|  |  |  |
| --- | --- | --- |
| **2019\_2\_java2\_12** | **학번 :** | **이름 :** |

* **내용 점검**

1. Map 인터페이스

|  |
| --- |
| **import** java.util.\*;  **class** Student {  **private** **int** number;  **private** String name;  **public** Student(**int** number, String name) {  **this**.number = number;  **this**.name = name;  }  **public** String toString() {  **return** name;  }  }  **public** **class** MapTest {  **public** **static** **void** main(String[] args) {  Map<String, Student> st = **new** HashMap<String, Student>();  st.put("20090001", **new** Student(20090001, "구준표"));  st.put("20090002", **new** Student(20090002, "금잔디"));  st.put("20090003", **new** Student(20090003, "윤지후"));  System.***out***.println(st); // 모든 항목을 출력한다.  st.remove("20090002"); // 하나의 항목을 삭제한다.  st.put("20090003", **new** Student(20090003, "소이정")); // 하나의 항목을 대치한다.  System.***out***.println(">>>>>>> 키값에 대응하는 value 출력");  System.***out***.println(st.get("20090003")); // 키 값에 해당하는 value 반환  System.***out***.println("\n >>>>>> 1: 람다식을 이용한 원소 출력 ");  st.forEach((key, value) -> {  System.***out***.println("key = " + key + ", value = " + value);  });  System.***out***.println("\n >>>>>> 2: Map.Entry 인터페이스를 이용한 원소 출력 ");  **for** (Map.Entry<String, Student> s : st.entrySet()) { // 모든 항목을 방문한다  String key = s.getKey();  Student value = s.getValue();  System.***out***.println("key=" + key + ", value=" + value);  }  }  } |
| **[실행 결과]** |

1. Collection 클래스 - 정렬

|  |
| --- |
| import java.util.\*;  class Student implements Comparable<Student>{  int number; String name;  public Student(int number, String name) {  this.number = number; this.name = name; }  public String toString() {  return "학번 : " + number + "\t이름 : " + name + "\n"; }  public int compareTo(Student s ){ //재정의  return this.number - s.number;  }  }  public class SetTest {  public static void main(String args[]) {  Student array[]={new Student(2009002, "김철수"), new Student(2009001, "이철수"),  new Student(2009003, "박철수")};  List<Student> list=Arrays.asList(array); //배열을 리스트로 변환  Collections.sort(list); //크기순으로 정렬  System.out.println(list);  }  } |
| **[실행결과]** |

1. 예외처리

|  |
| --- |
| **class** Account {  **int** balance; // 잔액  @Override  **public** String toString() {  **return** "Account [balance=" + balance + "]";  }  **int** withdraw(**int** amount) **throws** Exception {  **if** (balance < amount)  // 현재 메소드에서 발생하는 예외 종류를 표시하는 throws 절  **throw** **new** Exception("잔액이 부족합니다.");  balance -= amount;  **return** amount;  }  **int** deposit(**int** amount) **throws** Exception{  **if**(amount < 0)  **throw** **new** Exception("입금액이 음수 입니다.");  balance += amount;  **return** amount;  }  }  **public** **class** ExceptionExample {  **public** **static** **void** error(String str) {  System.***out***.println(str + "예외를 정상 처리 하였습니다 ");  }  **public** **static** **void** main(String args[]) {  Account obj = **new** Account();  **try** {  **int** amount = obj.withdraw(100000000); // 예외 발생 메소드  System.***out***.println("인출액:" + amount);  } **catch** (Exception e) { // 발생된 예외 처리  String msg = e.getMessage(); // 오류 메시지 출력  System.***out***.println(msg);  *error*("인출 ");  }  System.***out***.println(obj);  **try** {  **int** amount = obj.deposit(-100); // 예외 발생 메소드  System.***out***.println("입금액:" + amount);  } **catch** (Exception e) { // 발생된 예외 처리  String msg = e.getMessage(); // 오류 메시지 출력  System.***out***.println(msg);  *error*("입금");  }  System.***out***.println(obj);  }  } |
| **[실행결과]** |

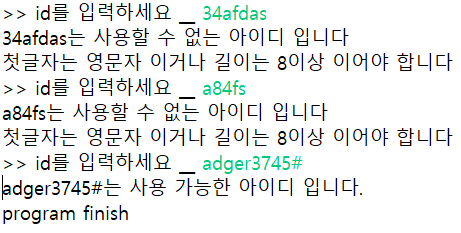
* **실습 과제**

1. HashMap을 사용하여 주사위를 100번 던졌을 때 주사위 눈의 빈도수를 계산하는 프로그램을 작성하시오

|  |  |
| --- | --- |
|  | **public** **class** Answer1 {  **public** **static** **void** main(String args[]) {  Map<Integer, Integer> map;  map=Count.*count*(); //빈도수를 계산한 HashMap을 반환받아 저장  System.***out***.println("Map에 저장된 빈도수를 출력합니다");  Count.*write*(map); //HashMap에 저장된 원소 출력  }  } |

|  |
| --- |
| **[프로그램 소스]** |
| **[실행결과]** |

1. 키보드로 입력 받은 문자열에 대하여 길이가 8미만이거나 첫 글자가 영문자가 아니면 예외를 발생시켜 “사용할 수 없는 아이디”를 출력하고, 길이가 8이상이고 첫 글자가 영문자 이면 “사용 가능한 아이디 입니다.” 를 출력하는 프로그램을 클래스를 최소 2개 사용하여 작성하세요.



|  |
| --- |
| **[프로그램 소스]** |
| **[실행결과]** |

1. 간단한 주소록을 처리하는 프로그램을 HashMap을 이용하여 작성하시오

- <id, Person>의 key와 value 저장, id는 String

- 임의의 데이터를 저장

- id를 입력하면 해당 Person 데이터 출력.

- HashMap에 저장된 모든 데이터 출력. 단, Map.Entry 인터페이스를 이용한 원소 출력

**class** Person{

**private String** name; //이름

**private String** num; //전화번호

**public** Person(String name, **String** num){

**this**.name=name;

**this**.num=num;

}

**public** String toString(){

**return** " [이름 : " + name + ", 전화번호 : " + num +"]" ;

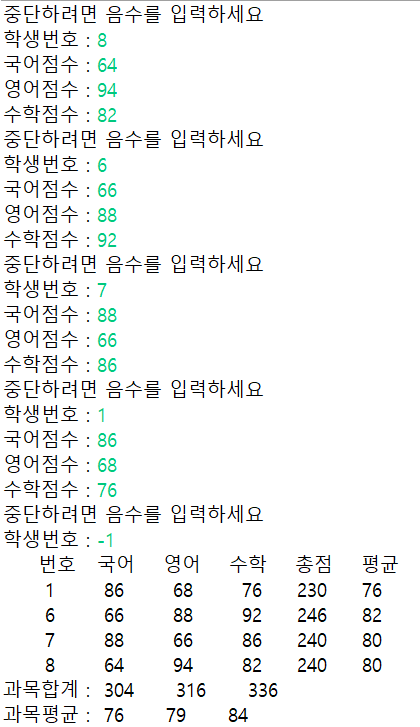
}

}

|  |
| --- |
| **[프로그램 소스]** |
| **[실행 결과]** |

1. 한 학급 학생들의 학생 번호와 국어, 영어, 수학 시험 점수를 입력 받아 학생 번호순으로 정렬하고 학생 개인의 총점과 평균 및 과목별 총점과 평균을 구하여 입력 자료와 함께 출력하는 성적처리 프로그램을 작성하시오.

* 조건 1: ArrayList를 사용할 것
* 조건 2: 번호 순으로 정렬할 때 Collection 클래스의 sort() 메소드를 사용할 것
* 조건 3: 2개 이상의 클래스로 나누어 작성할 것



|  |
| --- |
| **[프로그램 소스]** |
| **[실행 결과]** |

1. Book 클래스 이용하여 Book 정보를 ArrayList에 저장하고, 저장된 모든 데이터를 출력하는 프로그램을 작성하시오. 다음과 같은 메뉴를 제공하도록 한다.

1: 저장 2:대출 3:검색 4:리스트 출력 5. 삭제 6. 종료

public class Book {

private String number; //관리번호

private String title; //제목

private String author; //저자

private boolean status; // 대출 상태(대출 중-> true, 대출가능 -> false)

//생성자 – 본인작성

- 형식매개변수로 필드 초기화, 필드 status는 false로 할 것

// toString() 메소드 – 본인작성

- 객체 상태를 문자열로 반환(관리번호, 제목, 저자, 대출상태

//void rentBook() 메소드 – 본인작성

- status 필드값이 false이면 true로 변경한 후 “대출완료” 출력

- ture 이면 “대출 불가(대출 중)” 출력

//모든 필드에 대한 접근자 메소드 – 본인작성

}

import java.util.\*;

public BookTest{

public static void main(String[] args){

//메뉴를 선택

//1번 : ArrayList에 저장 – 필요한 자료는 입력을 받는다

//2번 : 대출하고자 하는 책 제목을 입력 받은 후 rentBook()메소드로 처리

//3번 : 검색하고자 하는 책 제목을 입력 받은 후 객체 상태 출력

//4번 : ArrayList에 저장 된 모든 원소 출력

//5번 : 입력된 관리번호와 일치하는 책을 ArrayList 에서 삭제

//6번 : 반복문 종료

}

}

|  |
| --- |
| **[프로그램 소스]** |
| **[실행결과]** |