

1811943 황성미 5일차 과제

▼ 3.5 Interface

함수의 틀을 정의한 것

이름과 변수에 대한 정의만 하고 execution code는 없음(사람들과의 상호작용을 위해 개념을 맞추기 위함)

▼ 인터페이스와 연관된 단어

1. IDL(함수의 이름과 속성을 정의한 것)

JAVA RMI(메서드)

C RPC(프로시저)

⇒ 다 “함수”를 지칭!(procedure, function, module, method)

2. API

응용 프로그램을 개발할 때 필요한 함수(= 인터페이스)의 이름

예를 들어 통계에서 필요한 함수 중 평균을 사용하고 싶은데 평균에 대한 함수가 미리 만들어놓은 걸 API라 부름

3. called : 함수의 이름(불러주는 대상이 됨)

calling

4. 라이브러리 : 함수 집합체

5. JAR : 자바 클래스들의 모음(메서드, 함수)

▼ 예시

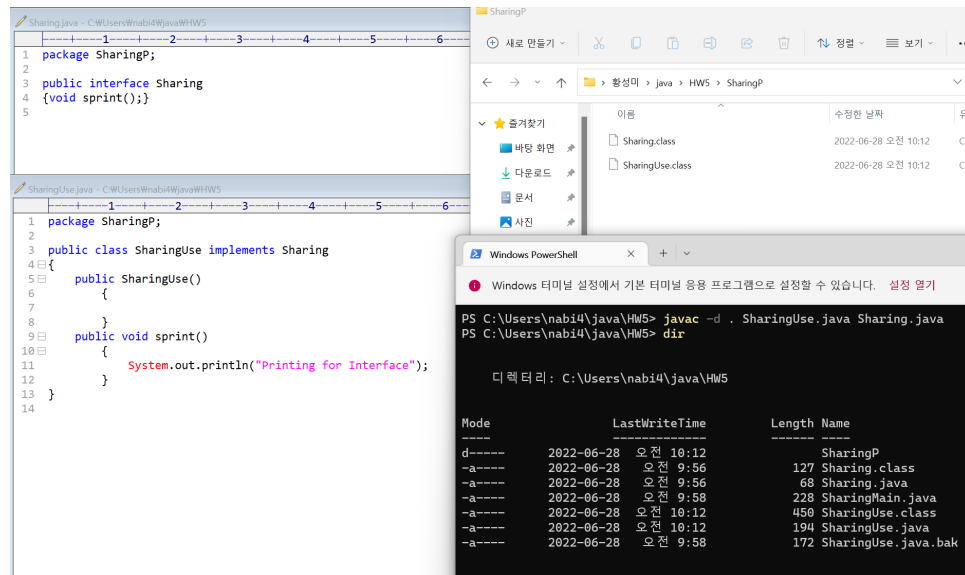
Interface Student { study(); play(); } 로 정의

개발자가 class My_Class implements Student{ study() {.....} play() {.....}}

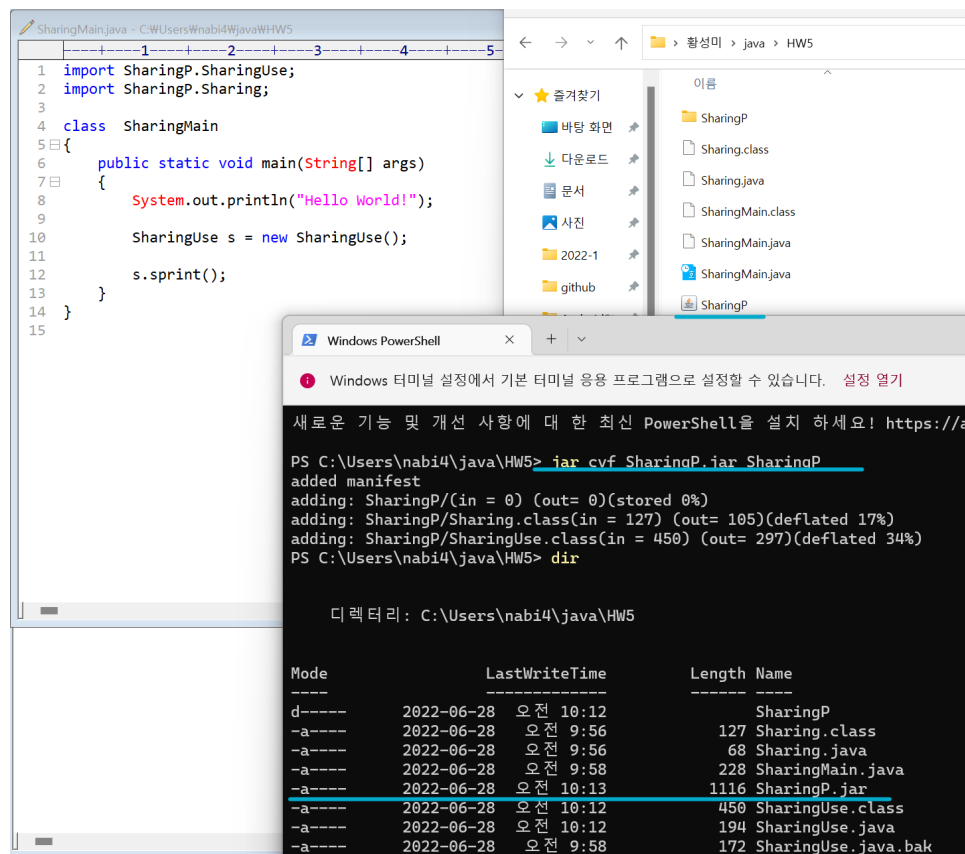
처럼

Student를 상속받아 execution code를 작성

▼ Lab3.5



Sharing.java , SharingUse.java를 SharingP 패키지로 만들어줌



jar cvf SharingP.jar SharingP
를 통해 공유를 위한 jar 파일이 생성됨

```

1 import SharingP.SharingUse;
2 import SharingP.Sharing;
3
4 class SharingMain
5 {
6     public static void main(String[] args)
7     {
8         System.out.println("Hello World!");
9
10        SharingUse s = new SharingUse();
11
12        s.print();
13    }
14 }
15

```

```

Mode                LastWriteTime         Length Name
-----
d-----         2022-06-28 오전 10:12             SharingP
-a-----         2022-06-28 오전 9:56             127 Sharing.class
-a-----         2022-06-28 오전 9:56             68 Sharing.java
-a-----         2022-06-28 오전 9:58             228 SharingMain.java
-a-----         2022-06-28 오전 10:13            1116 SharingP.jar
-a-----         2022-06-28 오전 10:12            450 SharingUse.class
-a-----         2022-06-28 오전 10:12            194 SharingUse.java
-a-----         2022-06-28 오전 9:58            172 SharingUse.java.bak

PS C:\Users\nabi4\java\HW5> javac SharingMain.java -classpath SharingP.jar
SharingMain.java:12: error: cannot find symbol
    s.print();
      ^
symbol:   method print()
location: variable s of type SharingUse
1 error

PS C:\Users\nabi4\java\HW5> javac SharingMain.java -classpath SharingP.jar
PS C:\Users\nabi4\java\HW5> java SharingMain
Hello World!
Printing for Interface

```

s.print()라는 오류를 발견하여 s.sprint()로 변경
javac SharingMain.java -classpath SharingP.jar 를 통해 jar 파일을 연동시킨 후
java SharingMain을 통해 실행됨

+))

```

1 package SharingP;
2
3 public interface Sharing
4 {void sprint();}
5

```

```

1 package SharingP;
2
3 public class SharingUse implements Sharing
4 {
5     public SharingUse()
6     {
7     }
8
9     public void print()
10    {
11        System.out.println("Printing for Interface");
12        System.out.println("황성미");
13    }
14 }
15

```

```

1 import SharingP.SharingUse;
2 import SharingP.Sharing;
3
4 class SharingMain
5 {
6     public static void main(String[] args)
7     {
8         System.out.println("Hello World!");
9         System.out.println("1811943");
10
11        SharingUse s = new SharingUse();
12
13        s.sprint();
14    }
15 }
16

```

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 기능 및 개선 사항에 대한 최신 PowerShell을 설치하세요! https://aka.ms/

PS C:\Users\nabi4\java\HW5> javac -d . SharingUse.java Sharing.java
PS C:\Users\nabi4\java\HW5> jar cvf SharingP.jar SharingP
added manifest
adding: SharingP/(in = 0) (out= 0)(stored 0%)
adding: SharingP/Sharing.class(in = 127) (out= 105)(deflated 17%)
adding: SharingP/SharingUse.class(in = 477) (out= 328)(deflated 31%)
PS C:\Users\nabi4\java\HW5> javac SharingMain.java -classpath SharingP.jar
PS C:\Users\nabi4\java\HW5> java SharingMain
Hello World!
1811943
Printing for Interface
황성미
PS C:\Users\nabi4\java\HW5>

```

SharingUse.java와 SharingMain.java 파일에 이름과 학번을 출력하는 코드를 덧붙여 학번과 이름까지 출력해봄.

▼ 3.6 static and final

▼ static

변경하면 안되는 값들을 고정시킴

함수 자체를 static을 사용해도 되지만 멤버 필드를 static으로 하는 경우가 많음

상속을 해도 오버라이딩할 수 없음

▼ 멤버 필드

객체들 간의 전역 변수처럼 사용

클래스 이름으로 접근 가능, 클래스 변수라고도 함

▼ 메서드

static으로 선언한 클래스 메서드에서는 오버라이딩되지 않음

클래스 메서드라고도 하며 인스턴스가 아닌 클래스 이름으로 메서드를 호출할 수 있음

메서드에서는 super, this를 사용할 수 없고, static이 아닌 멤버 필드는 접근할 수 없음

▼ final

상속을 통해서 재사용하지 않게 하고 싶을때

final과 abstract 클래스는 동시에 선언될 수 없음

▼ 메서드

오버라이딩 불가

▼ 멤버 필드

값 변경 불가, 상수의 의미로 사용됨

▼ 3.7 Inner Class

클래스 안에 정의된 클래스

▼ Member class

클래스의 멤버로 내부 클래스가 선언된 경우

자신을 포함하는 클래스의 멤버 필드와 메서드들을 자유롭게 접근 가능

▼ Local class

자바 코드 블록 안에서 정의된 내부 클래스

정의된 블록 안에서만 사용 가능

자신을 포함하는 클래스의 멤버 필드와 메서드 자유롭게 접근 가능. 지역 변수도 접근 가능하지만 이때 지역변수가 final로 선언되어있어야 함

▼ Non-named class

메서드 내에서 정의

클래스를 정의하면서 바로 사용하는 방식

이벤트 핸들링을 하기 위해 만들어짐(인터페이스에서 정의한 함수를 구현할 때 inner class가 만들어짐)

이름이 있는 inner class(여러 번 사용할 때), 없는 inner class(한 번만 사용하고 끝날 때)가 만들어질 수 있음

클래스 이름 앞에 \$표시가 있으면 inner class임

세미콜론으로 끝남

→ 7.2 이벤트 프로그래밍에서 더 자세히

```
class EventTestMyFrame2 extends JFrame {
    private JButton button;
    private JLabel label;

    public EventTestMyFrame2() {
        this.setSize(300, 200);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setTitle("Event Programming Style 2");

        JPanel panel = new JPanel();
        button = new JButton("Button");
        label = new JLabel("Please, Click the Button");
        button.addActionListener(new MyListener());
        panel.add(button);
        panel.add(label);
        this.add(panel);
        this.setVisible(true);
    }

    /* Inner Class, named MyListener */
    private class MyListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            if (e.getSource() == button) {
                label.setText("Clicked the Button");
            }
        }
    }
}
```

a named inner class

```
class EventTestMyFrame3 extends JFrame {
    private JButton button;
    private JLabel label;

    public EventTestMyFrame3() {
        this.setSize(300, 200);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setTitle("Event Programming Style 3");

        JPanel panel = new JPanel();
        button = new JButton("Button");
        label = new JLabel("Please, Click the Button");
        // implements for event
        button.addActionListener(new ActionListener() {
            /* 무명식 Class */
            public void actionPerformed(ActionEvent e) {
                if (e.getSource() == button) {
                    label.setText("Clicked the Button");
                }
            }
        });
        panel.add(button);
        panel.add(label);
        this.add(panel);
        this.setVisible(true);
    }
}
```

an unnamed inner class

왼쪽의 MyListener코드가 named inner class

보통 우리는 GUI가 작동하면서 어떤 코드를 실행할 것인가에 대해 배울 것이기 때문에 오른쪽의 unnamed inner class 형태를 많이 사용할 것임

▼ 3.8 Import classes in java

상당히 많은 클래스에 대한 공부 필요함!

▼ Object class

모든 자바 클래스의 슈퍼 클래스

특별히 extends를 사용하지 않아도 자동적으로 상속

object class의 모든 메서드는 모든 자바 클래스에서 사용 가능

▼ Data Type class

Boolean : boolean 기본 타입을 표현하기 위한 클래스

Byte : byte 기본 자료 타입을 표현하기 위한 클래스

Character : Character 형태의 생성자를 가짐

Double : Double 생성자를 가짐

Integer : Integer 두 개의 생성자 존재

Short : Short 두 개의 생성자 존재

▼ Math class

수학 함수와 이에 필요한 상수들로 구성

모든 멤버 필드 및 메서드들이 final로 정의되어있어 Math 클래스를 상속해서 사용하거나 오버라이드 할 수 없음

▼ String class

문자열을 나타내기 위해 사용

문자열의 내용이나 길이를 변경할 수 없음

▼ StringTokenizer class

문자열을 개별적인 토큰으로 분리하는 방법 제공

java.util 패키지에 있음

▼ Vector class

동적으로 크기가 변하는 배열의 일종

java.util 클래스에 있음

▼ 3.9 Exception Handling in Java(중요)

발생하는 에러에 대해 해결 방법을 제시해주는 것

resilient code를 사용하여 예외 처리

어떤 조건이 되었을때 죽지않고 계속 프로그램이 진행될 수 있도록 함

에러의 종류(error_내가 구현한 프로그램의 잘못, failure_내 프로그램이 아닌 나의 프로그램을 도와주는 시스템(OS 등등)에서 생긴 잘못, fault_가장 심각, 하드웨어에서 고장)

▼ 자바의 예외처리

1. `ArithmeticException` : 0으로 나누는 경우에 주로 발생하는 예외 상황
2. `ArrayIndexOutOfBoundsException` : 배열의 크기보다 큰 원소를 접근하려고 할 때 발생하는 예외 상황
3. `NegativeArraySizeException` : 배열의 크기가 음수로 된 경우에 발생하는 예외 상황
4. `NullPointerException` : 힙에 객체가 생성되지 않았을 때 발생 → `new()`; 해주면 됨!

▼ 예외처리 과정

어떤 함수가 조건을 수행하는 함수라 하자.

이 함수를 수행하다 조건이 안 맞으면 비정상적으로 프로그램이 죽는데,

이를 try-catch를 사용하여 조건이 안 맞으면 예외처리를 발생시켜 catch문을 수행하게 하게끔 함

▼ 예시

“레스토랑 예약”과 관련된 상황이 생각하기 좋음!

<자리가 없으면?>, <해당 음식의 재료가 다 떨어졌다면?>

▼ throws / throw / try-catch-finally

▼ throws

함수가 실행될 때 발생할 수 있는 모든 예외 상황에 대한 조건 선언

▼ throw

예외를 발생시켜줌 `new Exception`

▼ try-catch-finally

예외처리를 처리해주는 과정

`try` : 성공적으로 모든 문장이 수행되거나 아니면 도중에 예외가 발생해서 `catch`문으로 제어가 넘어감

`catch` : 여러 개 존재할 수 있으며 만약 예외가 발생한다면 `catch`문에 선언된 예외 타입과 실제로 발생한 예외 타입이 일치하거나, 혹은 발생한 예외 타입의 슈퍼 타입이 선언된 `catch` 문 수행

`finally`는 예외 발생 여부와 관계없이 항상 마지막으로 한 번 더 수행되는 부분 (있어도 되고 없어도 됨)

▼ 예외처리 순서

1. 예외 클래스 정의(예외 종류에 따라 클래스가 많아질 수도 있음)
2. 예외가 발생할 수 있는 메소드를 throws를 이용해 구현
3. 조건에 따라 throw 발생
4. try-catch로 예외 처리 해결

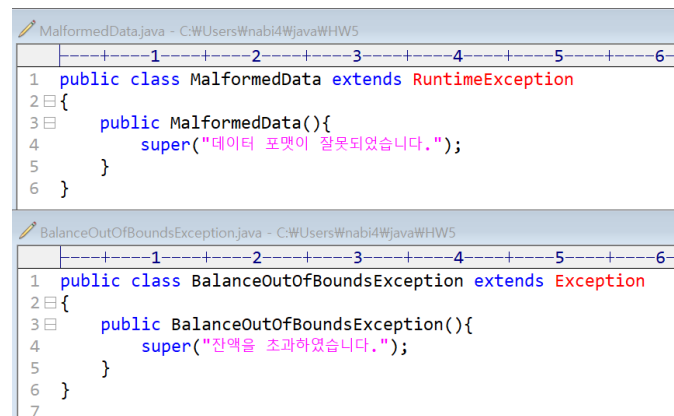
⇒ 적어도 총 3개의 클래스가 만들어진다고 할 수 있음(1, 2-3, 4마다 하나씩)

▼ Lab

예외발생 클래스 2개

함수 deposit, withdraw에서 예외 발생(throws) → 조건에 따라 처리(throw)

main : try-catch를 이용해 예외처리 해결



```

MalformedData.java - C:\Users\Wnabi4\java\HW5
1 public class MalformedData extends RuntimeException
2 {
3     public MalformedData(){
4         super("데이터 포맷이 잘못되었습니다.");
5     }
6 }

BalanceOutOfBoundsException.java - C:\Users\Wnabi4\java\HW5
1 public class BalanceOutOfBoundsException extends Exception
2 {
3     public BalanceOutOfBoundsException(){
4         super("잔액을 초과하였습니다.");
5     }
6 }
7
  
```

1. 예외 클래스 정의(MalformedData: 입력받은 금액이 음수일때 발생, BalanceOutOfBoundsException : 출금액이 잔액보다 많으면 발생 총 2개의 예외에 대해 정의함)


```

Account.java - C:\Users\Wnabi4W\java\HW5
1 public class Account
2 {
3     private long balance;
4     private String name;
5
6     public Account(String name){
7         this.name = name;
8     }
9
10    public void deposit(int amount) throws MalformedURLException{
11        if(amount <= 0){
12            throw new MalformedURLException();
13        }
14        balance = balance + amount;
15    }
16
17    public void withdraw(int amount)
18        throws BalanceOutOfBoundsException, MalformedURLException{
19        if(amount <= 0){
20            throw new MalformedURLException();
21        }
22        if(balance < amount){
23            throw new BalanceOutOfBoundsException();
24        }
25        balance = balance - amount;
26    }
27
28    public void check(){
29        System.out.println(name + ":" + balance);
30    }
31 }
32
33 }
34

```

2-3. 예외가 발생할 수 있는 상황에 대해 throws 작성후, 각각의 조건에 맞게 throw를 작성

```

Bank.java - C:\Users\Wnabi4W\java\HW5
1 public class Bank
2 {
3     public static void main(String[] args)
4     {
5         Account hong = new Account("홍길동");
6
7         hong.deposit(20);
8         //hong.deposit(-100); //MalformedURLException 예외 발생
9
10        try{
11            //1st try
12            hong.withdraw(15);
13            //hong.deposit(-20);
14
15            //2nd try
16            hong.withdraw(-50); //MalformedURLException 예외 발생
17
18        }catch (BalanceOutOfBoundsException be){
19            be.printStackTrace();
20        }catch (MalformedURLException me){
21            me.printStackTrace();
22        }
23
24        //1st try
25        hong.check();
26        //hong.withdraw(1); //BalanceOutOfBoundsException 예외 발생
27
28        //2nd try
29        try
30        {
31            hong.withdraw(10); //BalanceOutOfBoundsException 예외 발생
32        }
33        catch (BalanceOutOfBoundsException be)
34        {
35            be.printStackTrace();
36        }
37
38        System.out.println("normal termination");
39    }
40 }
41 }

```

MalformedURLException: 데이터 포맷이 잘못되었습니다.
 at Account.withdraw(Account.java:20)
 at Bank.main(Bank.java:16)
 홍길동:15
 BalanceOutOfBoundsException: 잔액을 초과하였습니다.
 at Account.withdraw(Account.java:23)
 at Bank.main(Bank.java:32)
 normal termination
 계속하려면 아무 키나 누르십시오 . . .

4. try-catch로 예외처리 해결

초록색 : 입력받은 금액이 음수 → MalformedURLException

파란색 : 출금액이 잔액보다 많음 → BalanceOutOfBoundsException

```

1 public class Bank
2 {
3     public static void main(String[] args)
4     {
5         Account hong = new Account("홍길동");
6
7         //hong.deposit(20);
8         hong.deposit(-100); //MalformedURLException 예외 발생
9
10        try{
11            //1st try
12            hong.withdraw(15);
13            //hong.deposit(-20);
14
15            //2nd try
16            //hong.withdraw(-50); //MalformedURLException 예외 발생
17
18        }catch (BalanceOutOfBoundsException be){
19            be.printStackTrace();
20        }catch (MalformedURLException me){
21            me.printStackTrace();
22        }
23
24        //1st try
25        hong.check();
26        hong.withdraw(1); //BalanceOutOfBoundsException 예외 발생
27
28        //2nd try
29        try
30        {
31            hong.withdraw(10); //BalanceOutOfBoundsException 예외 발생
32        }
33        catch (BalanceOutOfBoundsException be)
34        {
35            be.printStackTrace();
36        }
37
38        System.out.println("normal termination");
39    }
40 }

```

Bank.java:27: error: unreported exception BalanceOutOfBoundsException: must be caught or declared to be thrown
hong.withdraw(1); //BalanceOutOfBoundsException 예외 발생
error
계속하려면 아무 키나 누르십시오 . . .

4. try-catch로 예외처리 해결(다른 상황)

노란색 : 출금액이 잔액보다 많음 → BalanceOutOfBoundsException

```

1 public class Bank
2 {
3     public static void main(String[] args)
4     {
5         Account hong = new Account("홍길동");
6
7         //hong.deposit(20);
8         hong.deposit(-100); //MalformedURLException 예외 발생
9
10        try{
11            //1st try
12            hong.withdraw(15);
13            //hong.deposit(-20);
14
15            //2nd try
16            //hong.withdraw(-50); //MalformedURLException 예외
17
18        }catch (BalanceOutOfBoundsException be){
19            be.printStackTrace();
20        }catch (MalformedURLException me){
21            me.printStackTrace();
22        }
23
24        //1st try
25        hong.check();
26        //hong.withdraw(1); //BalanceOutOfBoundsException 예외 발생
27
28        //2nd try
29        try
30        {
31            hong.withdraw(10); //BalanceOutOfBoundsException 예외 발생
32        }
33        catch (BalanceOutOfBoundsException be)
34        {
35            be.printStackTrace();
36        }
37
38        System.out.println("normal termination");
39    }
40 }

```

Exception in thread "main" MalformedURLException: 데이터 포맷이 잘못되었습니다.
at Account.deposit(Account.java:12)
at Bank.main(Bank.java:8)
계속하려면 아무 키나 누르십시오 . . .

4. try-catch로 예외처리 해결(다른 상황)

노란색 : 입력받은 금액이 음수 → MalformedURLException

▼ HW 3-2(자신만의 예외처리_더치페이 프로그램)

```

WorngformedMoney.java - C:\Users\Wnabi4Wjava\HW5
1 public class WorngformedMoney extends RuntimeException
2 {
3     public WorngformedMoney(){
4         super("음수의 금액은 계산할 수 없습니다. 0 이상의 수를 입력해주세요.");
5     }
6 }
7

WorngformedPeoplenum.java - C:\Users\Wnabi4Wjava\HW5
1 public class WorngformedPeoplenum extends RuntimeException
2 {
3     public WorngformedPeoplenum(){
4         super("0 이하의 사람 수는 더치페이를 할 수 없습니다. 1 이상의 사람 수를 입력해주세요.");
5     }
6 }
7

```

1. 예외 클래스 정의

WrongformedMoney : 더치페이할 금액을 음수로 입력한 경우

WrongformedPeoplenum : 더치페이할 인원 수를 음수 또는 0으로 입력한 경우

```

Dutch.java - C:\Users\Wnabi4Wjava\HW5
1 import java.util.Scanner;
2
3 public class Dutch
4 {
5     private int money;
6     private int peoplenum;
7     private int moneyperperson;
8     private String object;
9
10    Scanner in = new Scanner(System.in);
11
12    public Dutch(String object){
13        this.object = object;
14    }
15
16    public void inputmoney() throws WorngformedMoney
17    {
18        System.out.println("더치페이하실 금액을 입력하세요.");
19        money = in.nextInt();
20        if(money < 0){
21            throw new WorngformedMoney();
22        }
23        this.money = money;
24        System.out.println(money + "원을 입력하셨습니다.");
25    }
26
27    public void inputpeoplenum() throws WorngformedPeoplenum
28    {
29        System.out.println("정입할 사람 수를 입력하세요.");
30        peoplenum = in.nextInt();
31        if(peoplenum <= 0){
32            throw new WorngformedPeoplenum();
33        }
34        this.peoplenum = peoplenum;
35        System.out.println(peoplenum + "명을 입력하셨습니다.");
36    }
37
38    public void check()
39    {
40        System.out.println(money + "원을 "
41            + peoplenum + "명으로 더치페이하겠습니다.");
42    }
43
44    public void divide()
45    {
46        moneyperperson = money/peoplenum;
47    }
48
49    public void show()
50    {
51        System.out.println("1인당 " + moneyperperson +
52            "원의 정산액입니다.");
53    }
54 }

DutchMain.java - C:\Users\Wnabi4Wjava\HW5
1 public class DutchMain
2 {
3     public static void main(String[] args)
4     {
5         int condition = 0;
6         Dutch example = new Dutch("tteokbokki");
7
8         while(true){
9             try{
10                 condition = 1;
11                 example.inputmoney();
12             }catch(WorngformedMoney wfm){
13                 wfm.printStackTrace();
14                 condition = 2;
15             }finally{
16                 if (condition == 2)
17                 {
18                     System.out.println("금액을 다시 입력해주세요.");
19                 }else{
20                     System.out.println("금액이 정상적으로 입력되었습니다.");
21                     break;
22                 }
23             }
24         }
25
26         while(true){
27             try{
28                 condition = 1;
29                 example.inputpeoplenum();
30             }catch(WorngformedPeoplenum wfp){
31                 wfp.printStackTrace();
32                 condition = 2;
33             }finally{
34                 if (condition == 2)
35                 {
36                     System.out.println("더치페이할 인원을 다시 입력해주세요.");
37                 }else{
38                     System.out.println("인원 수가 정상적으로 입력되었습니다.");
39                     break;
40                 }
41             }
42         }
43
44         example.check();
45         example.divide();
46         example.show();
47     }
48 }

```

2-3. Dutch.java : 앞에서 정의한 예외 클래스들을 throws와 throw를 통해 구현

check() 함수는 입력한 금액과 인원 수를 다시 확인하기 위한 함수

divide() 함수는 더치페이 금액을 계산할 함수

show() 함수는 이용자에게 1인당 금액을 보여주는 함수

4. DutchMain.java : 실제 실행시킬 프로그램. 예외처리가 발생하여도 다시 입력할 수 있도록 while(true)를 이용해주었고 상황에 따라 예외처리를 발생하도록 함

▼ 결과

```

더치페이하실 금액을 입력하세요.
15000
15000원을 입력하셨습니다.
금액이 정상적으로 입력되었습니다.
정산할 사람 수를 입력하세요.
5
5명을 입력하셨습니다.
인원 수가 정상적으로 입력되었습니다.
15000원을 5명으로 더치페이하겠습니다.
1인당 3000원씩 정산하면 됩니다!
계속하려면 아무 키나 누르십시오 . . .

```

정상 작동

```

더치페이하실 금액을 입력하세요.
-5000
WrongformedMoney: 음수의 금액은 계산할 수 없습니다. 0 이상의 수를 입력해주세요:)
    at Dutch.inputmoney(Dutch.java:21)
    at DutchMain.main(DutchMain.java:11)
금액을 다시 입력해주세요.
더치페이하실 금액을 입력하세요.
5000
5000원을 입력하셨습니다.
금액이 정상적으로 입력되었습니다.
정산할 사람 수를 입력하세요.
2
2명을 입력하셨습니다.
인원 수가 정상적으로 입력되었습니다.
5000원을 2명으로 더치페이하겠습니다.
1인당 2500원씩 정산하면 됩니다!
계속하려면 아무 키나 누르십시오 . . .

```

정산할 금액이 음수로 입력된 경우 예외처리를 발생시킨 후 다시 입력하게 함

```

더치페이하실 금액을 입력하세요.
6000
6000원을 입력하셨습니다.
금액이 정상적으로 입력되었습니다.
정산할 사람 수를 입력하세요.
-6
WrongformedPeoplenum: 0 이하의 사람 수는 더치페이를 할 수 없습니다. 1 이상의 사람 수를 입력해주세요:)
    at Dutch.inputpeoplenum(Dutch.java:33)
    at DutchMain.main(DutchMain.java:30)
더치페이할 인원을 다시 입력해주세요.
0
WrongformedPeoplenum: 0 이하의 사람 수는 더치페이를 할 수 없습니다. 1 이상의 사람 수를 입력해주세요:)
    at Dutch.inputpeoplenum(Dutch.java:33)
    at DutchMain.main(DutchMain.java:30)
더치페이할 인원을 다시 입력해주세요.
6
6명을 입력하셨습니다.
인원 수가 정상적으로 입력되었습니다.
6000원을 6명으로 더치페이하겠습니다.
1인당 1000원씩 정산하면 됩니다!
계속하려면 아무 키나 누르십시오 . . .

```

정산할 금액은 옳게 입력 / 정산할 사람 수를 음수 또는 0으로 입력한 경우 예외처리를 발생시킨 후 다시 입력하게 함

▼ 3.10 Generics in Java

▼ Generics

자주 사용하는 타입만 정의

String인지 Int인지에 대한 데이터 형 선언은 나중에 개발자가 사용할때 선언

메서드들이 저장되어있음

선언하는 방법 : (String)이라 선언을 해줘야하는데 이게 너무 불편하다보니

<String> 으로 앞에서 선언

Elimination of casts.

- ✓ The following code snippet without generics requires casting:
`List list = new ArrayList();`
`list.add("hello");`
`String s = (String) list.get(0);`
- ✓ When re-written to use generics, the code does not require casting:
`List<String> list = new ArrayList<String>();`
`list.add("hello");`
`String s = list.get(0); // no cast`

아래에서 더 자세히 설명

▼ Generic을 사용하는 이유

강제적으로 타입 변환이 발생하지 않음

반복적인 코드가 줄어들어 재사용성이 높아짐

컴파일 시 타입 오류를 체크하여 사전에 엄격한 타입체크가 가능해짐

▼ 사용법

클래스나 인터페이스 뒤에 <> 키워드를 작성하여 사용

타입인자-설명 : <T>-Type, <E>-Element, <K>-Key, <N>-Number, <V>-Value, <R>-Result

▼ 3.11 Collection types in java

▼ collection type

여러 요소를 하나의 유닛으로 묶은 것

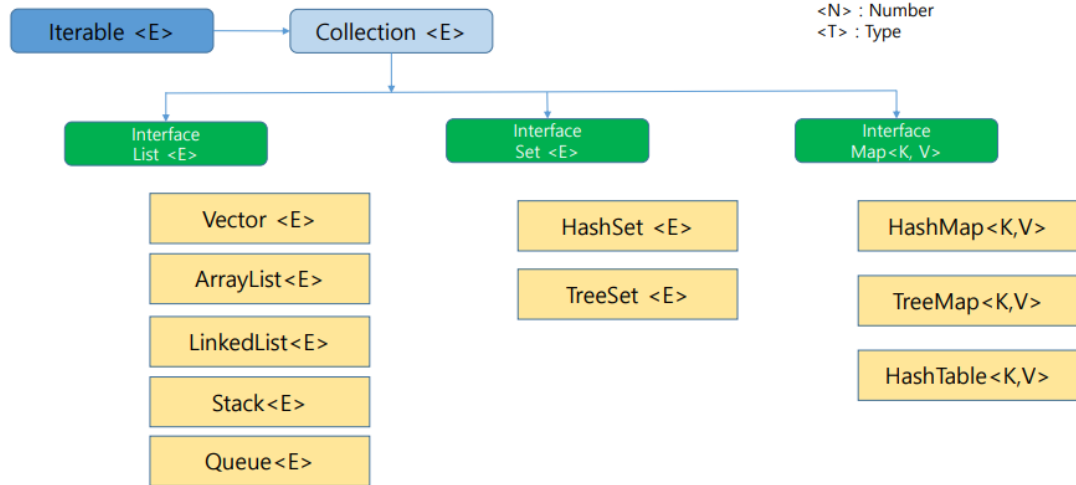
Stack, Queue, List, Array, Map, Table이 다 collection type

이 데이터들을 저장하는 방법에 따라 쓸 수 있는 알고리즘이 다름

저장하고 꺼낼때 필요한 기능들이 다 저장되어있어 저장, 검색, 조작에 용이

▼ 기본 구조

- Each collection class implements an interface from a hierarchy
 - Each class is designed for a specific type of storage



반복해서 사용할 함수(iterator interface - 메서드 포함되어있음) - 반복해서 사용할 부분은 모두 다 collection에 들어가있음(collection interface - 메서드 포함되어있음)

개발자가 알아서 상황에 맞게 사용하면 됨

▼ 참고

> 자료구조 별 일반 특징 비교 (대분류)

Class or Interface	Superinterfaces	Notes	중복허용	Null허용	멤버접근
Collection	Iterable	전체 Collection의 상위 인터페이스.	-	-	iterator
List	Collection	원하는 순서로 Element 삽입가능. 각 요소는 Index 번호를 부여 받는다.	Yes	-	iterator 또는 숫자 Index로 임의 접근
Set	Collection	중복 Element 불가능. 그러므로 쉽게 여부 중복확인 가능. 특정 순서(Order) 정할수 없음.	No	-	iterator
Queue	Collection	Output으로 나올 Element만 기본적으로 접근 가능하다.	Yes	-	Peeking or Polling or iterator
PriorityQueue	Queue	가장 우선순위가 높은 Element가 Head First가 된다.	Yes	No	Peeking or Polling
Deque	Queue	양 끝단에서 모두 삽입 / 삭제 가능	Yes	-	Peeking or Polling or iterator
Map	해당 없음	Key / Value로 구성된다.	키 : No - 값 : Yes	키 : 하나의 (null) 키 허용) - 값 : Yes	통상 Key를 통해 접근 혹은 (keySet ; entrySet ; values) View를 제공한다.

> 주요 자료구조 별 특징 비교 (실구현 Class)

Class	Base class	Base interfaces	중복허용	순서(Order) 존재	정렬여부	Thread-safe
ArrayList	AbstractList	List	Yes	Yes	No	No
LinkedList	AbstractSequentialList	List;Deque	Yes	Yes	No	No
Vector	AbstractList	List	Yes	Yes	No	Yes
HashSet	AbstractSet	Set	No	No	No	No
LinkedHashSet	HashSet	Set	No	Yes	No	No
TreeSet	AbstractSet	Set;NavigableSet;SortedSet	No	Yes	Yes	No
HashMap	AbstractMap	Map	No	No	No	No
LinkedHashMap	HashMap	Map	No	Yes	No	No
Hashtable	Dictionary	Map	No	No	No	Yes
TreeMap	AbstractMap	Map;NavigableMap;SortedMap	No	Yes	Yes	No

▼ Iterator Interface

1. object next(); : 이터레이션의 다음 요소 반환
2. boolean hasNext(); 해당 이터레이션이 다음 요소를 가지고 있으면 true, 없으면 false 반환
3. void remove(); 해당 반복자로 반환되는 마지막 요소를 현재 컬렉션에서 제거

▼ ListIterator Interface

1. void add(E e); 해당 리스트에 전달된 요소 추가
2. boolean hasNext(); 이 리스트 반복자가 해당 리스트를 순방향으로 순회할 때 다음 요소를 가지고 있으면 true, 없으면 false 반환
3. boolean hasPrevious(); 역방향으로 순회할 때 다음 요소를 가지고 있으면 true, 없으면 false 반환
4. E next(); 리스트의 다음 요소를 반환, 커서의 위치를 순방향으로 이동
5. void remove(); next()나 previous() 메서드에 의해 반환된 가장 마지막 요소를 리스트에서 제거

▼ Collection Interface

: 컬렉션 프레임 워크의 클래스 상속관계에서 가장 상위에 존재하는 인터페이스. 객체들의 모임을 표현하기 위해서 사용됨.

1. `int size()`; 컬렉션에 포함된 원소의 개수 리턴
2. `boolean isEmpty()`; 컬렉션이 비어있는지 여부 리턴
3. `boolean add(E o)`; 새로운 원소 객체를 추가, 새로운 객체가 올바르게 추가되면 `true`, 아니면 `false`를 리턴
4. `boolean remove(Object o)`; 매개변수로 전달된 객체를 컬렉션에서 제거. 컬렉션에서 객체가 제거되는 경우에 `true` 리턴
5. `boolean contains(Object o)`; 주어진 객체를 포함하고 있는지 여부 리턴
6. `Iterator iterator()`; 컬렉션에 포함된 원소 객체들을 iterator 형태로 리턴

▼ Lab 3.11

```
SimpleCollection.java - C:\Users\Wnabi4W\java\HW5
1  import java.util.ArrayList;
2  import java.util.Iterator;
3
4
5  public class SimpleCollection
6  {
7      public static void main(String[] args)
8      {
9          Collection c;
10         c = new ArrayList();
11
12         System.out.println(c.getClass().getName());
13
14         for (int i = 1; i <= 10; i++)
15         {
16             c.add(i + " * " + i + " = " + i*i);
17         }
18
19         Iterator iter = c.iterator();
20         while (iter.hasNext())
21         {
22             System.out.println(iter.next());
23         }
24     }
25 }
```

```
C:\Windows\System32\cmd.exe
SimpleCollection.java:9: error: cannot find symbol
    Collection c;
    ^
symbol:   class Collection
location: class SimpleCollection
1 error
계속하려면 아무 키나 누르십시오 . . .
```

ArrayList와 Iterator를 import해주고 컴파일했더니 계속 Collection에 대한 에러가 났다. 구글링 해봐도 Collection에 대한 해결방법을 찾지 못했지만


```
SimpleCollection.java - C:\Users\Wnabi4\java\HW5
1 import java.util.*;
2
3
4 public class SimpleCollection
5 {
6     public static void main(String[] args)
7     {
8         Collection c;
9         c = new ArrayList();
10
11         System.out.println(c.getClass().getName());
12
13         for (int i = 1; i <= 10 ; i++ )
14         {
15             c.add(i + " * " + i + " = " + i*i);
16         }
17
18         Iterator iter = c.iterator();
19         while (iter.hasNext())
20         {
21             System.out.println(iter.next());
22         }
23     }
24 }
```

```
C:\Windows\System32\cmd.exe
Note: SimpleCollection.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
계속하려면 아무 키나 누르십시오 . . .

C:\Windows\System32\cmd.exe
java.util.ArrayList
1 * 1 = 1
2 * 2 = 4
3 * 3 = 9
4 * 4 = 16
5 * 5 = 25
6 * 6 = 36
7 * 7 = 49
8 * 8 = 64
9 * 9 = 81
10 * 10 = 100
계속하려면 아무 키나 누르십시오 . . .
```

import java.util.*로 임포트해주시 주의사항은 뜨지만 잘 실행되는 것을 확인할 수 있었다.
주의사항 중 첫번째 줄(unchecked~)에 대한 주의사항은 collection 사용시 타입을 지정해주지 않아 발생하는 것이고
두번째 줄(recompile~)에 대한 주의사항은 자바 버전이 업그레이드되면서 없어진 기능을 사용했을 때 발생하는 메시지라고 함

[위 예제에 대한 설명]

Collection type의 배열 리스트 객체 c 생성후 for문을 이용해 구구단을 순서대로 추가함. 그 후 c를 iterator 객체에 대입하여 iterator의 next() 메서드를 사용하여 내부의 내용을 출력

▼ List implementations

▼ ArrayList

크기가 변경될 수 있는 배열을 이용해 List 인터페이스를 구현

랜덤한 접근 가능

삽입과 삭제시 큰 비용이 발생

▼ 메서드

1. Object get(int index) : 주어진 위치에 있는 객체 리턴
2. Object set(int index, Object element) : 주어진 위치에 있는 객체를 매개 변수로 주어진 element 객체로 교체
3. void add(int index, Object element) : 주어진 위치에 객체 삽입
4. Object remove(int index) : 주어진 위치에 맞는 객체를 삭제하고 리턴
5. void ensureCapacity(int minCapacity) : 용량이 최소한 minCapacity를 만족하도록 필요한 경우 용량을 증가시킴

▼ LinkedList

순차적인 접근 가능

삽입과 삭제시 적은 비용 발생

▼ 메서드

1. ListIterator listIterator : 모두 listIterator 타입으로 리턴
2. add() / remove() : 객체 삽입 / 삭제
3. void addFirst(Object o) / void addLast(Object o) : 처음 / 마지막 부분에 객체 삽입
4. Object getFirst() / Object getLast() : 첫 번째 / 마지막 객체 리턴
5. Object removeFirst() / Object removeLast() : 리스트의 첫 번째 / 마지막 객체를 제거하고 리턴

: 둘의 차이점

▼ HashSet

Set 인터페이스를 HashMap을 이용하여 구현한 클래스

null값을 가질 수 있고 원소들의 순서는 존재하지 않음

객체 검색과 삽입을 빠르게 할 수 있다는 장점(hashmap이 더 좋음)

▼ Hashing

linked list를 이용

hashCode() : 배열의 인덱스 접근 가능

equals() : 해당 인덱스의 목록에 있는지 확인

▼ TreeSet

Set 인터페이스를 TreeMap을 이용하여 구현한 클래스

▼ Map Interface

key/value 를 쌍으로 저장

key 는 unique , value는 unique하지 않아도 됨

▼ HashMap and TreeMap

▼ HashMap

key들의 집합(unique, unordered)

▼ TreeMap

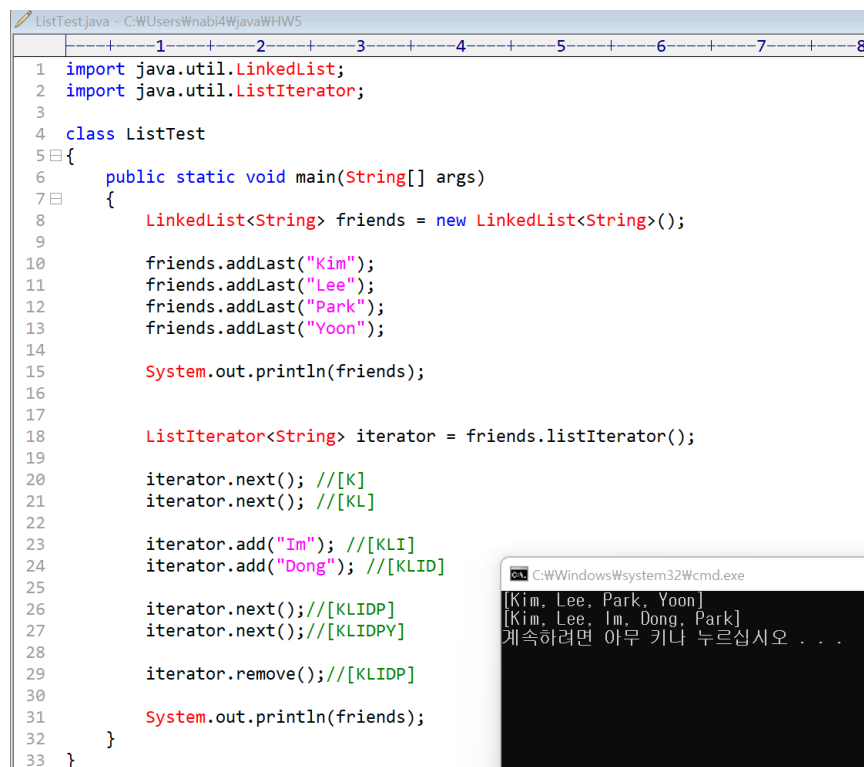
▼ Bulk

▼ 메서드

1. boolean containsAll(Collection c); 주어진 컬렉션에 있는 모든 원소 객체들을 포함하고 있는지 여부 리턴
2. boolean addAll(Collection c); 전달된 컬렉션에 포함된 모든 원소 객체를 추가
3. boolean removeAll(Collection c); 전달된 컬렉션에 포함된 모든 원소 객체를 제거
4. boolean retainAll(Collection c); 전달된 컬렉션에 포함된 원소들을 제외한 모든 원소들을 제거

⇒ ~All 메서드들 : 잘 실행되면 true

▼ Lab3.11(ListTest.java)



```
ListTest.java - C:\Users\wnabi4\java\HW5
1  import java.util.LinkedList;
2  import java.util.ListIterator;
3
4  class ListTest
5  {
6      public static void main(String[] args)
7      {
8          LinkedList<String> friends = new LinkedList<String>();
9
10         friends.addLast("Kim");
11         friends.addLast("Lee");
12         friends.addLast("Park");
13         friends.addLast("Yoon");
14
15         System.out.println(friends);
16
17
18         ListIterator<String> iterator = friends.listIterator();
19
20         iterator.next(); //[K]
21         iterator.next(); //[KL]
22
23         iterator.add("Im"); //[KLI]
24         iterator.add("Dong"); //[KLID]
25
26         iterator.next(); //[KLIDP]
27         iterator.next(); //[KLIDPY]
28
29         iterator.remove(); //[KLIDP]
30
31         System.out.println(friends);
32     }
33 }
```

C:\Windows\system32\cmd.exe
[Kim, Lee, Park, Yoon]
[Kim, Lee, Im, Dong, Park]
계속하려면 아무 키나 누르십시오 . . .

LinkedList와 Iterator를 이용하여 리스트 속 요소를 추가, 제거하고 배열하는 것을 볼 수 있음

▼ Lab3.11(WordCheck.java)

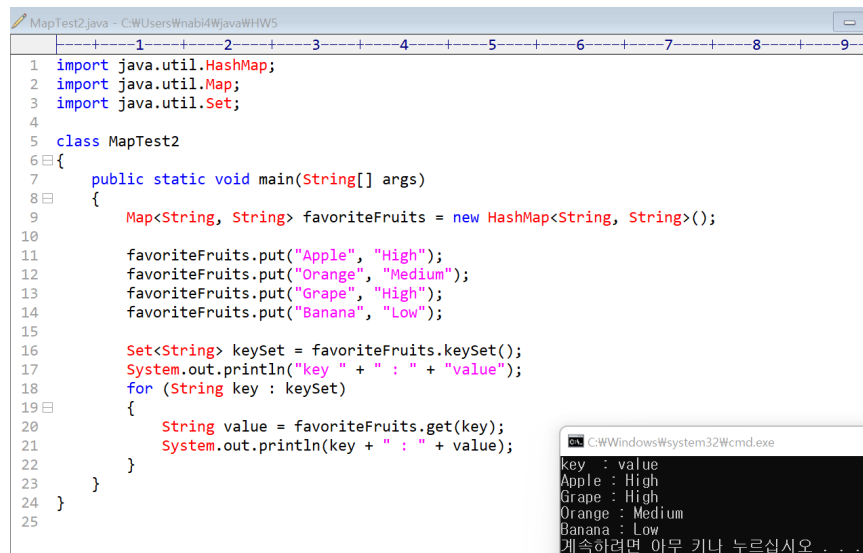
파일을 읽기 위한 Scanner

```
WordCheck.java - C:\Users\Wnabi4\java\HW5
1 import java.util.HashSet;
2 import java.util.Scanner;
3 import java.util.Set;
4 import java.io.File;
5 import java.io.FileNotFoundException;
6
7
8 public class WordCheck
9 {
10     public static void main(String[] args) throws FileNotFoundException
11     {
12         Set<String> dictionaryWords = readWords("words.txt");
13         Set<String> documentWords = readWords("test.txt");
14
15         for (String word : documentWords)
16         {
17             if(!dictionaryWords.contains(word))
18                 System.out.println(word);
19         }
20     }
21
22     public static Set<String> readWords(String filename) throws FileNotFoundException
23     {
24         Set<String> words = new HashSet<String>();
25         Scanner in = new Scanner(new File(filename));
26
27         in.useDelimiter("[^a-zA-Z]+");
28         while(in.hasNext())
29             words.add(in.next().toLowerCase());
30         return words;
31     }
32 }
33 }
```

```
test.txt - C:\Users\Wnabi4\java\HW5
1 SungMi Hwang is double majoring in Statistics and IT at Sookmyung Women's University.
2 She is taking a Java class.
3
words.txt - C:\Users\Wnabi4\java\HW5
1 SookMyung
2 Women
3 University
4 student
5 IT
6 JAVA
7 Programming
8 sungmi
C:\Windows\System32\cmd.exe
a
in
double
majoring
is
she
at
s
and
hwang
class
statistics
taking
계속하려면 아무 키나 누르십시오 . . .
```

HashSet을 이용하여 파일 속 단어를 읽어오는 예제
두 개의 파일을 읽은 뒤, 두 번째 파일(test.txt)의 단어 중 첫 번째 파일(words.txt)의 단어와 중복
되지 않은 단어만 출력

▼ Lab 3.11(MapTest2.java)



```
MapTest2.java - C:\Users\Wnabi4\java\HHW5
1 import java.util.HashMap;
2 import java.util.Map;
3 import java.util.Set;
4
5 class MapTest2
6 {
7     public static void main(String[] args)
8     {
9         Map<String, String> favoriteFruits = new HashMap<String, String>();
10
11         favoriteFruits.put("Apple", "High");
12         favoriteFruits.put("Orange", "Medium");
13         favoriteFruits.put("Grape", "High");
14         favoriteFruits.put("Banana", "Low");
15
16         Set<String> keySet = favoriteFruits.keySet();
17         System.out.println("key " + " : " + "value");
18         for (String key : keySet)
19         {
20             String value = favoriteFruits.get(key);
21             System.out.println(key + " : " + value);
22         }
23     }
24 }
25
```

```
C:\Windows\system32\cmd.exe
key : value
Apple : High
Grape : High
Orange : Medium
Banana : Low
계속하려면 아무 키나 누르십시오 . . .
```

Map형태도 key-value형태로 출력할 수 있음