|  |  |  |
| --- | --- | --- |
| **2019\_2\_java2\_04** | **학번 : 20135151** | **이름 : 이갑성** |

* **내용 점검**

1. 메소드에 대한 설명입니다. 빈칸에 알맞은 단어를 쓰세요
2. 리턴값이 없는 메소드는 리턴 타입을 ( void )로 해야 한다
3. 리턴 타입이 있는 메소드는 리턴값을 지정하기 위해 반드시 ( return )문이 있어야 한다
4. 매개변수의 개수를 모를 경우 ( … )를 이용해서 매개변수를 선언할 수 있다.
5. 같은 이름의 메소드를 여러 개 선언하는 것을 메소드 ( 오버로딩 )이라고 한다
6. 메소드 오버로딩에 대한 설명으로 맞는 것에 O표, 틀린것에 X표 하세요.
7. 동일한 이름의 메소드를 여러 개 선언하는 것을 말한다( O )
8. 반드시 리턴 타입이 달라야 한다( O )
9. 매개변수의 타입, 개수, 순서를 다르게 선언해야 한다( O )
10. 매개변수의 타입 및 개수에 따라 호출될 메소드가 선택된다 ( O )
11. 인스턴스 멤버와 정적 멤버에 대한 설명 중 맞는 것에 O표, 틀린것에 X표 하세요
12. 정적 멤버는 static으로 선언된 필드와 메소드를 말한다( O )
13. 인스턴스 필드는 생성자에서 초기화될 수 없다.( X )
14. 정적 필드와 메소드는 객체 생성없이 클래스를 통해 접근할 수 있다( O )
15. 인스턴스 필드와 메소드는 객체를 생성하고 사용해야 한다.( O )
16. MemberService 클래스에 login() 메소드와 logout() 메소드를 선언하려고 한다. login() 메소드를 호출할 때는 매개변수로 id와 password를 제공하고 logout() 메소드는 id만 매개변수로 제공한다.
17. login() 메소드는 id가 “hong”, password가 “12345”일 경우에만 true를 반환하고 그 외의 값은 false를 반한한다
18. logout() 메소드는 “로그아웃 되었습니다”가 출력되도록 하며 반환값은 없다.



**class** MemberService{

//본인이 완성할 것

**public** **boolean** login(String id, String password) {

**if**(id.equals("hong")) {

**if**(password.equals("12345"))

{

System.***out***.println("로그인 되었습니다.");

**return** **true**;

}

}

**return** **false**;

}

**public** **void** logout(String id) {

System.***out***.println(id + ": 로그아웃 되었습니다.");

}

}

**public** **class** MemberServiceTest {

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

//본인이 완성할 것

String id = "hong";

String password = "12345";

MemberService ms = **new** MemberService();

ms.login(id, password);

ms.logout(id);

}

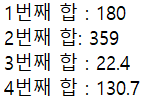
}

* 실행 결과 확인하기

|  |
| --- |
| **import** java.util.\*;  **class** Box{  **private** **int** ga, se, he; //가로, 세로, 높이 저장  **private** **int** volume; //부피 저장  **private** **static** **int** *cnt*=0; //생성된 객체 수를 저장하는 정적 필드    **public** Box() {  **this**(0,0,0); //오버로딩 된 생성자 호출, 항상 첫 문장에 작성  }  **public** Box(**int** ga, **int** se, **int** he){  **this**.ga=ga; **this**.se=se; **this**.he=he;  volume = ga\*se\*he; //부피 계산  *cnt*++;  }    **public** **int** getVolume(){  **return** volume;  }    **public** String toString(){  String str="가로 : " + ga +"\t세로 : " + se + "\t높이 : " + he;  str += "\t부피 : " + volume+"\t생성된 객체 수 : " + *cnt* ;  **return** str;  }  **public** **boolean** isSame(Box obj) {  //객체 내용을 비교하여 같으면 true 다르면 false 반환  **if**(ga==obj.ga && se==obj.se && he==obj.he)  **return** **true**;  **else**  **return** **false**;  }  }  **class** BoxMax{  //매개변수로 받은 객체의 필드 volume을 비교하여 큰 volume값을 갖는 객체를 반환  **public** **static** Box isMax(Box[] dim){  Box max=dim[0];  **for**(**int** i=1; i<dim.length;i++) {  **if**(max.getVolume() < dim[i].getVolume())  max=dim[i];  }  **return** max;  }  }  **public** **class** BoxTest {  **public** **static** **void** main(String[] args) {  Scanner in=**new** Scanner(System.***in***);  Box[] obj=**new** Box[5];    **for**(**int** i=0; i<obj.length;i++) {  System.***out***.println((i+1)+ "번 : 가로, 세로, 높이를 입력하세요 : ");  obj[i]=**new** Box(in.nextInt(), in.nextInt(),in.nextInt());  }  System.***out***.println("객체 내용 출력===");  **for**(Box temp : obj)  System.***out***.println(temp);    System.***out***.println("\n부피가 큰 박스 => \n" + BoxMax.*isMax*(obj));  }  **if**(obj[0].isSame(obj[1]))  System.***out***.println("동일한 객체 입니다");  **else**  System.***out***.println("서로 다른 객체 입니다");  }  } |
| **[실행결과]** |

* 실습 과제

1. 제시된 결과와 소스를 보고 VarType 클래스를 작성하시오. 단, 메소드를 호출할 때 인수 값은 임의로 할 것



**public** **class** VarTypeTest {

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

System.***out***.println("1번째 합 : " + VarType.*sum*(34,12,56,78));

System.***out***.println("2번째 합: " + VarType.*sum*(5, 67, 21, 99, 32, 45, 78, 12));

System.***out***.println("3번째 합 : " + VarType.*sum*(3.4, 1.2, 8.7, 5.4, 3.7));

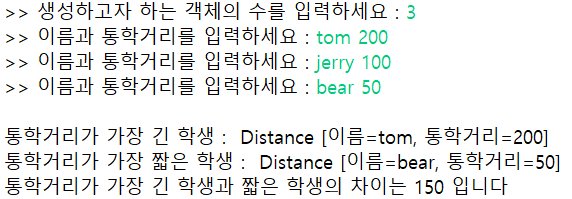
System.***out***.println("4번째 합 : " + VarType.*sum*(12.3,5.6,8.9,14.2,3.1,77.1,9.5));

}

}

|  |
| --- |
| **[프로그램 소스]**  **public** **class** VarType {  **public** **static** **int** sum(**int**... arr)  {  **int** result = 0;    **for**(**int** i = 0; i < arr.length; i++)  {  result += arr[i];  }    **return** result;  }  **public** **static** **double** sum(**double**... arr) {  **double** result = 0;    **for**(**int** i = 0; i < arr.length; i++)  {  result += arr[i];  }    **return** result;  }  }  **public** **class** VarTypeTest {  **public** **static** **void** main(String[] args) {  System.***out***.println("1번째 합 : " + VarType.*sum*(34,12,56,78));  System.***out***.println("2번째 합: " + VarType.*sum*(5, 67, 21, 99, 32, 45, 78, 12));  System.***out***.println("3번째 합 : " + VarType.*sum*(3.4, 1.2, 8.7, 5.4, 3.7));  System.***out***.println("4번째 합 : " + VarType.*sum*(12.3,5.6,8.9,14.2,3.1,77.1,9.5));  }  } |
| **[실행 결과]** |

1. 통학 거리가 가장 긴 학생과 짧은 학생, 그리고 거리 차이를 계산하여 출력하는 프로그램을 완성하세요.



**class** Distance{

**private** String name; //이름

**private** **int** dist; //통학거리

//매개변수로 필드값을 초기화하는 생성자 – 본인작성

//이름과 통학거리 필드에 대한 getter, setter 메소드 – 본인작성

//객체 내용(이름과 통학거리)를 문자열로 반환하는 toString() 메소드 – 본인작성

}

**class** DistanceComp{

//객체 배열을 매개변수로 받아 통학 거리가 가장 긴 객체를 반환하는 longdistance 정적 메소드 – 본인 작성

//객체 배열을 매개변수로 받아 통학 거리가 가장 짧은 객체를 반환하는 shortdistance() 정적 메소드 – 본인 작성

}

**public** **class** DistanceTest {

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

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

Distance[] dist; //객체 배열 선언

Distance longdist=**null**, shortdist=**null**;

System.***out***.print(">> 생성하고자 하는 객체의 수를 입력하세요 : ");

**int** cnt=in.nextInt();

dist =**new** Distance[cnt]; //객체 배열 생성

**for**(**int** i=0; i<cnt;i++){

System.***out***.print(">> 이름과 통학거리를 입력하세요 : ");

dist[i]=**new** Distance(in.next(), in.nextInt()); //객체 배열 초기화

}

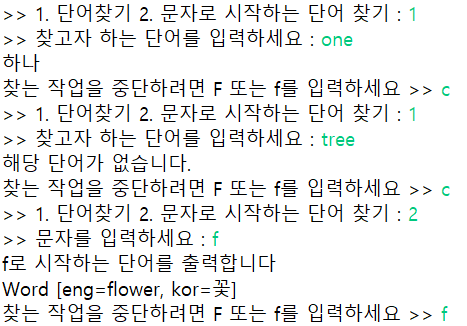
//제시된 결과처럼 출력될 수 있도록 나머지 부분을 완성하시오. – 본인작성

}

}

|  |
| --- |
| **[프로그램 소스]**  **public** **class** Distance {  **private** String name; //이름  **private** **int** dist; //통학거리  //매개변수로 필드값을 초기화하는 생성자 – 본인작성  **public** Distance(String name, **int** dist)  {  **this**.name = name;  **this**.dist = dist;  }  //이름과 통학거리 필드에 대한 getter, setter 메소드 – 본인작성  **public** **void** setName(String name) {  **this**.name = name;  }  **public** String getName() {  **return** name;  }  **public** **void** setDist(**int** dist)  {  **this**.dist = dist;  }  **public** **int** getDist() {  **return** dist;  }  //객체 내용(이름과 통학거리)를 문자열로 반환하는 toString() 메소드 – 본인작성  **public** String toString() {  **return** "Distance [이름= " + name + ", 통학거리= " + dist + "]";  }  }  **public** **class** DistanceComp {  //객체 배열을 매개변수로 받아 통학 거리가 가장 긴 객체를 반환하는 longdistance 정적 메소드 – 본인 작성  **public** **static** Distance longdistance(Distance[] obj) {  Distance longDist = obj[0];    **for**(**int** i = 1; i < obj.length; i++) {  **if**(longDist.getDist() < obj[i].getDist())  {  longDist = obj[i];  }  }  **return** longDist;  }  //객체 배열을 매개변수로 받아 통학 거리가 가장 짧은 객체를 반환하는 shortdistance() 정적 메소드 – 본인 작성  **public** **static** Distance shortdistance(Distance[] obj) {  Distance shortDist = obj[0];    **for**(**int** i = 1; i < obj.length; i++)  {  **if**(shortDist.getDist() > obj[i].getDist())  {  shortDist = obj[i];  }  }  **return** shortDist;  }  }  **import** java.util.\*;  **public** **class** DistanceTest {  **public** **static** **void** main(String[] args) {  Scanner in=**new** Scanner(System.***in***);  Distance[] dist; //객체 배열 선언  Distance longdist=**null**, shortdist=**null**;  **int** compareDistance = 0; //가장 긴 거리와 가장 짧은 거리의 차이값 저장 변수  System.***out***.print(">> 생성하고자 하는 객체의 수를 입력하세요 : ");  **int** cnt=in.nextInt();  dist =**new** Distance[cnt]; //객체 배열 생성  **for**(**int** i=0; i<cnt;i++){  System.***out***.print(">> 이름과 통학거리를 입력하세요 : ");  dist[i]=**new** Distance(in.next(), in.nextInt()); //객체 배열 초기화  }    //제시된 결과처럼 출력될 수 있도록 나머지 부분을 완성하시오. – 본인작성  longdist = DistanceComp.*longdistance*(dist);  shortdist = DistanceComp.*shortdistance*(dist);  System.***out***.println("통학거리가 가장 긴 학생: " + longdist);  System.***out***.println("통학거리가 가장 짧은 학생: " + shortdist);  compareDistance = longdist.getDist() - shortdist.getDist();  System.***out***.println("통학거리가 가장 긴 학생과 짧은 학생의 차이는 " + compareDistance + " 입니다.");    }  } |
| **[실행 결과]** |

1. 키보드로 영 단어를 입력하면 단어 뜻을 한글로 출력하는 프로그램을 작성하시오. 영 단어와 한글은 객체 배열 생성시 초기화 하도록 한다.



**class** Word {

**private** String eng; //영 단어

**private** String kor; //영 단어의 의미

//매개변수로 필드를 초기화 하는 생성자 – 본인작성

//각 필드에 대한 getter 메소드 – 본인작성

// 객체 내용을 문자열로 반환하는 toString() 메소드 – 본인작성

}

**class** WordProcess {

//Word 객체 배열과 찾고자 하는 영 단어를 매개변수로 받아, 영 단어에 해당하는 한글 단어를 반환하는 search() 메소드, 클래스 이름으로 호출되도록 할 것 – 본인 작성

// Word 객체 배열과 문자를 매개변수로 받아, 매개변수로 받은 문자로 시작하는 단어를 출력하는 search() 메소드, 반환값은 없으며 클래스 이름으로 호출되도록 할 것. 단, 해당 문자로 시작하는 단어가 없으면 단어가 없음을 출력할 것 – 본인 작성

}

**public** **class** WordTest {

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

//객체 배열을 선언하면서 초기화 할 때 10개 이상의 단어를 사용할 것

Word str[] = { **new** Word("one", "하나"), **new** Word("student", "학생"), **new** Word("apple", "사과"),

**new** Word("school", "학교"), **new** Word("flower", "꽃") };

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

//반복문을 사용하여 제시된 결과처럼 출력될 수 있도록 main() 메소드를 완성할 것

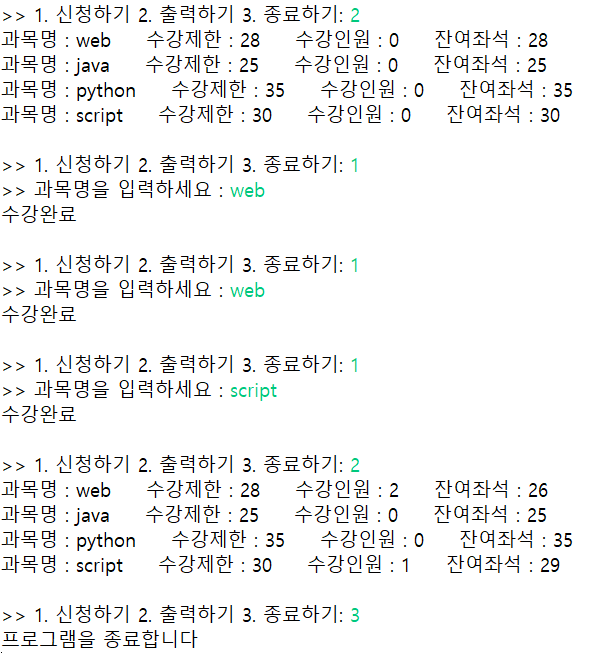
//단어를 검색할 때 WordProcess 클래스에 있는 메소드를 사용할 것

}

}

|  |
| --- |
| **[프로그램 소스]**  public class Word {  private String eng; //영 단어  private String kor; //영 단어의 의미  //매개변수로 필드를 초기화 하는 생성자 – 본인작성  public Word(String eng, String kor) {  this.eng = eng;  this.kor = kor;  }  //각 필드에 대한 getter 메소드 – 본인작성  public String getEng() {  return eng;  }  public String getKor() {  return kor;  }  // 객체 내용을 문자열로 반환하는 toString() 메소드 – 본인작성  public String toString() {  return "Word [eng=" + eng + ", kor=" + kor + "]";  }  }  public class WordProcess {  //Word 객체 배열과 찾고자 하는 영 단어를 매개변수로 받아, 영 단어에 해당하는 한글 단어를 반환하는 search() 메소드, 클래스 이름으로 호출되도록 할 것 – 본인 작성  public static String search(Word[] obj, String findEng) {  String findKor = null;    for(int i = 0; i < obj.length; i++)  {  if(obj[i].getEng().equals(findEng))  {  findKor = obj[i].getKor();  break;  }  }    return findKor;  }  // Word 객체 배열과 문자를 매개변수로 받아, 매개변수로 받은 문자로 시작하는 단어를 출력하는 search() 메소드, 반환값은 없으며 클래스 이름으로 호출되도록 할 것.  //단, 해당 문자로 시작하는 단어가 없으면 단어가 없음을 출력할 것 – 본인 작성  public static void search(Word[] obj, char firstEng)  {  boolean find = false;    for(int i = 0; i < obj.length; i++)  {  if(obj[i].getEng().charAt(0) == firstEng)  {  System.***out***.println(obj[i]);  find = true;  }    }  if(find == false)  System.***out***.println(firstEng + "으로 시작하는 단어가 없습니다.");  }    }  import java.util.\*;  public class WordTest {  public static void main(String[] args) {  Scanner in = new Scanner(System.***in***);  // 객체 배열을 선언하면서 초기화 할 때 10개 이상의 단어를 사용할 것  Word str[] = { new Word("one", "하나"), new Word("student", "학생"), new Word("apple", "사과"),  new Word("school", "학교"), new Word("flower", "꽃"), new Word("computer", "컴퓨터"), new Word("child", "아이"),  new Word("snow", "눈"), new Word("hard", "고된"), new Word("pretty", "이쁜") };  // 반복문을 사용하여 제시된 결과처럼 출력될 수 있도록 main() 메소드를 완성할 것  // 단어를 검색할 때 WordProcess 클래스에 있는 메소드를 사용할 것  char keepGoing, firstEng;  String findEng, findoutKor;  int menuSelect;  do {  System.***out***.print(">> 1.단어찾기 2.문자로 시작하는 단어 찾기: ");  menuSelect = in.nextInt();  switch (menuSelect) {  case 1:  System.***out***.print(">>찾고자 하는 단어를 입력하세요: ");  findEng = in.next();  findoutKor = WordProcess.*search*(str, findEng);  if (findoutKor != null)  System.***out***.println(findoutKor);  else  System.***out***.println("해당 단어가 없습니다.");  break;  case 2:  System.***out***.print(">>문자를 입력하세요: ");  firstEng = in.next().charAt(0);  System.***out***.println(firstEng + "로 시작하는 단어를 출력합니다.");  WordProcess.*search*(str, firstEng);  break;  default:  System.***out***.println("메뉴에 없는 번호를 입력하셨습니다.");  break;  }  System.***out***.print("찾는 작업을 중단하려면 F또는 f를 입력하세요 >> ");  keepGoing = in.next().charAt(0);  } while (keepGoing != 'f' && keepGoing != 'F');  System.***out***.println("프로그램을 종료합니다.");  }  } |
| **[실행 결과]** |

1. 다음과 같이 수강신청을 처리하는 프로그램을 작성하시오. 단, 잔여 좌석이 없으면 “잔여 좌석이 없습니다”를 출력하도록 한다. 과목을 입력하면 해당과목의 수강인원을 1씩 증가시킨다.



**class** Sugang {

**private** **int** maxcount, count, seat; //수강최대인원, 현재 수강신청한 인원, 잔여좌석

//필요한 메소드를 본인이 추가하여 완성할 것

}

**public** **class** SugangTest {

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

Sugang[] sugang= {**new** Sugang(28, "web"), **new** Sugang(25, "java"), **new** Sugang(35, "python"),

**new** Sugang(30, "script")};

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

//제시된 결과를 참조하여 main() 메소드를 본인이 완성할 것

}

}

|  |
| --- |
| **[프로그램 소스]**    public class Sugang {  private String lecture;  private int maxcount; // 수강최대인원  private int count; // 현재 수강신청한 인원  private int seat; // 잔여좌석  // 필요한 메소드를 본인이 추가하여 완성할 것  public Sugang(int maxcount, String lecture) {  this.maxcount = maxcount;  this.lecture = lecture;  this.count = 0;  this.seat = maxcount;  }    //과목명의 get메소드  public String getLecture() {  return lecture;  }    //현재 수강신청한 인원의 get&set메소드  public int getCount() {  return count;  }  public void setCount(int cnt) {  count += cnt;  }    //현재 잔여좌석의 get&set메소드  public int getSeat() {  return seat;  }  public void setSeat(int cnt) {  seat -= cnt;  }    public String toString() {  return "과목명: " + lecture + "\t수강제한: " + maxcount +  "\t수강인원: " + count + "\t잔여좌석: " + seat;  }    }  public class SugangHub {  public static void apply(Sugang[] obj, String lecture)  {  boolean find = false;  for(int i = 0; i < obj.length; i++) {  if(obj[i].getLecture().equals(lecture))  {  find = true;  if(obj[i].getSeat() == 0) {  System.***out***.println("잔여 좌석이 없습니다.");  }  else  {  obj[i].setCount(1);  obj[i].setSeat(1);  System.***out***.println("수강완료");  }  }  }  if(find == false) {  System.***out***.println("개설되지 않은 강좌입니다.");  }  }  public static void printLecture(Sugang[] obj) {  for (int i = 0; i < obj.length; i++) {  System.***out***.println(obj[i]);  }  }  }  import java.util.\*;  public class SugangTest {  public static void main(String[] args) {  Scanner sc = new Scanner(System.***in***);  Sugang[] sugang = { new Sugang(28, "web"), new Sugang(25, "java"), new Sugang(35, "python"),  new Sugang(30, "script") };  // 제시된 결과를 참조하여 main() 메소드를 본인이 완성할 것  int selectMenu = 0;  String lecture;  do {  System.***out***.print(">> 1.신청하기 2.출력하기 3.종료하기 : ");  selectMenu = sc.nextInt();  switch(selectMenu)  {  case 1:  System.***out***.print(">> 과목명을 입력하세요: ");  lecture = sc.next();  SugangHub.*apply*(sugang, lecture);  break;  case 2:  SugangHub.*printLecture*(sugang);  break;    default:  break;  }  }while(selectMenu != 3);    System.***out***.println("프로그램을 종료합니다.");  }  } |
| **[실행 결과]** |