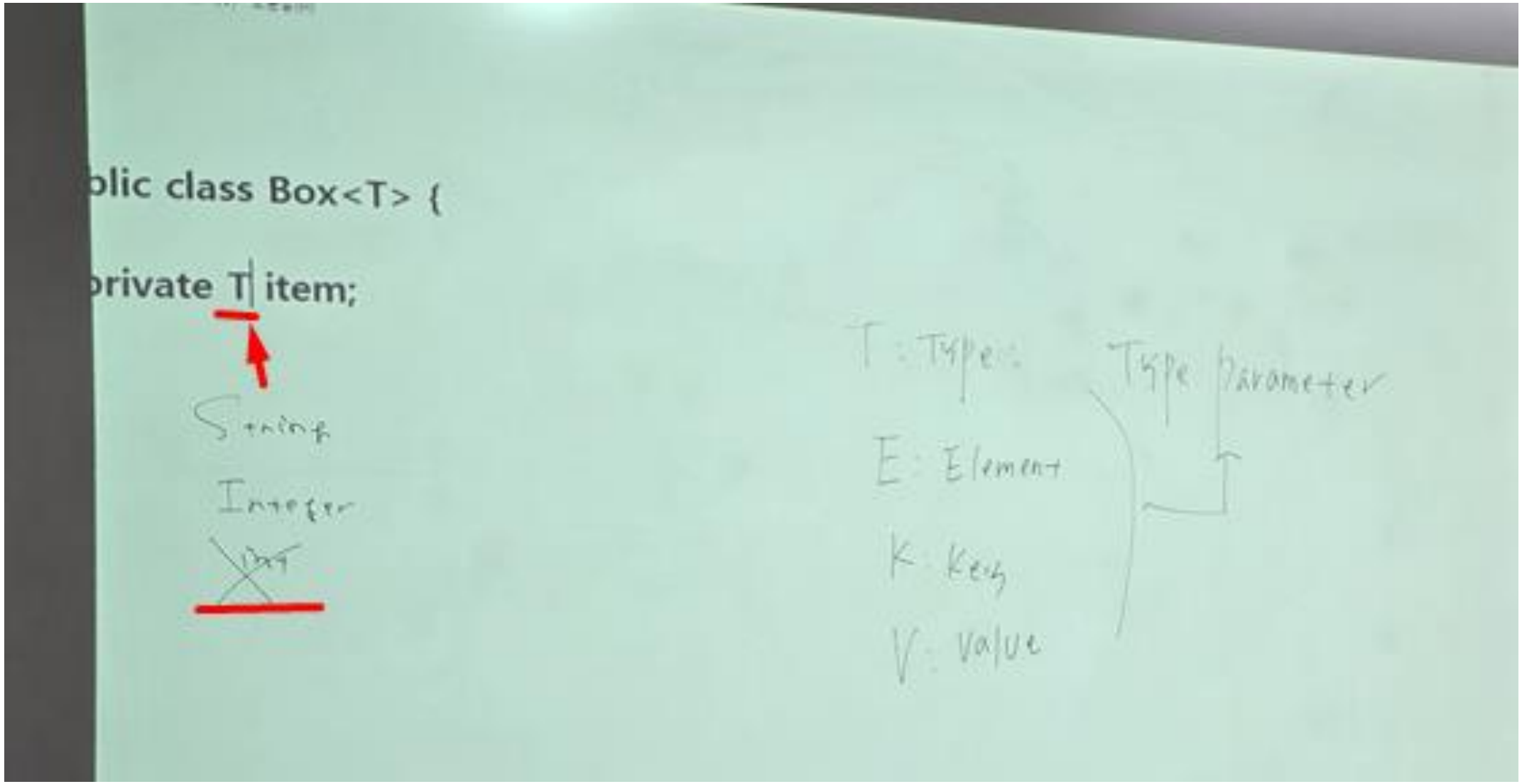
E : Element

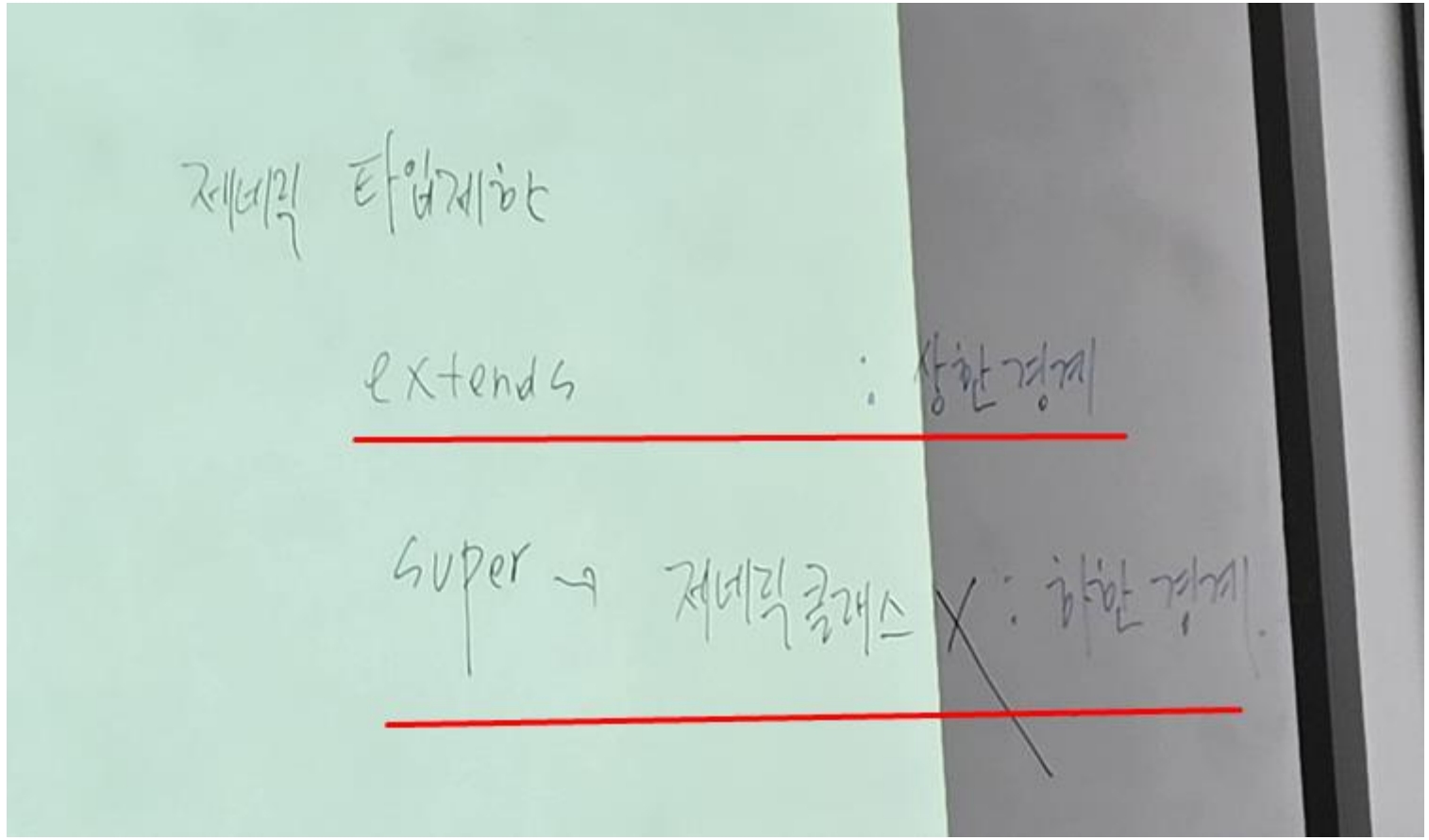
K : Key

V : Value

Type Parameter




olic class Box<T> {
rivate T item;
String
Integer
Dog
T : Type
Type Parameter
E : Element
K : Key
V : Value


제네릭 타입제한
extends : 상한경계
super → 제네릭클래스 X : 하한경계

```java
package generic;


// 제네릭 클래스
class Box<T extends Number> {  // Type Parameter

    private T item;

    public Box() {
    }

    public Box(T item) {
        this.item = item;
    }


    public void setItem(T item) {
        this.item = item;
    }

    public T getItem() {
        return this.item;
    }

}


public class GenericExam1 {
    Run | Debug
    public static void main(String[] args) {

        // 객체 생성
        Box<String> box1 = new Box<>("Apple");
```

Number 자리에 인터페이스도 올 수 있다

ces:

s:

cLong, BigDecimal, BigInteger, Byte, Double, DoubleAccumulator, DoubleAdder, Float, Integer, Long, LongAccumulator, LongAdder, Short

---

ss **Number**

able

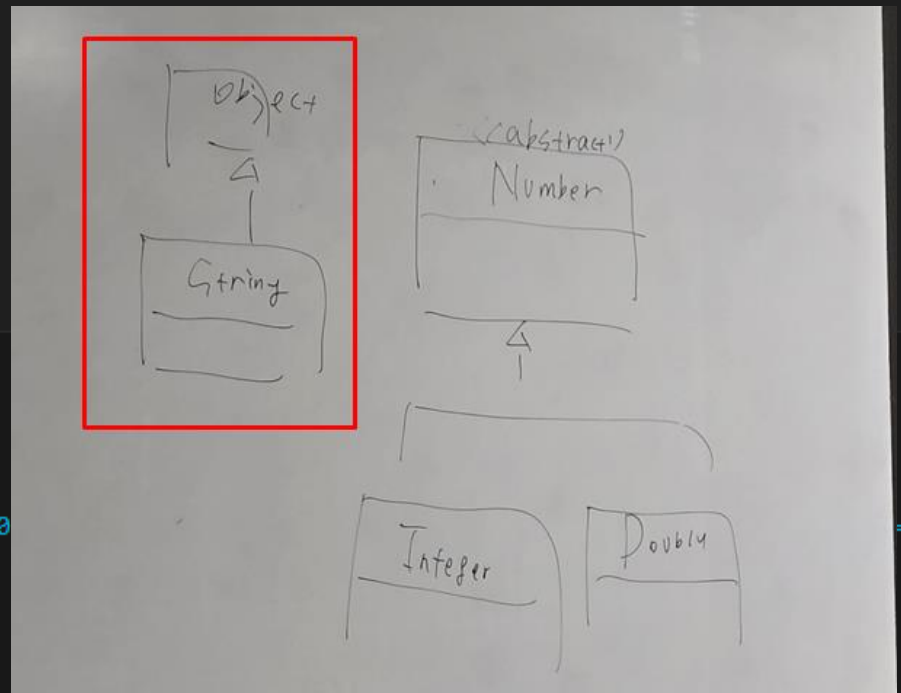
ber is the superclass of platform classes representing numeric values that are convertible to the primitive types byte, double, float, int, long, and short. The specific semantics o particular Number implementation to a given primitive type is defined by the Number implementation in question. For platform classes, the conversion is often analogous to a narrow e conversion as defined in *The Java Language Specification* for converting between primitive types. Therefore, conversions may lose information about the overall magnitude of a nu n return a result of a different sign than the input. See the documentation of a given Number implementation for conversion details.

ification:

ve Conversion
tive Conversion

```java
    private T item;

    public Box() {
    }

    public Box(T item) {
        this.item = item;
    }


    public void setItem(T item) {
        this.item = item;
    }

    public T getItem() {
        return this.item;
    }

}


public class GenericExam1 {
    Run | Debug
    public static void main(String[] args) {

        // 객체 생성
        Box<String> box1 = new Box<>("Apple");
```

Number 이거나 Number 상속받은 하위클래스타입들만 올 수 있어서
String가 오류가 난다 (컴파일에러가 난다)



PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
10
PS C:\자바\javaApp> ^C
PS C:\자바\javaApp>
PS C:\자바\javaApp>  c:; cd 'c:\자바\javaApp'; & 'C:\Program Files\Amazon Corretto\jdk17.0                                                    =n,susp
ailsInExceptionMessages' '-cp' 'C:\자바\javaApp\bin' 'generic.GenericExam1'
Apple
10
PS C:\자바\javaApp> ^C
```

AND 물어봐야함

```
class AnimalList <T extends LandAnimal & WarmBlood> {

                    ...

}
```

● 제네릭 메소드

```
public static <T  extends Comparable <?  super T>>

                                    void sort (List<T> list)
```

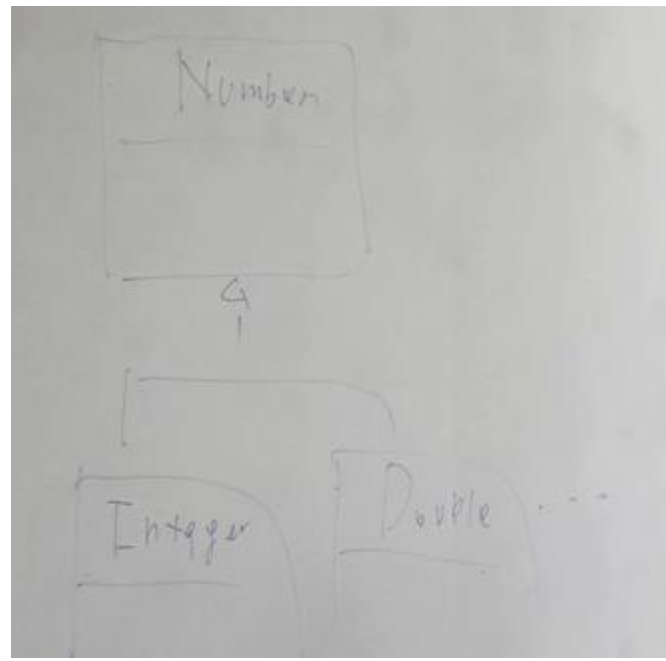
- T는 Comparable 인터페이스를 구현한 타입이어야 한다.

● 제네릭 메소드

제네릭클래스에서는 ? 사용못하고
제네릭 메소드에서는 ? 사용가능

```
public static <T  extends Comparable <?  super T>>

                                    void sort (List<T> list)
```

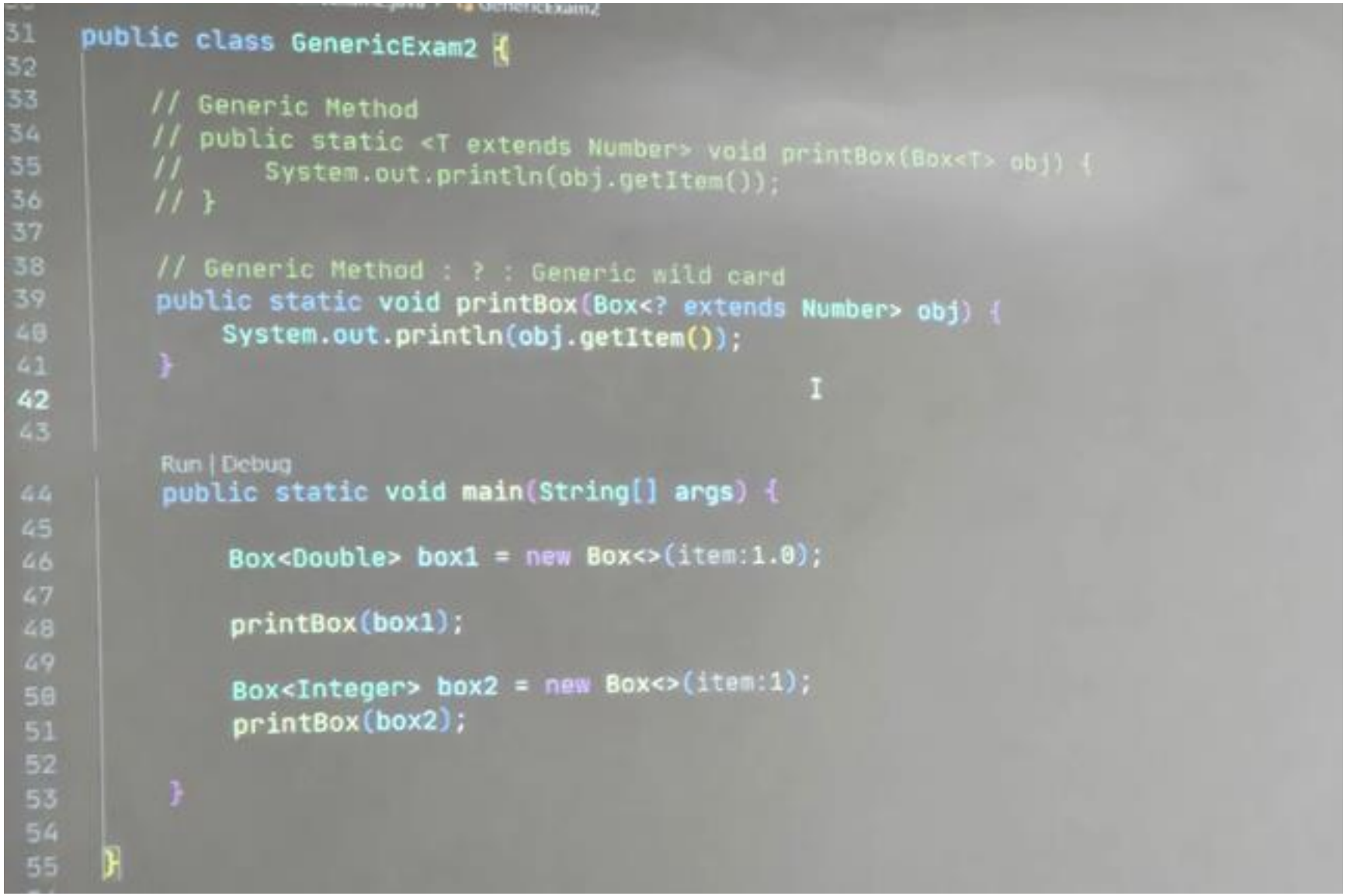
- T는 Comparable 인터페이스를 구현한 타입이어야 한다.


Number
Integer
Double

```java
package generic;


// 제네릭 클래스
class Box<T extends Number> {  // Type Parameter

        private T item;

        public Box() {
        }

        public Box(T item) {
            this.item = item;
        }

       // setter method
        public void setItem(T item) {
            this.item = item;
        }

        // getter method
        public T getItem() {
            return this.item;
        }


}


public class GenericExam2 {

    // Generic Method
    public static <T extends Number> void printBox(Box<T> obj) {
        System.out.println(obj.getItem());

    }



    public static void main(String[] args) {

        Box<Double> box1 = new Box<>(1.0);

        printBox(box1);

        Box<Integer> box2 = new Box<>(1);

        printBox(box2);
    }
}
```

```java
package generic;


// 제네릭 클래스
class Box<T extends Number> {  // Type Parameter

        private T item;

        public Box() {
        }

        public Box(T item) {
            this.item = item;
        }

       // setter method
        public void setItem(T item) {
            this.item = item;
        }

        // getter method
        public T getItem() {
            return this.item;
        }


}

public class GenericExam2 {

    // Generic Method
    public static <T extends Number> void printBox(Box<T> obj) {
        System.out.println(obj.getItem());

    }



    Run | Debug
    public static void main(String[] args) {

        Box<Double> box1 = new Box<>(item:1.0);

        printBox(box1);

        Box<Integer> box2 = new Box<>(item:1);

        printBox(box2);
    }
}

```

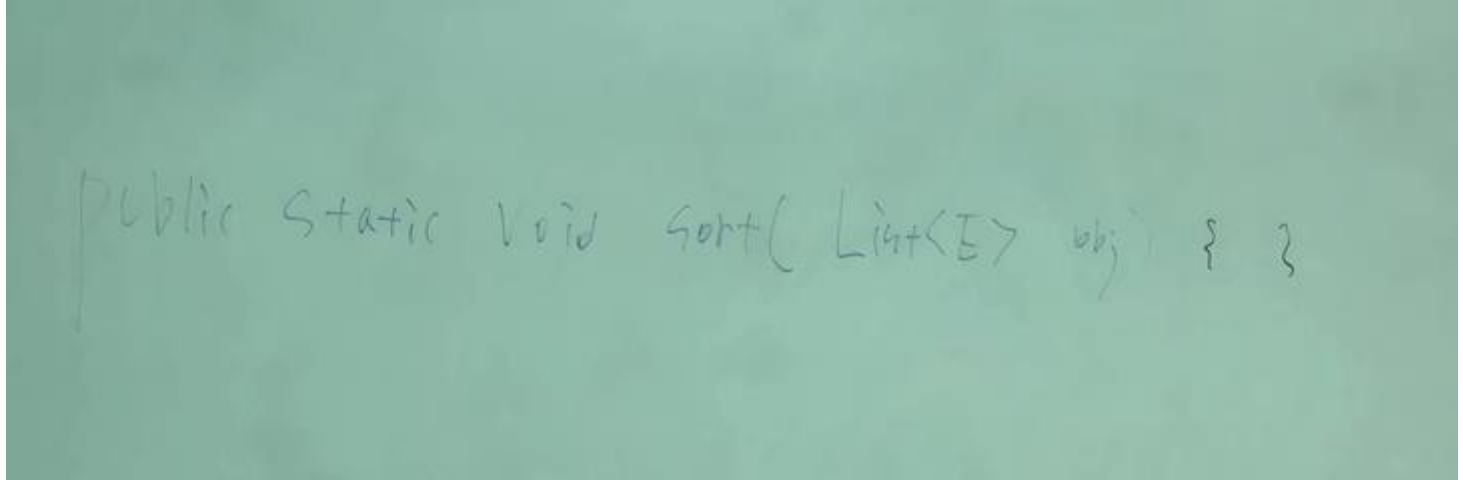
```java

public class GenericExam2 {

    // Generic Method
    public static <T extends Number> void printBox(Box<T> obj) {
        System.out.println(obj.getItem());

    }



    Run | Debug
    public static void main(String[] args) {

        Box<Double> box1 = new Box<>(item:1.0);

        printBox(box1);

        Box<Integer> box2 = new Box<>(item:1);

        printBox(box2);
    }
}

```

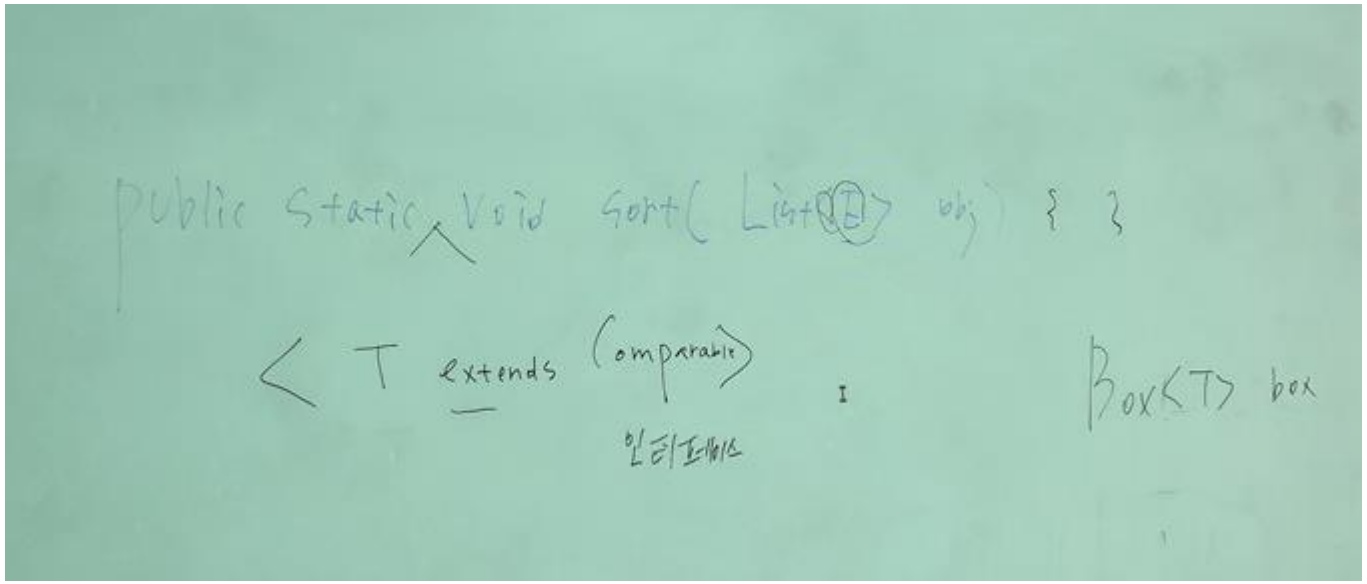

public class GenericExam2 {
// Generic Method
// public static <T extends Number> void printBox(Box<T> obj) {
//   System.out.println(obj.getItem());
// }
// Generic Method : ? : Generic wild card
public static void printBox(Box<? extends Number> obj) {
System.out.println(obj.getItem());
}
Run | Debug
public static void main(String[] args) {
Box<Double> box1 = new Box<>(item:1.0);
printBox(box1);
Box<Integer> box2 = new Box<>(item:1);
printBox(box2);
}
}

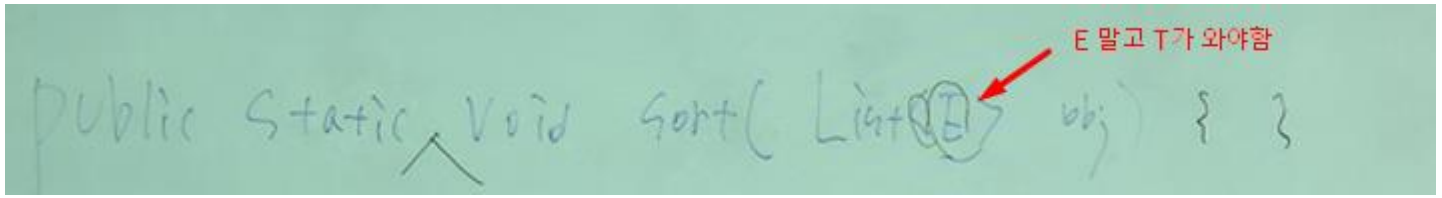
● 제네릭 메소드

```
public static <T  extends Comparable <?  super T>>

                                    void sort (List<T> list)
```

- T는 Comparable 인터페이스를 구현한 타입이어야 한다.


public static void sort(List<E> obj) { }


public static void sort(List<E> obj) { }
< T extends Comparable >
인터페이스
Box<T> box


E 말고 T가 와야함
public static void sort(List<E> obj) { }

Module java.base
Package java.lang

Interface Comparable<T>

Type Parameters:

T - the type of objects that this object may be compared to

All Known Subinterfaces:

ArrayType, ByteValue, CharValue, ChronoLocalDate, ChronoLocalDateTime<D>, Chronology, ChronoZonedDateTime<D>, ClassType, Delayed, DoubleValue, Field, FloatValue, IntegerValue, InterfaceType, LocalVariable, Location, LongValue, Method, Name, Path, ProcessHandle, ReferenceType, RunnableScheduledFuture<V>, ScheduledFuture<V>, ShortValue

All Known Implementing Classes:

AbstractChronology, AbstractRegionPainter.PaintContext.CacheMode, AccessMode, AclEntryFlag, AclEntryPermission, AclEntryType, AssociationChangeNotification.AssocChangeEvent, AttributeTree.ValueKind, Authenticator.RequestorType, BigDecimal, BigInteger, Boolean, Byte, ByteBuffer, Calendar, CardTerminals.State, CaseTree.CaseKind, CatalogFeatures.Feature, CertPathValidatorException.BasicReason, Character, Character.UnicodeScript, CharBuffer, Charset, ChronoField, ChronoUnit, ClientInfoStatus, CLinker.TypeKind, CollationKey, Collector.Characteristics, Component.BaselineResizeBehavior, CompositeName, CompoundName, ConversionComparator.Comparison, CRLReason, CryptoPrimitive, Date, Date, DayOfWeek, Desktop.Action, Diagnostic.Kind, Dialog.ModalExclusionType, Dialog.ModalityType, DirectMethodHandleDesc.Kind, Doclet.Option.Kind, DocletEnvironment.ModuleMode, DocTree.Kind, DocumentationTool.Location, Double, DoubleBuffer, DrbgParameters.Capability, DropMode, Duration, ElementKind, Elements.Origin, ElementType, Enum, File, FileTime, FileVisitOption, FileVisitResult, Float, FloatBuffer, FocusEvent.Cause, FormatStyle, Formatter.BigDecimalLayoutForm, FormSubmitEvent.MethodType, GraphicsDevice.WindowTranslucency, GregorianCalendar, GroupLayout.Alignment, HandlerResult, HijrahChronology, HijrahDate, HijrahEra, HttpClient.Redirect, HttpClient.Version, InquireType, Instant, IntBuffer, Integer, IsoChronology, IsoEra, JapaneseChronology, JapaneseDate, JavaFileObject.Kind, JConsoleContext.ConnectionState, JDBCType, JTable.PrintMode, KeyRep.Type, LambdaExpressionTree.BodyKind, LayoutStyle.ComponentPlacement, LdapName, LinkOption, LocalDate, LocalDateTime, Locale.Category, Locale.FilteringMode, Locale.IsoCountryCode, LocalTime, Long, LongBuffer, MappedByteBuffer

Module java.base
Package java.lang

Class String

java.lang.Object
    java.lang.String

All Implemented Interfaces:

Serializable, CharSequence, Comparable<String>, Constable, ConstantDesc

---

```
public final class String
extends Object
implements Serializable, Comparable<String>, CharSequence, Constable, ConstantDesc
```

The `String` class represents character strings. All string literals in Java programs, such as `"abc"`, are implemented as instances of this class.

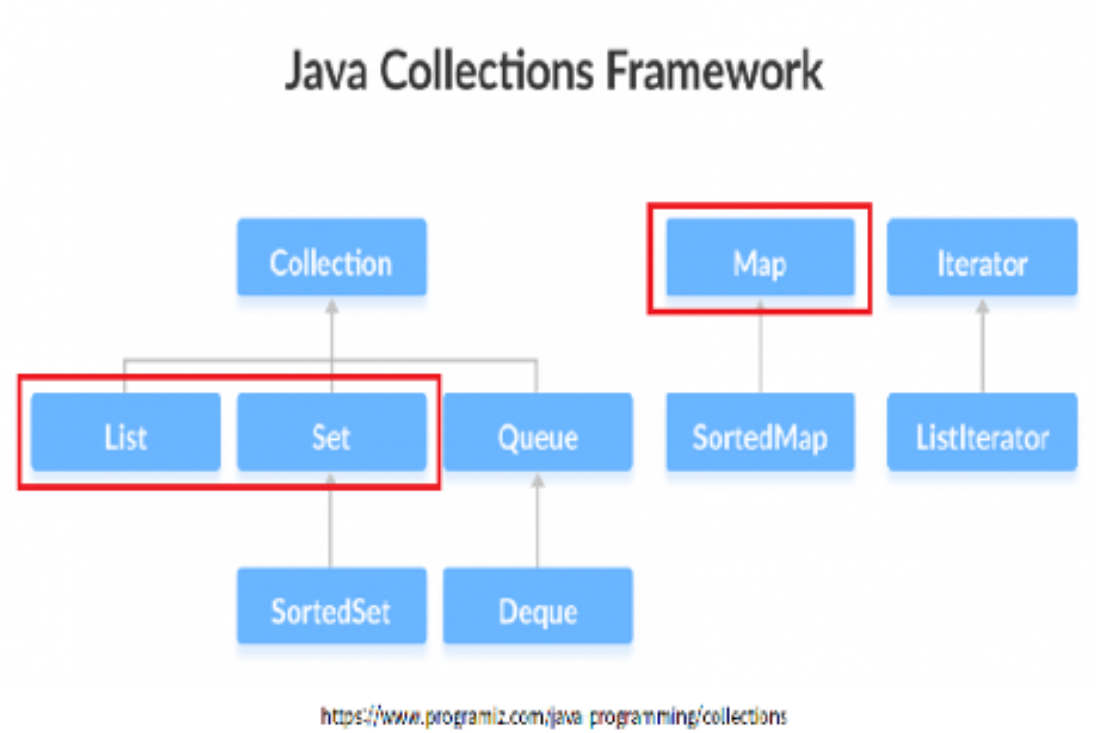
Strings are constant; their values cannot be changed after they are created. String buffers support mutable strings. Because String objects are immutable they can be shared. For example:

```
String str = "abc";
```

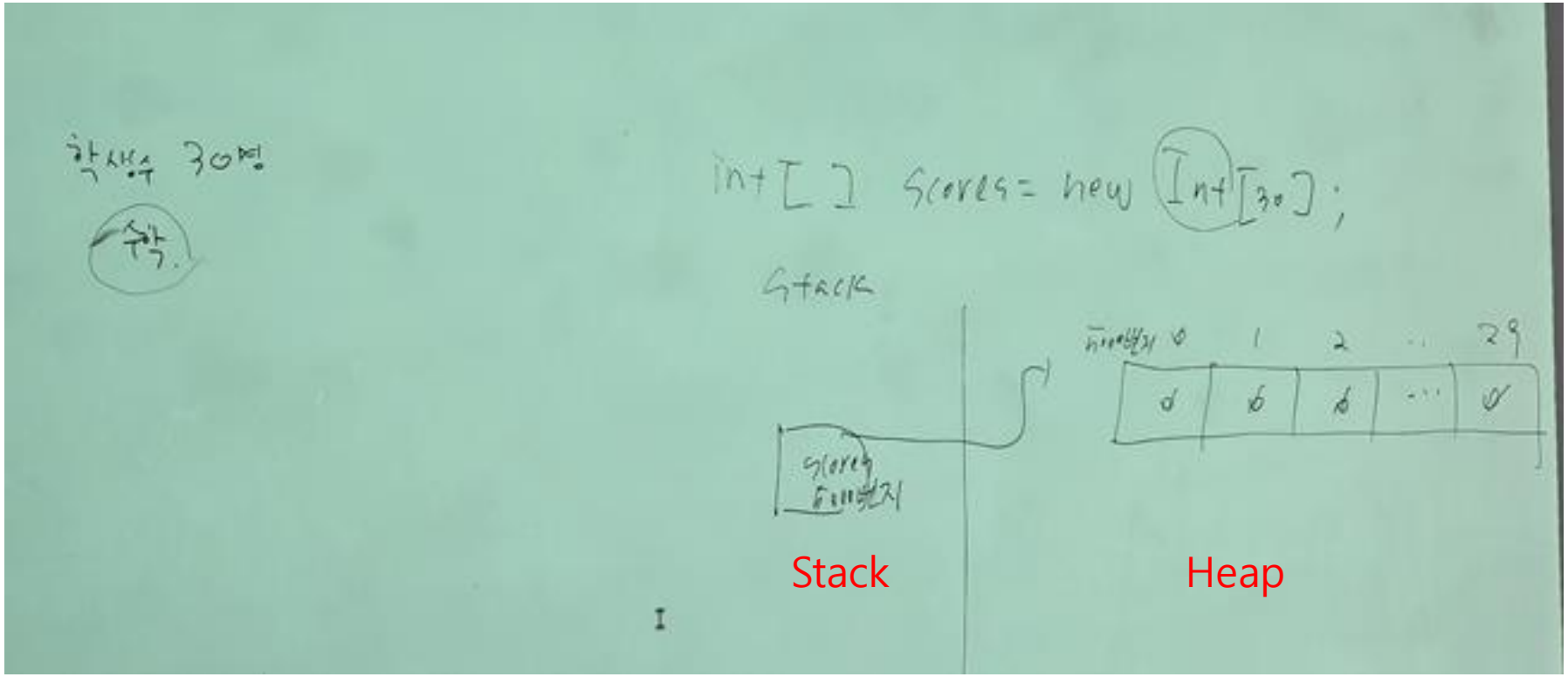
자바 컬렉션 프레임워크

· 다수의 데이터를 쉽고 효과적으로 처리할 수 있는 표준화된 방법을 제공한다.

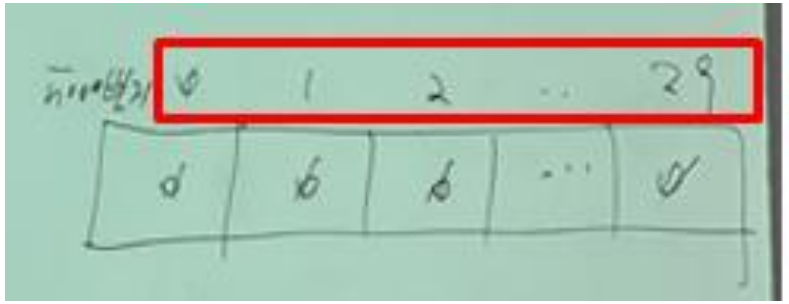
· 데이터를 저장하는 자료 구조와 데이터를 처리하는 알고리즘을 구조화하여 클래스로 구현해 놓은것이다.



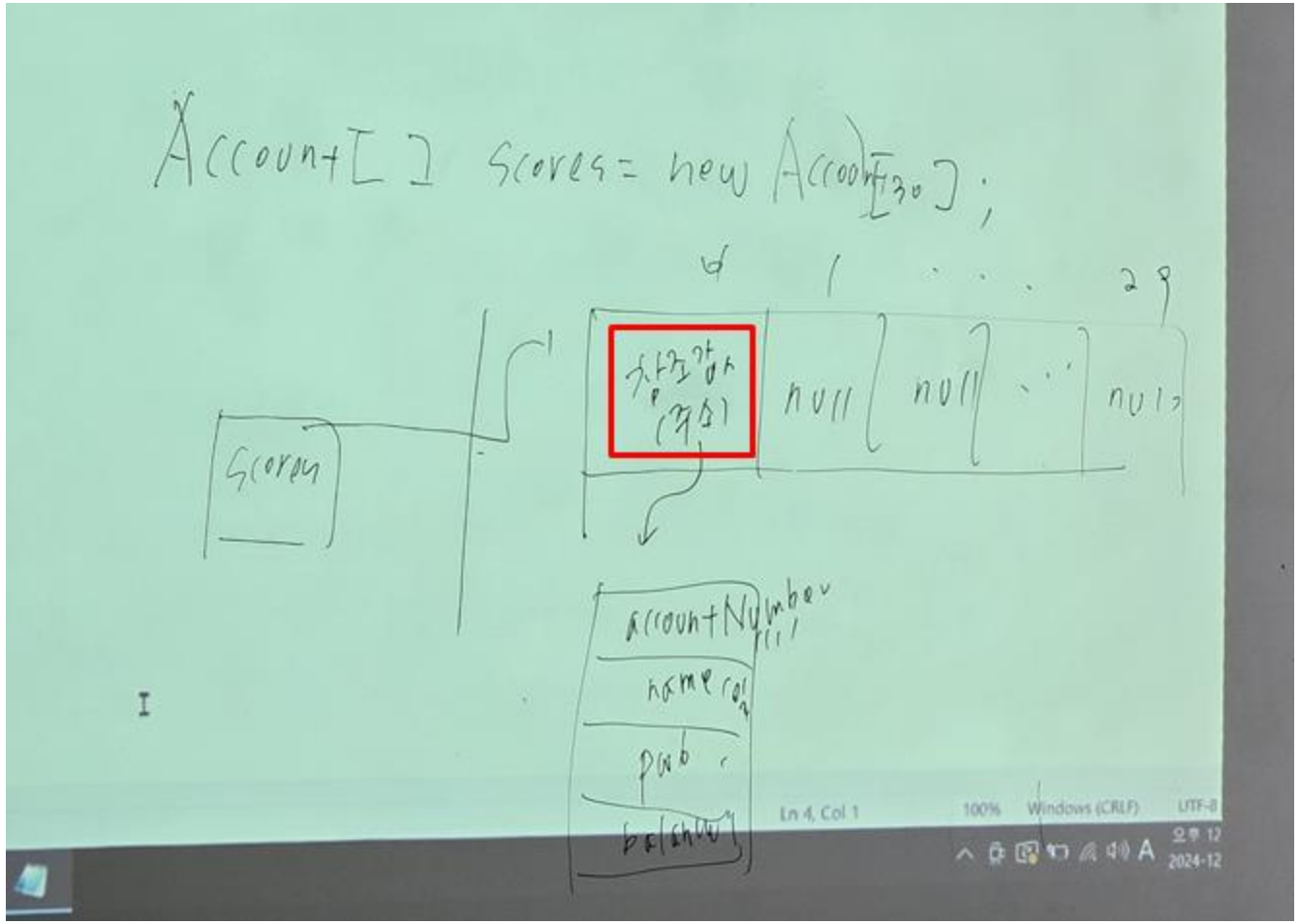
배열



배열은 동일한 데이터타입만 저장 할 수 있다



배열


Account[] scores = new Account[30];
0 1 ... 29
scores
null
null
null
accountNumber
name
pwd
balance
Ln 4, Col 1
100%
Windows (CRLF)
UTF-8
2024-12

kage java.util

age java.util

ains the collections framework, some internationalization support classes, a service loader, properties, random number generation, string parsing and scanning classes, base64 encoding and decoding, a bit array, and several r
y collection classes and legacy date and time classes.

Collections Framework

n overview, API outline, and design rationale, please see:

Collections Framework Documentation

tutorial and programming guide with examples of use of the collections framework, please see:

Collections Framework Tutorial

ated Packages

ule	Package	Description
base	**java.util.concurrent**	Utility classes commonly useful in concurrent programming.
base	**java.util.function**	*Functional interfaces* provide target types for lambda expressions and method references.
base	**java.util.jar**	Provides classes for reading and writing the JAR (Java ARchive) file format, which is based on the standard ZIP file format with an optional manifest file.
logging	**java.util.logging**	Provides the classes and interfaces of the Java 2 platform's core logging facilities.
prefs	**java.util.prefs**	This package allows applications to store and retrieve user and system preference and configuration data.
base	**java.util.random**	This package contains classes and interfaces that support a generic API for random number generation.
base	**java.util.regex**	Classes for matching character sequences against patterns specified by regular expressions.
base	**java.util.spi**	Service provider classes for the classes in the java.util package.
base	**java.util.stream**	Classes to support functional-style operations on streams of elements, such as map-reduce transformations on collections.
base	**java.util.zip**	Provides classes for reading and writing the standard ZIP and GZIP file formats.

Classes and Interfaces | Interfaces | Classes | Enum Classes | Exceptions | Errors

s	Description
ection<E>	The root interface in the *collection hierarchy*.
parator<T>	A comparison function, which imposes a *total ordering* on some collection of objects.

n<E>

collection

rvices, BlockingDeque<E>, BlockingQueue<E>, Deque<E>, EventSet, List<E>, NavigableSet<E>, Queue<E>, Set<E>, SortedSet<E>, TransferQueue<E>

s:

tList, AbstractQueue, AbstractSequentialList, AbstractSet, ArrayBlockingQueue, ArrayDeque, ArrayList, AttributeList, BeanContextServicesSupport, BeanContextSupport, ConcurrentHashMap.KeySetView, Concu
urrentSkipListSet, CopyOnWriteArrayList, CopyOnWriteArraySet, DelayQueue, EnumSet, HashSet, JobStateReasons, LinkedBlockingDeque, LinkedBlockingQueue, LinkedHashSet, LinkedList, LinkedTransferQueue, Pr
eUnresolvedList, Stack, SynchronousQueue, TreeSet, Vector

---

on<E>

ction hierarchy. A collection represents a group of objects, known as its *elements*. Some collections allow duplicate elements and others do not. Some are ordered and others unordered. The JDK does not provide any *direct* im
more specific subinterfaces like `Set` and `List`. This interface is typically used to pass collections around and manipulate them where maximum generality is desired.

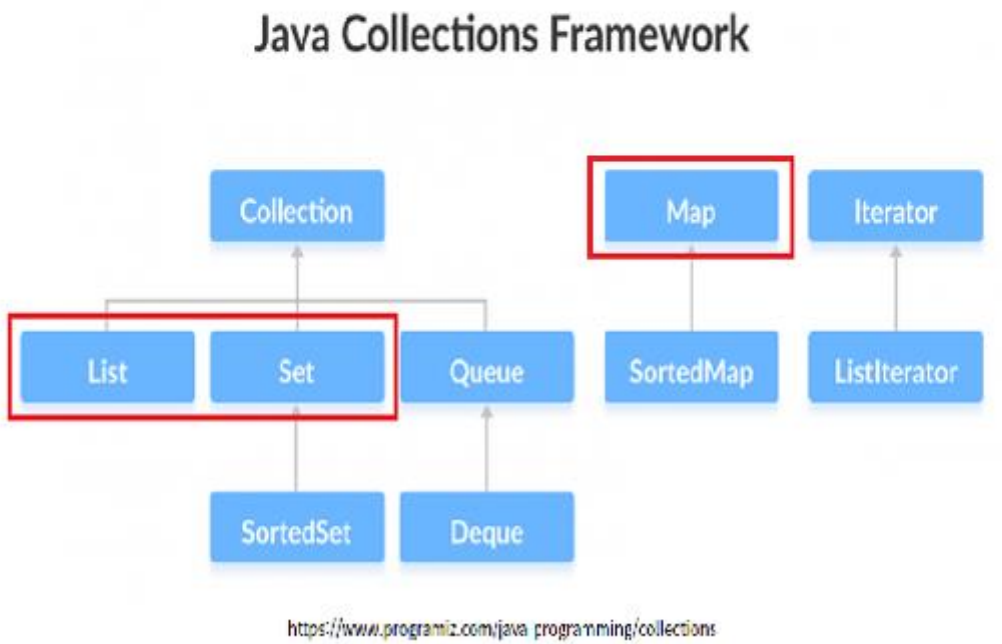
collections that may contain duplicate elements) should implement this interface directly.

배열은 어레이리스트 설명을 위한 부연 설명이었고 다시 본론으로 ~~

자바 컬렉션 프레임워크

· 다수의 데이터를 쉽고 효과적으로 처리할 수 있는 표준화된 방법을 제공한다.

· 데이터를 저장하는 자료 구조와 데이터를 처리하는 알고리즘을 구조화하여 클래스로 구현해 놓은것이다.



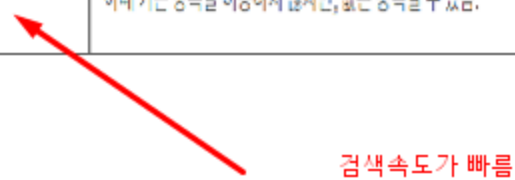
● Collection 인터페이스에서 제공하는 메소드

메소드	설명
boolean add(E e)	해당 컬렉션(collection)에 전달된 요소를 추가함. (선택적 기능)
void clean()	해당 컬렉션의 모든 요소를 제거함. (선택적 기능)
boolean contains(Object o)	해당 컬렉션이 전달된 객체를 포함하고 있는지를 확인함.
boolean equals(Object o)	해당 컬렉션과 전달된 객체가 같은지를 확인함.
boolean isEmpty()	해당 컬렉션이 비어있는지를 확인함.
Iterator<E> iterator()	해당 컬렉션의 반복자(iterator)를 반환함.
boolean remove(Object o)	해당 컬렉션에서 전달된 객체를 제거함. (선택적 기능)
int size()	해당 컬렉션의 요소의 총 개수를 반환함.
Object[] toArray()	해당 컬렉션의 모든 요소를 Object 타입의 배열로 반환함.

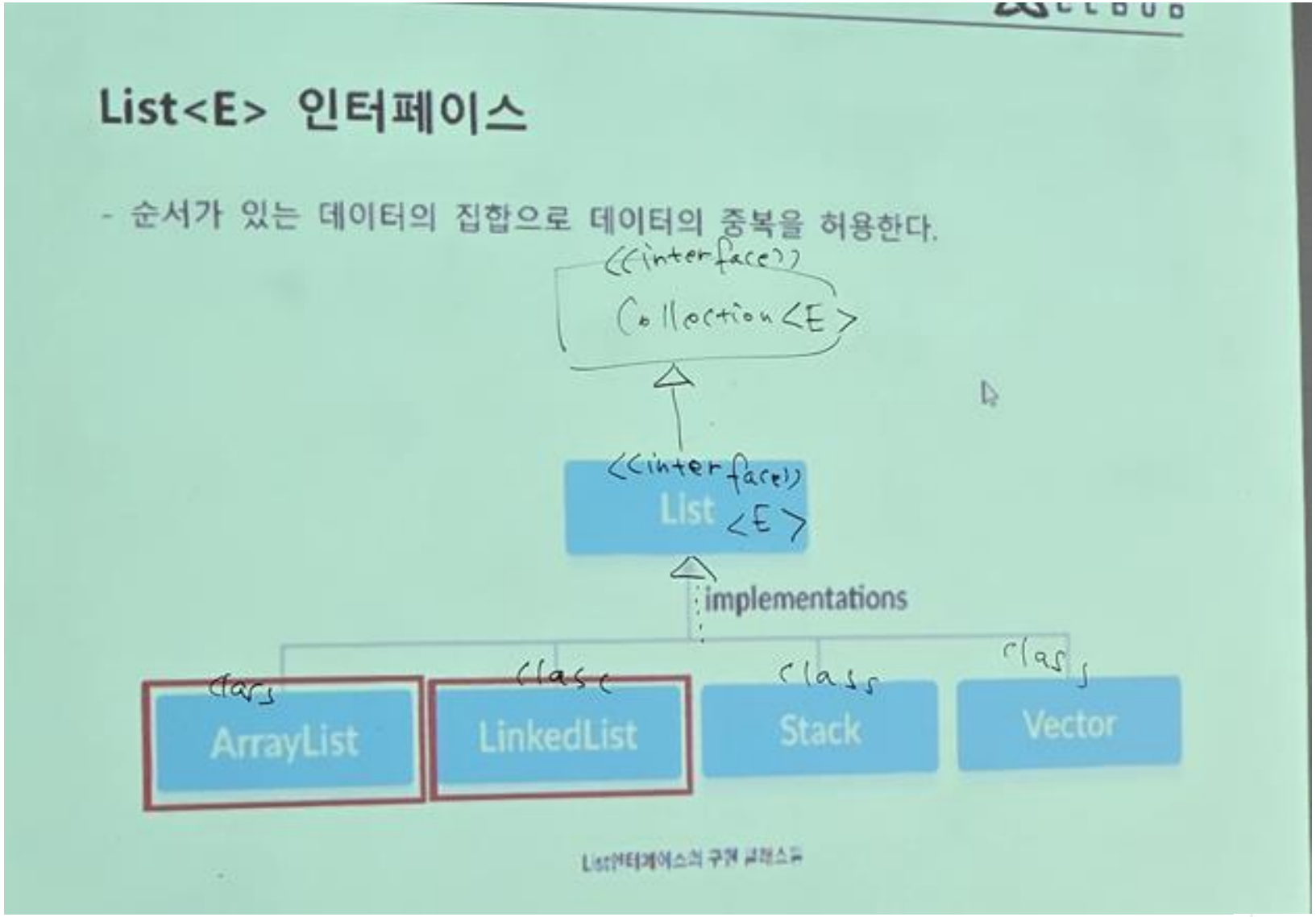
● 주요 인터페이스

인터페이스	설명	구현 클래스
List<E>	순서가 있는 데이터의 집합으로, 데이터의 중복을 허용함.	Vector, ArrayList, LinkedList, Stack, Queue
Set<E>	순서가 없는 데이터의 집합으로, 데이터의 중복을 허용하지 않음.	HashSet, TreeSet
Map<K, V>	키와 값의 한 쌍으로 이루어지는 데이터의 집합으로, 순서가 없음. 이때 키는 중복을 허용하지 않지만, 값은 중복될 수 있음.	HashMap, TreeMap, Hashtable, Properties

Map<K, V>	키와 값의 한 쌍으로 이루어지는 데이터의 집합으로, 순서가 없음. 이때 키는 중복을 허용하지 않지만, 값은 중복될 수 있음.	HashMap, TreeMap, Hashtable, Properties



검색속도가 빠름



List 중 에는 ArrayList 가 가장검색속도가 빠름, 크기변경(가변적) 가능한 객체 배열

All Implemented Interfaces:
Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, Random

Direct Known Subclasses:
AttributeList, RoleList, RoleUnresolvedList

```
public class ArrayList<E>
extends AbstractList<E>
implements List<E>, RandomAccess, Cloneable, Serializable
```

Resizable-array implementation of the List interface. Implements all optio
This class is roughly equivalent to Vector, except that it is unsynchronize

The size, isEmpty, get, set, iterator, and listIterator operations run
compared to that for the LinkedList implementation.

Module java.base
Package java.util

Class ArrayList<E>

java.lang.Object
  java.util.AbstractCollection<E>
    java.util.AbstractList<E>
      java.util.ArrayList<E>

Type Parameters:

E - the type of elements in this list

All Implemented Interfaces:

Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess

Direct Known Subclasses:

AttributeList, RoleList, RoleUnresolvedList

```
public class ArrayList<E>
extends AbstractList<E>
implements List<E>, RandomAccess, Cloneable, Serializable
```

Resizable-array implementation of the List interface. Implements all optional list operations, and permits all elements, including null. In addition to implementing the List interface, this class provides methods to ma
(This class is roughly equivalent to Vector, except that it is unsynchronized.)

The size, isEmpty, get, set, iterator, and listIterator operations run in constant time. The add operation runs in *amortized constant time*, that is, adding n elements requires O(n) time. All of the other operations r
compared to that for the LinkedList implementation.

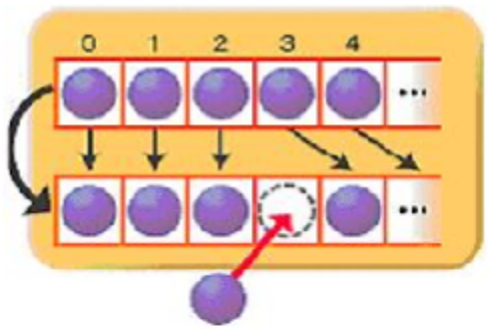
Each ArrayList instance has a *capacity*. The capacity is the size of the array used to store the elements in the list. It is always at least as large as the list size. As elements are added to an ArrayList, its capacity grows
beyond the fact that adding an element has constant amortized time cost.

● ArrayList<E> 클래스

· ArrayList 클래스는 배열을 이용하기 때문에 인덱스를 이용해 배열 요소에 빠르게 접근할 수 있다. (임의의 접근이 빠르다)

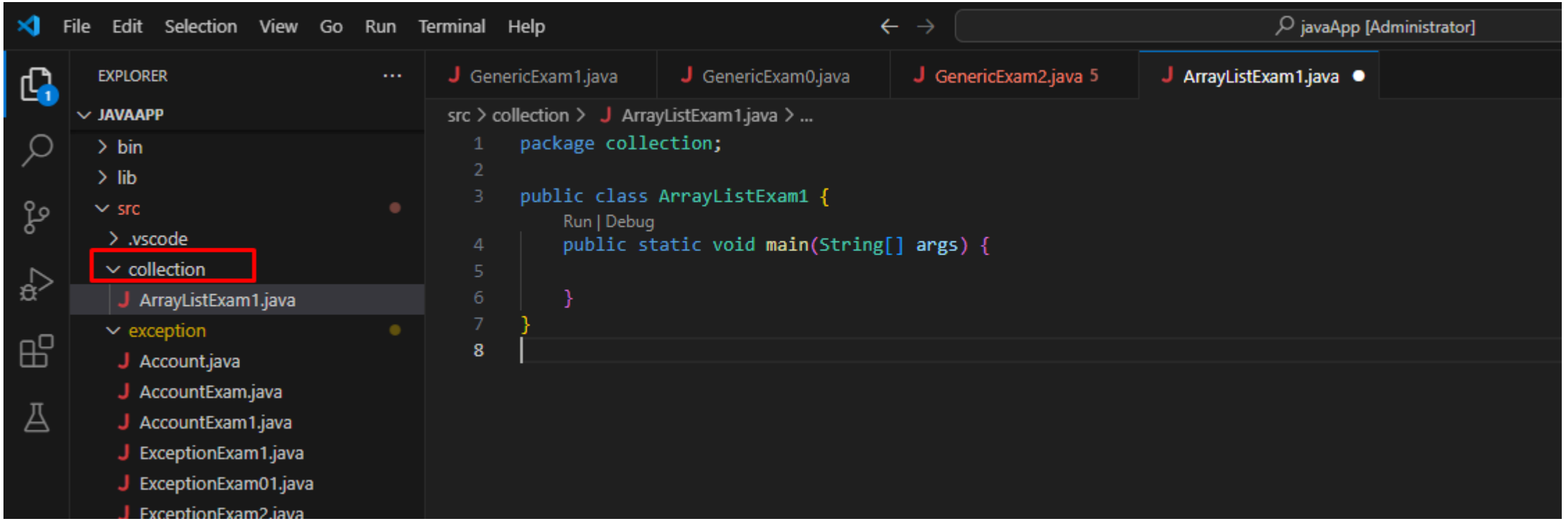
· 기존 배열과 차이점은 배열의 크기가 가변적이다.

· 요소의 추가 및 삭제 시 성능이 느리다는 단점을 갖고 있다.

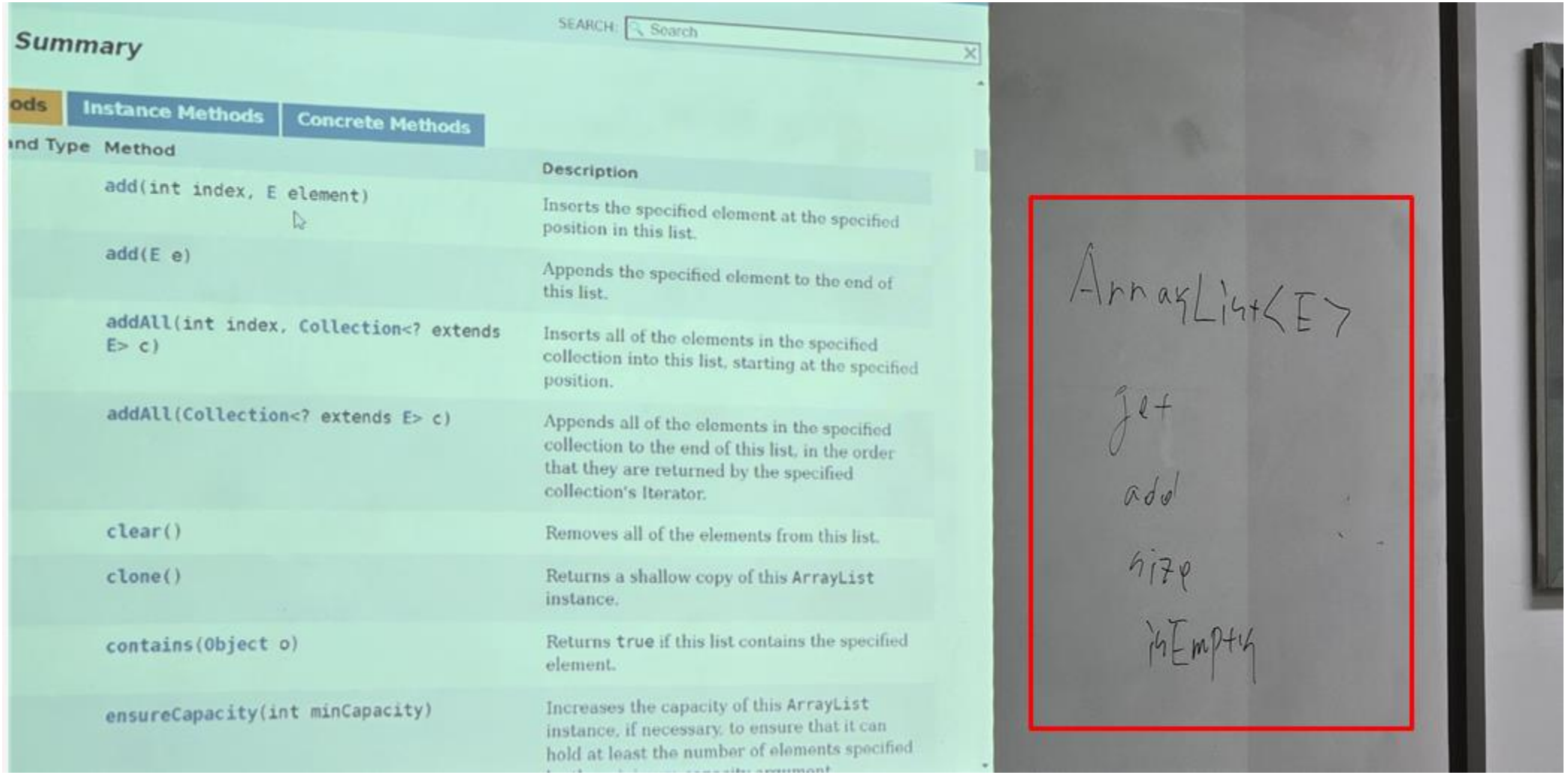


[문법]

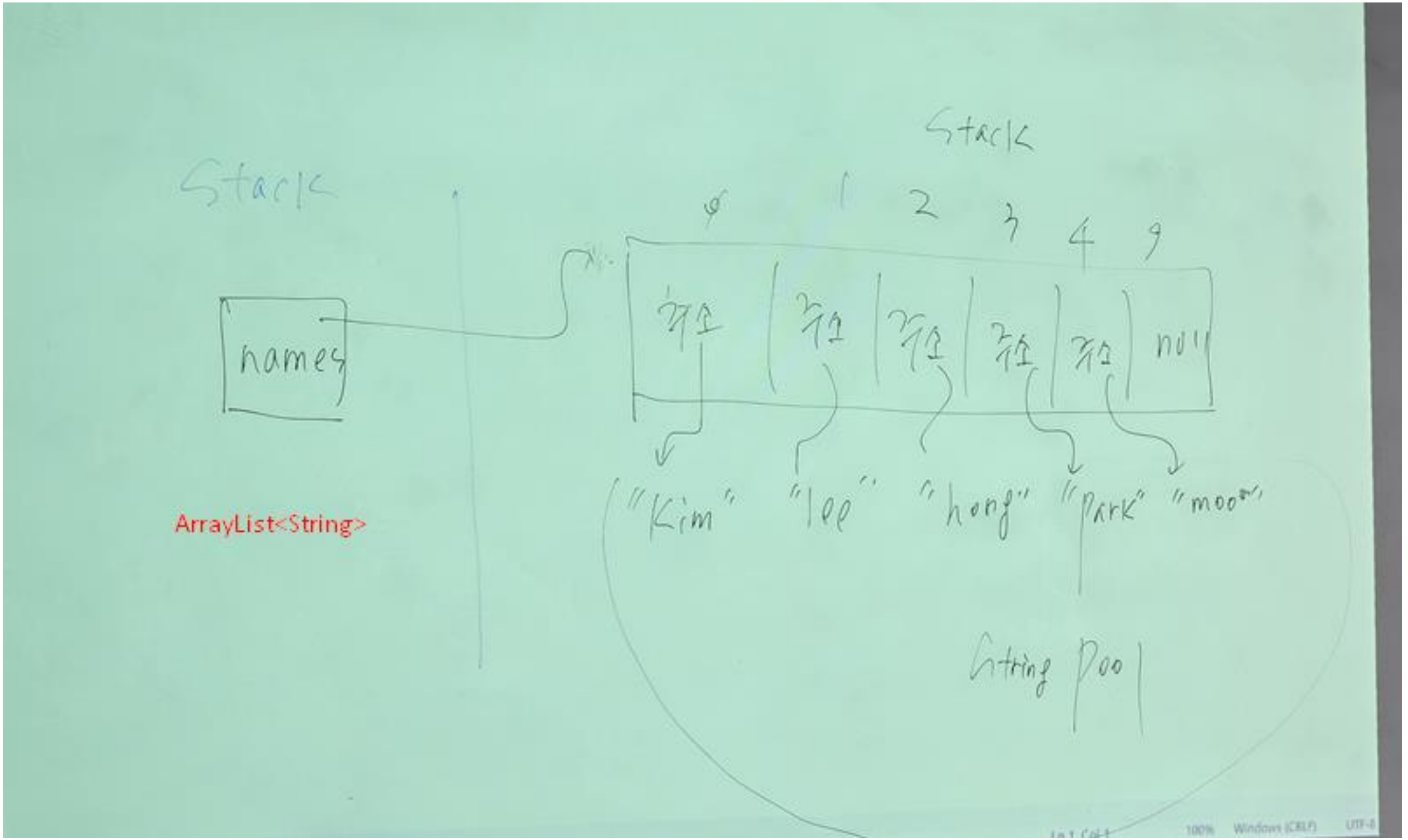
```
ArrayList<클래스타입>  list = new ArrayList<클래스타입>();
```



패키지는 소문자로 해야한다, 오류는 안나지만 룰 이다



Add는 E String니까 문자열만 저장 할 수 있다 정도


Stack
Stack
0 1 2 3 4 9
주소
주소
주소
주소
주소
null
names
"Kim" "lee" "hong" "park" "moon"
String Pool
ArrayList<String>

GenericExam1.java | GenericExam2.java | ArrayListExam1.java

src > collection > ArrayListExam1.java > ...

```java
package collection;

import java.util.ArrayList;

public class ArrayListExam1 {
    Run | Debug
    public static void main(String[] args) {

        ArrayList<String> names = new ArrayList<>();

        names.add(e:"kim");
        names.add(e:"lee");
        names.add(e:"hong");
        names.add(e:"park");
        names.add(e:"moon");
        // names.add("1");  <- 이건 Error

        for(String name : names) {
            System.out.println(name);
        }

    }
}
```


```java
package collection;

import java.util.ArrayList;

public class ArrayListExam1 {
    public static void main(String[] args) {

        ArrayList<String> names = new ArrayList<>();

        names.add("kim");
        names.add("lee");
        names.add("hong");
        names.add("park");
        names.add("moon");
        // names.add("1");  <- 이건 Error

        for(String name : names) {
            System.out.println(name);
        }

    }
}
```

NullPointerException - if the specified array is null

get

public E get(int index)

Returns the element at the specified position in this list.

Specified by:

get in interface List<E>

Specified by:

get in class AbstractList<E>

Parameters:

index - index of the element to return

Returns:

the element at the specified position in this list

Throws:

IndexOutOfBoundsException - if the index is out of range (index < 0 || index >= size())

set

public E set(int index,
E element)

Replaces the element at the specified position in this list with the specified element.

Specified by:

set in interface List<E>

Overrides:

set in class AbstractList<E>

Parameters:

index - index of the element to replace

함수형 인터페이스

```java
public class ArrayListExam1 {
    Run | Debug
    public static void main(String[] args) {

        ArrayList<String> names = new ArrayList<>();

        names.add(e:"ki
        names.add(e:"le
        names.add(e:"ho
        names.add(e:"pa
        names.add(e:"mo
        // names.add("1

        // for(String n
        //     System.o
        // }
        names.stream().forEach(System.out::println);  // 함수형 코드, 요즘 많이 쓰는 코딩 기법
    }
}
```

```
void java.util.stream.Stream.forEach(Consumer<? super String> action)
```

Performs an action for each element of this stream.

This is a terminal operation.

The behavior of this operation is explicitly nondeterministic. For parallel stream pipelines, this operation does *not* guarantee to respect the encounter order of the stream, as doing so would sacrifice the benefit of parallelism. For any given element, the action may be performed at whatever time and in whatever thread the library chooses. If the action accesses shared state, it is responsible for providing the required synchronization.

- **Parameters:**
  - **action** a non-interfering action to perform on the elements

GenericExam1.java | GenericExam2.java | ArrayListExam1.java | ArrayListExam1cp.java

src > collection > ArrayListExam1.java > ...

```java
package collection;

import java.util.ArrayList;

public class ArrayListExam1 {
    Run | Debug
    public static void main(String[] args) {

        ArrayList<String> names = new ArrayList<>();

        names.add(e:"kim");
        names.add(e:"lee");
        names.add(e:"hong");
        names.add(e:"park");
        names.add(e:"moon");
        // names.add("1");  <- 이건 Error

        // for(String name : names) {
        //     System.out.println(name);
        // }
        names.stream().forEach(System.out::println);  // 함수형 코드, 요즘 많이 쓰는 코딩 기법
    }
}

```

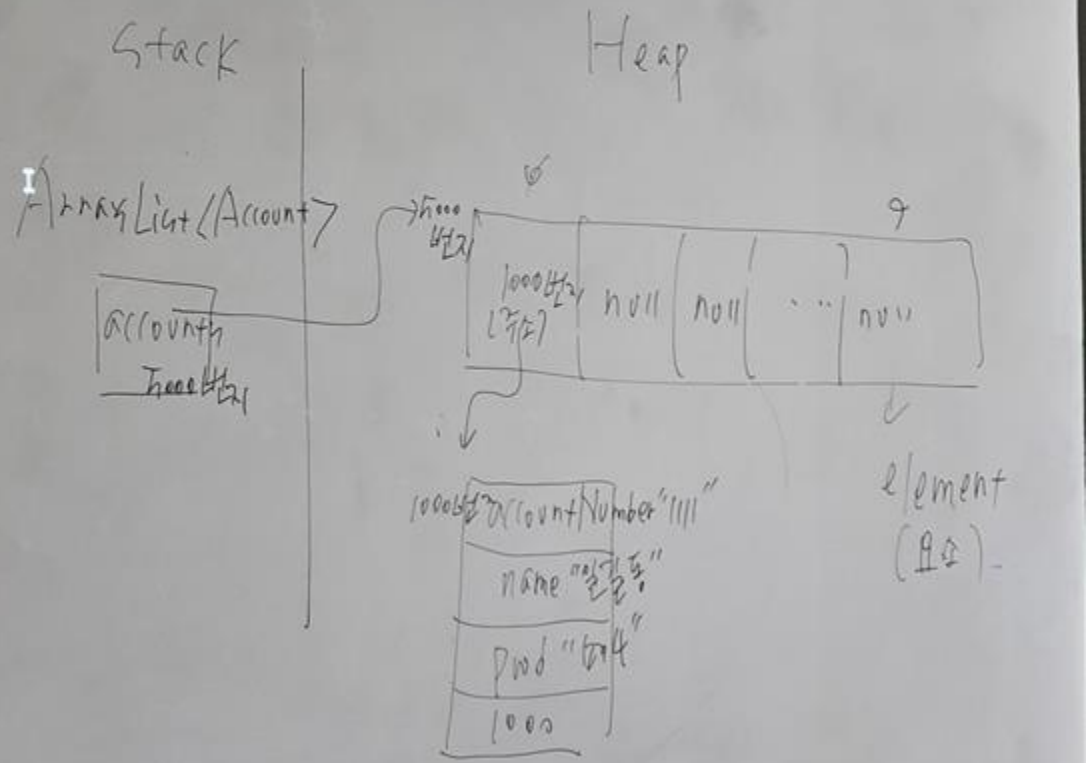
```java
package collection;

import java.util.ArrayList;

public class ArrayListExam1 {
    public static void main(String[] args) {

        ArrayList<String> names = new ArrayList<>();

        names.add("kim");
        names.add("lee");
        names.add("hong");
        names.add("park");
        names.add("moon");
        // names.add("1");  <- 이건 Error

        // for(String name : names) {
        //     System.out.println(name);
        // }
        names.stream().forEach(System.out::println);  // 함수형 코드, 요즘 많이 쓰는 코딩 기법
    }
}
```

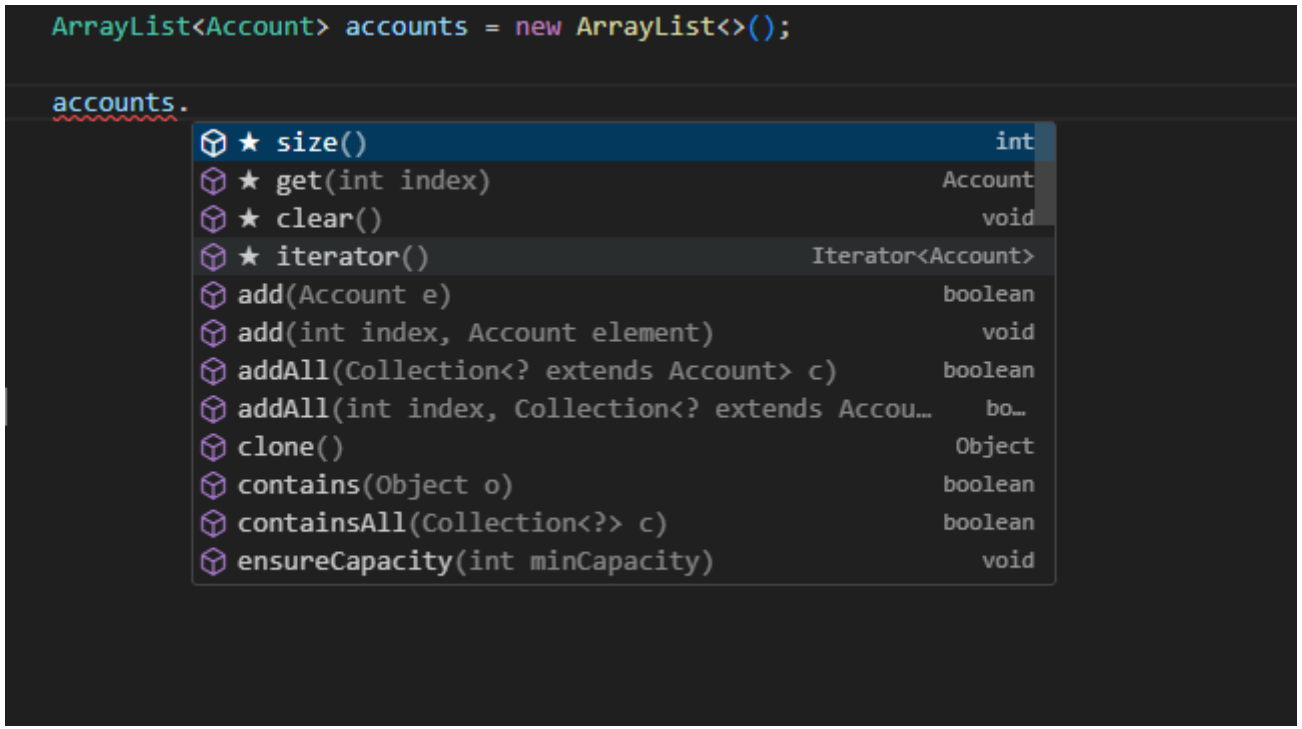
```java
package collection;

import java.util.ArrayList;

public class ArrayListExam2 {

    Run | Debug
    public static void main(String[] args) {

        ArrayList<Account> accounts = new ArrayList<>();

    }

}
```

Element Type

원소 데이터

```java
package collection;

import java.util.ArrayList;

public class ArrayListExam2 {
    Run | Debug
    public static void main(String[] args) {

        ArrayList<Account> accounts = new ArrayList<>();

        accounts.add(new Account(accountNumber:"1111", name:"일길동", pwd:"1234", balance:1000));

    }
}
```




ArrayList<Account> accounts = new ArrayList<>();
accounts.
★ size() int
★ get(int index) Account
★ clear() void
★ iterator() Iterator<Account>
add(Account e) boolean
add(int index, Account element) void
addAll(Collection<? extends Account> c) boolean
addAll(int index, Collection<? extends Accou… bo…
clone() Object
contains(Object o) boolean
containsAll(Collection<?> c) boolean
ensureCapacity(int minCapacity) void

Module java.base
Package java.util

Interface Map<K,V>

Type Parameters:

K - the type of keys maintained by this map

V - the type of mapped values

All Known Subinterfaces:

Bindings, ConcurrentMap<K,V>, ConcurrentNavigableMap<K,V>, NavigableMap<K,V>, SortedMap<K,V>

All Known Implementing Classes:

AbstractMap, Attributes, AuthProvider, ConcurrentHashMap, ConcurrentSkipListMap, EnumMap, HashMap, Hashtable, Headers, IdentityHashMap, LinkedHashMap, PrinterStateReasons, Properties, Provider, R
UIDefaults, WeakHashMap

```
public interface Map<K,V>
```

An object that maps keys to values. A map cannot contain duplicate keys; each key can map to at most one value.

This interface takes the place of the Dictionary class, which was a totally abstract class rather than an interface.

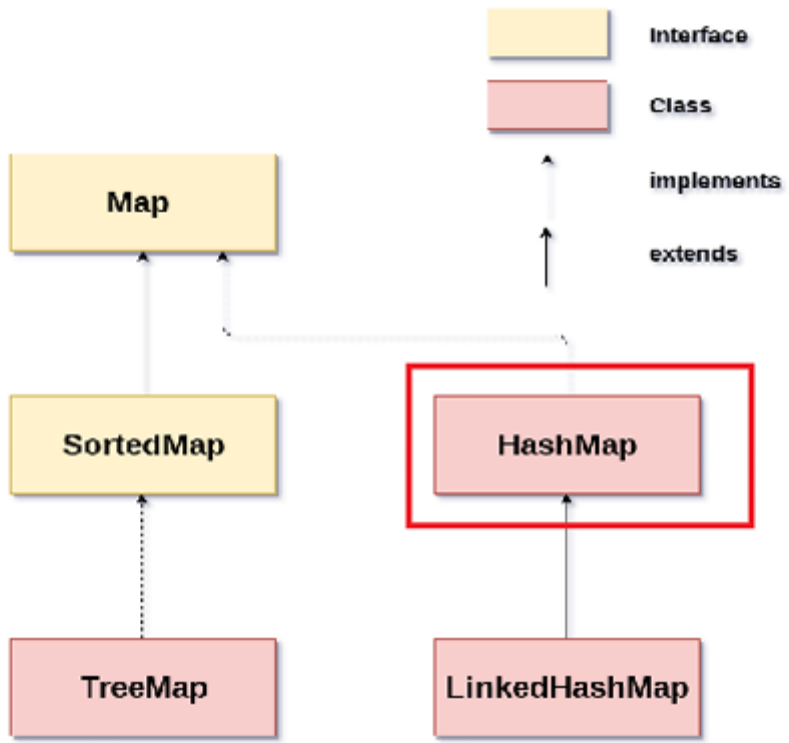
The Map interface provides three *collection views*, which allow a map's contents to be viewed as a set of keys, collection of values, or set of key-value mappings. The *order* of a map is defined as the order in which the map implementations, like the TreeMap class, make specific guarantees as to their order; others, like the HashMap class, do not.

Note: great care must be exercised if mutable objects are used as map keys. The behavior of a map is not specified if the value of an object is changed in a manner that affects equals comparisons while the object is permissible for a map to contain itself as a key. While it is permissible for a map to contain itself as a value, extreme caution is advised: the equals and hashCode methods are no longer well defined on such a map.

Map<K, V> 인터페이스

· 키와 값의 한쌍으로 이루어지는 한 쌍의 집합으로 순서가 없음.

· 키는 중복을 허용하지 않으나 값은 중복될 수 있음.



Module java.base
Package java.util

Class HashMap<K,V>

java.lang.Object
    java.util.AbstractMap<K,V>
        java.util.HashMap<K,V>

Type Parameters:

K - the type of keys maintained by this map

V - the type of mapped values

All Implemented Interfaces:

Serializable, Cloneable, Map<K,V>

Direct Known Subclasses:

LinkedHashMap, PrinterStateReasons

```
public class HashMap<K,V>
extends AbstractMap<K,V>
implements Map<K,V>, Cloneable, Serializable
```

Hash table based implementation of the Map interface. This implementation provides all of the optional map operations, and permits null values and the null key. (The HashMap class is roughly equivalent to Hashtable, exc
makes no guarantees as to the order of the map; in particular, it does not guarantee that the order will remain constant over time.

This implementation provides constant-time performance for the basic operations (get and put), assuming the hash function disperses the elements properly among the buckets. Iteration over collection views requires time
number of buckets) plus its size (the number of key-value mappings). Thus, it's very important not to set the initial capacity too high (or the load factor too low) if iteration performance is important.

An instance of HashMap has two parameters that affect its performance: *initial capacity* and *load factor*. The *capacity* is the number of buckets in the hash table, and the initial capacity is simply the capacity at the time the h
the hash table is allowed to get before its capacity is automatically increased. When the number of entries in the hash table exceeds the product of the load factor and the current capacity, the hash table is *rehashed* (that is
approximately twice the number of buckets.

As a general rule, the default load factor (.75) offers a good tradeoff between time and space costs. Higher values decrease the space overhead but increase the lookup cost (reflected in most of the operations of the HashMa
in the map and its load factor should be taken into account when setting its initial capacity, so as to minimize the number of rehash operations. If the initial capacity is greater than the maximum number of entries divided b

If many mappings are to be stored in a HashMap instance, creating it with a sufficiently large capacity will allow the mappings to be stored more efficiently than letting it perform automatic rehashing as needed to grow the
sure way to slow down performance of any hash table. To ameliorate impact, when keys are Comparable, this class may use comparison order among keys to help break ties.

put

```
public V put(K key,
             V value)
```

Associates the specified value with the specified key in this map. If the map previously contained a mapping for the key, the old value is replaced.

Specified by:

put in interface Map<K,V>

Overrides:

put in class AbstractMap<K,V>

Parameters:

key - key with which the specified value is to be associated

value - value to be associated with the specified key

Returns:

the previous value associated with key, or null if there was no mapping for key. (A null return can also indicate that the map previously associated null with key.)

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	clear()	Removes all of the mappings from this map.
Object	clone()	Returns a shallow copy of this HashMap instance: the keys and values themselves are not clo
V	compute(K key, BiFunction<? super K,? super V,? extends V> remappingFunction)	Attempts to compute a mapping for the specified key and its current mapped value (or null
V	computeIfAbsent(K key, Function<? super K,? extends V> mappingFunction)	If the specified key is not already associated with a value (or is mapped to null), attempts t and enters it into this map unless null.
V	computeIfPresent(K key, BiFunction<? super K,? super V,? extends V> remappingFunction)	If the value for the specified key is present and non-null, attempts to compute a new mappi
boolean	containsKey(Object key)	Returns true if this map contains a mapping for the specified key.
boolean	containsValue(Object value)	Returns true if this map maps one or more keys to the specified value.
Set<Map.Entry<K,V>>	entrySet()	Returns a Set view of the mappings contained in this map.
V	get(Object key)	Returns the value to which the specified key is mapped, or null if this map contains no map
boolean	isEmpty()	Returns true if this map contains no key-value mappings.
Set<K>	keySet()	Returns a Set view of the keys contained in this map.
V	merge(K key, V value, BiFunction<? super V,? super V,? extends V> remappingFunction)	If the specified key is not already associated with a value or is associated with null, associat
V	put(K key, V value)	Associates the specified value with the specified key in this map.
void	putAll(Map<? extends K,? extends V> m)	Copies all of the mappings from the specified map to this map.
V	remove(Object key)	Removes the mapping for the specified key from this map if present.