|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **[유헬스2\_14주\_실습과제]** | **학번:** |  | **이름:** |  |

* 예외처리 : try, catch, finally

1. 예외가 발생한 곳에서 처리

import java.util.\*;

public class DivideByZero {

public static void main(String[] args){

int x,y;

Scanner sc=new Scanner(System.*in*);

System.*out*.print("피젯수: "); x=sc.nextInt();

System.*out*.print("젯수: "); y=sc.nextInt();

try{ //예외 발생 가능한 문장을 try 블록에 작성

int result=x/y; //예외 발생

System.*out*.println(" x/y = " + result);

}

catch(ArithmeticException e){ //발생한 예외를 처리

System.*out*.println("0으로 나눌 수 없습니다");

}

System.*out*.println("프로그램을 종료합니다.");

}

}

// 프로그램 실행 시 y값을 0과 0이 아닌 값으로 각각 입력하였을 때 결과를 제시하시오

|  |
| --- |
|  |

2. 예외를 상위 메소드로 전달

public class Account {

private int balance; //잔액

public void setBalance(int balance) {

this.balance = balance;

}

//발생한 예외를 호출한 곳으로 전달, 현재 메소드에서 발생하는 예외종류를 throws 절로 표시

int withdraw(int amount) throws Exception {

if (balance < amount)

//throw절을 이용하여 인위적으로 예외객체 생성

throw new Exception("잔액이 부족합니다.");

balance -= amount;

return amount;

}

}

public class ExceptionExample {

public static void main(String[] args) {

Account obj = new Account();

obj.setBalance(100000);

try { //예외 발생 가능성이 있는 문장을 try블럭에 작성

int amount = obj.withdraw(500000000); //예외 발생 메소드

System.*out*.println("인출액:" + amount);

}

catch (Exception e) { //발생된 예외 처리

System.*out*.println(e.getMessage()); //오류 메시지 출력

}

finally{ //예외 발생과 상관없이 항상 실행

System.*out*.println("실행을 종료합니다.");

}

}

}

// withdraw()메소드 실매개 변수 값을 5000으로 수정하기 전과 후의 결과를 제시하시오

|  |
| --- |
| **[실행결과]** |

* 사용자 정의 예외 클래스
* Exception 을 상속받아 구현
* 생성자에서 super(msg)로 구현
* msg에 저장된 문자열은 메소드 getMessage()에 의해 반환
  + 생성된 예외의 발생
* 생성된 예외 클래스는 필요한 경우 new를 사용하여 예외 발생(throws)
* 메소드 선언에서 throws 사용
* 발생되는 예외가 여러 개라면 여러 예외 유형을 쉼표로 구분하여 기술

class ZeroException extends Exception{ //사용자 정의 예외 클래스

public ZeroException(){

super("0으로 나눌 수는 없슴");

}}

public class ExceptionTest {

public static void main(String[] args) {

double result;

try{

result= quotient(20, 0);}

catch(ZeroException e){ //사용자 정의 예외 처리

System.out.println(e. getMessage());

}

}

public static double quotient(int n, int d) throws ZeroException{

if(d==0)

throw new ZeroException(); //예외 발생

return (double)n/d;}

}

|  |
| --- |
| **[실행결과]** |

* 스레드
* 다중 스레딩(multi-threading)은 하나의 프로그램이 동시에 여러 가지 작업을 할 수 있도록 하는 것
* 각각의 작업은 스레드(thread)라고 불린다
* 스레드를 생성하는 방법
  + java.lang.Thread 클래스를 상속하는 방법
  + java.lang.Runnable 인터페이스를 이용하는 방법

//사용 예1 :Thread 클래스 상속

//숫자를 출력하는 스레드

class DigitThread extends Thread {

//스레드로 처리할 내용을 run()에서 작성

public void run() {

for (int cnt = 0; cnt < 10; cnt++)

System.*out.print(cnt); }}*

public class MultithreadExample1 {

public static void main(String[] args) {

Thread thread = new DigitThread(); // 스레드 객체 생성

thread.start(); // 스레드 시작

for (char ch = 'A'; ch <= 'Z'; ch++)

System.*out.print(ch);*

}

}

//사용 예2: Runnable 인터페이스 구현

class MyRunnable implements Runnable {

String myName;

public MyRunnable(String name) { //스레드를 구분하기 위해 이름 설정

myName = name;}

public void run() {

for (int i = 10; i >= 0; i--)

System.out.print(myName + i + " ");

}}

public class TestThread {

public static void main(String[] args) {

Thread t1 = new Thread( new MyRunnable("A") );

Thread t2 = new Thread( new MyRunnable("B") );

t1.start();

t2.start();}}

|  |
| --- |
| **[실행결과]** |

* 프로그램 과제

1. 음수가 전달되면 사용자 예외를 발생한다. 빈 공란을 채워 넣은 후 실행결과를 제시하시오

**//사용자 정의 예외 클래스 MyException**

class MyException {

public MyException(String str){

//형식매개변수 str을 수퍼클래스 생성자로 전달

}

}

public class MyExceptionTest {

**//사용자 정의 예외 MyException을 메소드를 호출한 곳으로 던진다**

public static void checkNegative(int no) {

if(no < 0)

**//사용자 정의 예외 MyException를 인위적으로 생성**

throw new MyException(no+ "는 사용 불가(음수데이터)");

else

System.*out*.println(no+ "는 사용 가능한 데이터");

}

public static void main(String[] args) {

{

*checkNegative*(10);

*checkNegative*(-1);

}catch(MyException e){

System.*out*.println(e.getMessage());

}

{

System.*out*.println("항상 실행되는 부분입니다.");

}

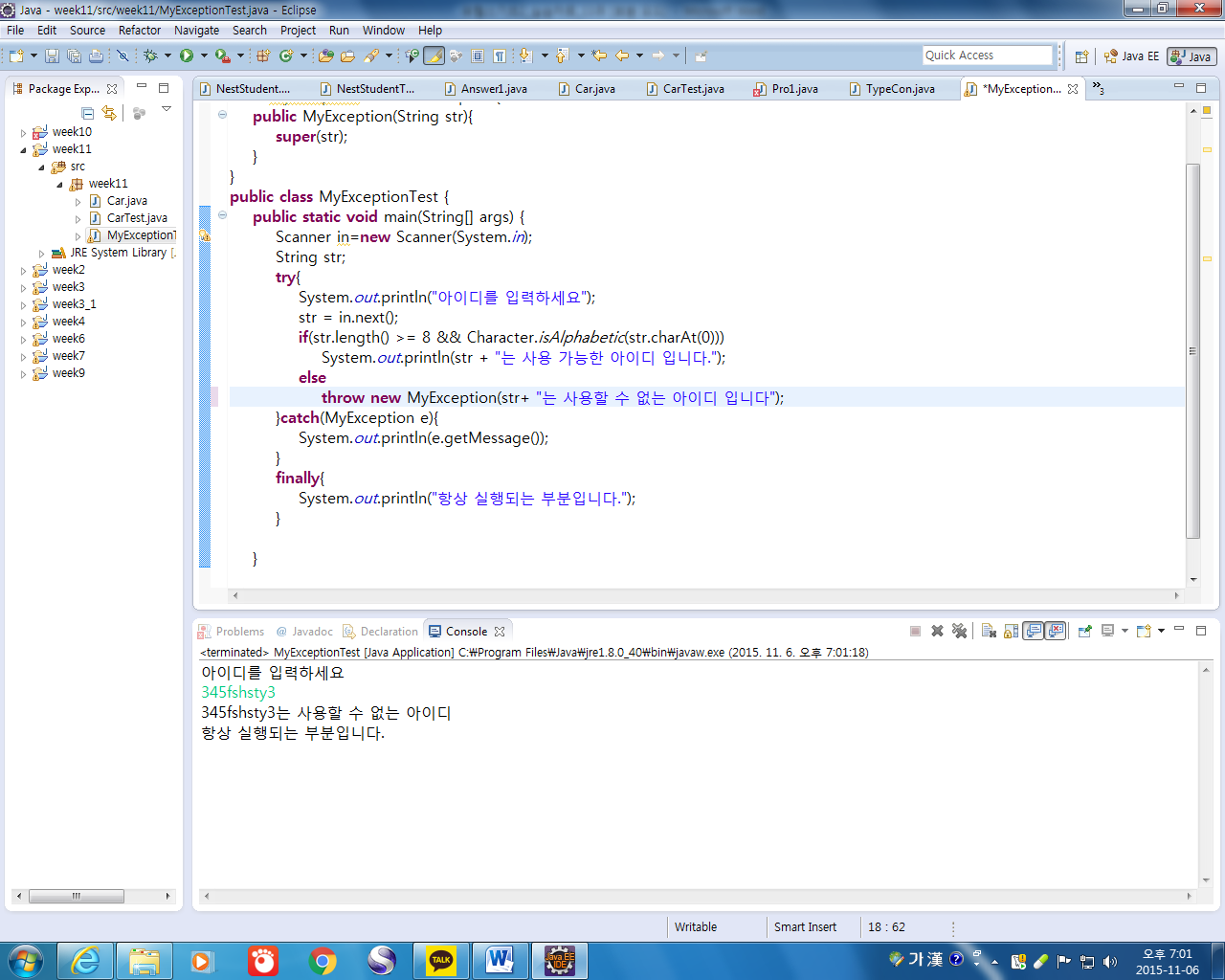
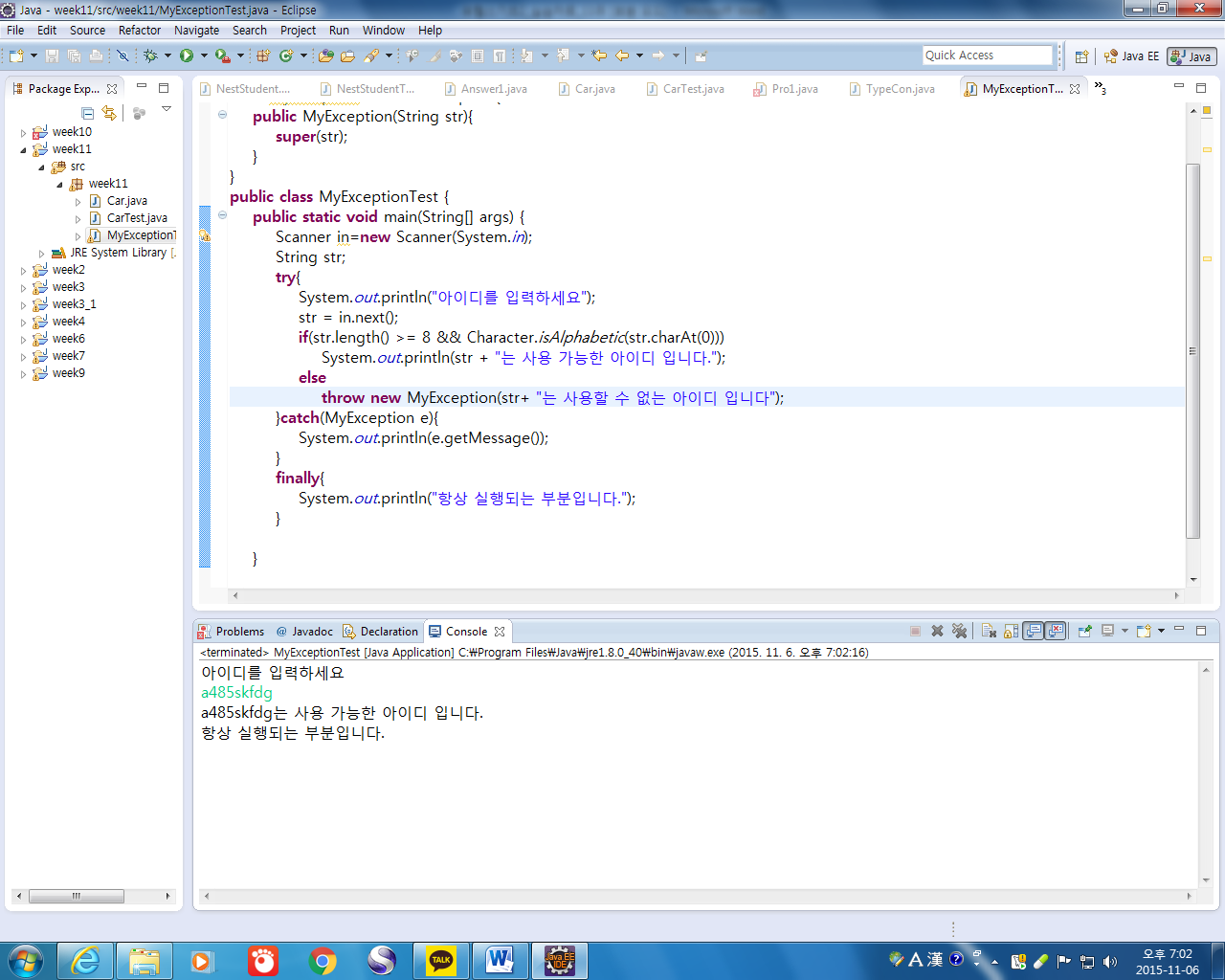
}

}

|  |
| --- |
| **[프로그램 소스]**  **class** MyException **extends** Exception {  **public** MyException(String str) {  **super**(str); // 형식매개변수 str을 수퍼클래스 생성자로 전달  }  }  **public** **class** MyExceptionTest {  // 사용자 정의 예외 MyException을 메소드를 호출한 곳으로 던진다  **public** **static** **void** checkNegative(**int** no) **throws** MyException {  **if** (no < 0)  // 사용자 정의 예외 MyException를 인위적으로 생성  **throw** **new** MyException(no + "는 사용 불가(음수데이터)");  **else**  System.***out***.println(no + "는 사용 가능한 데이터");  }  **public** **static** **void** main(String[] args) {  **try** {  *checkNegative*(10);  *checkNegative*(-1);  } **catch** (Exception e) {  System.***out***.println(e.getMessage());  } **finally** {  System.***out***.println("항상 실행되는 부분입니다.");  }  }  } |
| **[실행결과]** |

1. 키보드로 입력 받은 문자열에 대하여 길이가 8미만이거나 첫 글자가 영문자가 아니면 NotUseId 사용자 예외를 발생시켜 “사용할 수 없는 아이디”를 출력하고, 길이가 8이상이고 첫 글자가 영문자 이면 “사용 가능한 아이디 입니다.” 를 출력한다.

[출력예시]

|  |
| --- |
| **[프로그램 소스]**  **import** java.util.\*;  **class** MyException1 **extends** Exception {  **public** MyException1(String str) {  **super**(str); // 형식매개변수 str을 수퍼클래스 생성자로 전달  }  }  **public** **class** ID {  // 사용자 정의 예외 MyException을 메소드를 호출한 곳으로 던진다  **public** **static** **void** NotUseId(String no) **throws** MyException1 {  **if** (no.length() < 8  || !((**int**) no.charAt(0) >= 65 && (**int**) no.charAt(0) <= 122))  **throw** **new** MyException1(no + "는 사용할 수 없는 아이디");  **else**  System.***out***.println(no + "는 사용 가능한 아이디입니다.");  }  **public** **static** **void** main(String[] args) {  Scanner scan = **new** Scanner(System.***in***);  **try** {  System.***out***.println("아이디를 입력하세요.");  *NotUseId*(scan.next());  } **catch** (Exception e) {  System.***out***.println(e.getMessage());  }  }  } |
| **[실행결과]** |

1. 생성된 난수 만큼 문자와 숫자를 번갈아 출력하는 스레드 프로그램을 작성하시오. 숫자와 문자를 출력하는 스레드 클래스를 각각 작성. 난수 범위는 1에서 50

import java.util.\*;

public class ThreadCharNum {

public static void main(String[] args) {

//난수를 생성

//숫자와 문자를 출력하는 스레드 객체 각각 생성

//이때 난수를 생성자 형식 매개변수로 전달

//스레드 시작

}

}

public class CharThread { //생성된 난수 만큼 문자 출력하는 스레드

//출력할 문자와 횟수를 저장하기 위한 필드 선언

//매개변수로 문자와 횟수를 받아 필드 초기화하는 생성자

public void run() {

// 횟수만큼 문자를 출력

}

}

public class NumThread { //생성된 난수 만큼 숫자 출력하는 스레드

//출력할 숫자와 횟수를 저장하기 위한 필드 선언

//매개변수로 숫자와 횟수를 받아 필드 초기화하는 생성자

public void run() {

// 횟수만큼 숫자를 출력

}

}

|  |
| --- |
| **[프로그램 소스]**    **import** java.util.\*;  **public** **class** ThreadCharNum {  **public** **static** **void** main(String[] args) {  // 난수를 생성  **int** chran = (**int**) (Math.*random*() \* 26 + 65);// 문자 랜덤  **int** chcount = (**int**) (Math.*random*() \* 30 + 1);// 문자 출력횟수랜덤  **int** numran = (**int**) (Math.*random*() \* 9 + 1);// 숫자 랜덤  **int** numcount = (**int**) (Math.*random*() \* 30 + 1);// 숫자 출력횟수랜덤  CharThread cha = **new** CharThread((**char**) chran, chcount);  NumThread num = **new** NumThread(numran, numcount);  // 숫자와 문자를 출력하는 스레드 객체 각각 생성  // 이때 난수를 생성자 형식 매개변수로 전달  cha.start();  num.start();// 스레드 시작  }  }  **class** CharThread **extends** Thread { // 생성된 난수 만큼 문자 출력하는 스레드  **char** ch;  **int** in;// 출력할 문자와 횟수를 저장하기 위한 필드 선언  CharThread(**char** ch, **int** in) {// 매개변수로 문자와 횟수를 받아 필드 초기화하는 생성자{  **this**.ch = ch;  **this**.in = in;  }  **public** **void** run() {  **for** (**int** i = 0; i < in; i++) {// 횟수만큼 문자를 출력  System.***out***.print(ch);  }  }  }  **class** NumThread **extends** Thread { // 생성된 난수 만큼 숫자 출력하는 스레드  **int** num;  **int** in;// 출력할 숫자와 횟수를 저장하기 위한 필드 선언  NumThread(**int** num, **int** in) {// 매개변수로 숫자와 횟수를 받아 필드 초기화하는 생성자  **this**.num = num;  **this**.in = in;  }  **public** **void** run() {  **for** (**int** i = 0; i < in; i++) {// 횟수만큼 숫자를 출력  System.***out***.print(num);  }  }  } |
| **[실행결과]** |