# <프로젝트 최종 보고서 – 뮤지엄 버킷리스트>

서양화.미술경영학과 201512651 김진웅

## 1. 목표와 동기

미술 데이터를 활용한 프로젝트를 진행하고 싶었습니다. 그런데 미술 전시가 대중에게 쉽게 노출이 되어 있음에도 불구하고 유저 데이터를 수집할 수 있는 플랫폼이 잘 형성되어 있지 않았습니다. 반대의 사례로 영화의 경우 네이버 리뷰, 비디오 대여 플랫폼인 넷플릭스 등 데이터를 수집할 수 있는 플랫폼이 잘 구축 되어 있었습니다. 미술 데이터 수집을 어떻게 해야 할지 계속 고민하다가 방향을 바꾸어 제가 처해있는 문제였던 어떻게 하면 미술 데이터를 쉽게 수집할 수 있을까에 초점을 맞추게 되었습니다. 그러던 중 여행 버킷리스트에서 아이디어를 얻어서 뮤지엄 버킷리스트 프로젝트를 진행하게 되었습니다. 버킷리스트가 좋은 플랫폼이라고 생각한 이유는 유저 데이터를 쉽게 수집할 수 있다는 점입니다. 유저의 가입 정보를 기반으로 선호도 데이터를 수집해 통계 정보를 쉽게 획득할 수 있으며 더 나아가서 커뮤니티로까지 발전을 시킨다면 유저 리뷰 데이터까지 확보할 수 있기 때문입니다. 하지만 본 프로젝트에서는 완성도를 위하여 기초 기능의 탄탄한 구현을 목표로 했고 따라서 로그인, 버킷리스트, 통계 기능을 목표로 설정했습니다.

## 2. 데이터 설명

메인 데이터는 공공데이터 포털에 게시되어 있는 전국 박물관, 미술관 데이터를 이용했습니다. 그러나 해당 데이터 셋에 중복 데이터와 누락 데이터가 많아 데이터를 가공해야 했습니다. 레코드가 1000개를 조금 넘는 소량의 데이터였기 때문에 중복 및 누락 데이터는 엑셀로 간단히 전처리했습니다. 추가적으로 데이터 사이에 공백 등 세세한 부분을 바꾸는 경우는 파이썬을 가지고 데이터를 가공했습니다. (데이터 자체가 소량이라 라이브러리 없이 간단히 진행할 수 있었습니다.)

가공한 데이터가 세부 분류가 가능하지도 않았고, N : M의 관계로 엮기가 어렵다보니 미술관의 성격에 따라 세부 키워드를 추가하는 작업을 진행했습니다. 카테고리 파일이 존재하며 메인 데이터에는 넘버링만 있어 두 데이터 파일이 엮이는 구조입니다. 세부 키워드를 추가하는 작업은 엑셀 함수로 미술관 이름을 적극 활용했고 이름으로 걸러지지 않는 정보들은 직접 찾아서 뮤지엄의 세부 성격을 추가하는 과정을 거쳤습니다.

## 3. 메뉴 & 주요기능 코드 설명

(1) 로그인 & 회원가입

회원가입을 제공합니다. 회원가입에 성공하면 유저 데이터베이스 파일에 ID와 PW가 추가됩니다. 로그인시 ID와 PW를 정확히 입력하면 유저 개인의 데이터 파일을 오픈하는 구조입니다.

[signUp 메서드 코드]

```
void signUp(Scanner scan) throws IOException {
    boolean check2 = true;
    System.out.println("회원가입을 진행합니다.\n"
            + "*비밀번호는 5자 이상으로 설정해주세요.\n");
    System.out.print("ID >> ");
    id = scan.next();
    System.out.print("PW >> ");
    pw = scan.next();
    if (pw.length() < 4) {
        System.out.println("비밀번호는 5자 이상으로 설정하세요.\n");
        check2 = false;
    }

    if (!check2) {
        System.out.println("회원가입을 다시 진행해주세요. \n");
    }

    if (check2) {
        br = new BufferedReader(
                new FileReader("c:/MUSEUM_BUCKET/data/User.txt"));

        while (true) {
            idpw = new String[2];
            check = false;

            while (true) {
                check = false;
```

```
while (true) {
    check = false;
    String line = br.readLine();
    if (line == null) {
        break;
    }
    idpw = line.split(" ");
    if (idpw[0].equals(id)) {
        check = true;
        break;
    }
}
if (check) {
    System.out.println(
            "\n" + id + "는(은) 이미 사용중인 ID입니다.\n");
    break;
}

if (!check) {
    openCheck();
    BufferedWriter fw = new BufferedWriter(
            new FileWriter("c:/MUSEUM_BUCKET/data/User.txt", true
    fw.newLine();
    fw.write(id + " " + pw);
    System.out.println("회원가입이 완료되었습니다.\n");
    fw.close();
```

회원가입을 진행하면 회원 데이터베이스를 오픈하여 입력받은 정보가 존재하는지 체크합니다. 성공적으로 회원가입을 마치면 유저 데이터베이스에 입력 받은 정보가 추가됩니다.

회원가입

```
>> 2
회원가입을 진행합니다.
*비밀번호는 5자 이상으로 설정해주세요.

ID >> testID
PW >> 12341234
회원가입이 완료되었습니다.


====================
 Museum BucketList
====================

(1) 로그인
(2) 회원가입

(0) 종료
====================
```

유저 데이터베이스에 추가

```
User.txt - 메모장
파일(F)  편집(E)  서식(O)  보기(V)
kim 12341234
park 12341234
choi 12341234
jin 12341234
kimjin 12341234
chain 12341234
black 12341234
red 12341234
blue 12341234
yellow 12341234
jinsang 12341234
jinsang2 12341234
Jinung 12341234
ki 12341234
jins 1234
testID 12341234
```

로그인

```
====================
 Museum BucketList
====================

(1) 로그인
(2) 회원가입

(0) 종료
====================

>> 1
ID >> testID
PW >> 12341234
====================
  testID님 환영합니다.
====================
```

(설문조사에서 비회원도 사용할 수 있는 게 더 좋지 않겠냐는 피드백을 받았는데 유저 데이터를 수집을 목표로 하고 있기 때문에 로그인을 해야만 사용할 수 있도록 설계했습니다.)

(2) 검색 기능 – 1. 자율 검색/ 2. 키워드 검색/ 3. 전체 출력

- 1. 자율 검색

자율 검색은 자유로운 키워드 하나를 입력받아 해당 키워드와 매치되는 데이터들을 출력하는 기능입니다.

[kwdSearch 메서드(자율 검색) 코드]
키워드가 걸리기 제일 쉬운 주소 필드부터 시작하여 데이터와 키워드가 매치되면 출력하고 check를 ture로 주는 패턴을 사용했습니다. 또한 유저가 검색 기능을 사용하면 검색 기능뿐 아니라 유저 검색어 데이터 파일에 검색어 데이터를 저장하여 수집하기도 합니다.
user.allData.write(kwd)가 해당 코드입니다.

```
void kwdSearch(Main main, User user, Scanner scan) {
    boolean check = false;
    String kwd = null;
    System.out.println(
            "검색 예시: 국립, 사립, 서울시, 미술관, 테마파크...");
    System.out.print(">> ");
    kwd = scan.next();

    try {
        user.bw2.write(kwd + " ");
        user.openMasterData();
        user.allData.write(kwd);
        user.allData.newLine();
        user.refreshSerach();
        user.refreshMaster();
    } catch (IOException e) {
        e.printStackTrace();
    }

    if (!check) {
        for (Museum a : main.museumList) {
            if (a.address.contains(kwd.substring(0, 2))) {
                a.print();
                check = true;
            }
        }
        System.out.println();
    }
```

```
    if (kwd.endsWith("관") || kwd.equals("테마파크") ||
            kwd.equals("갤러리")){

        for (Museum a : main.museumList) {
            if (a.category.style.equals(kwd)) {
                a.print();
                check = true;
            }
        }
        System.out.println();
    }

    if(!check) {
        for (Museum a : main.museumList) {
            if (a.name.contains(kwd)) {
                a.print();
                check = true;
            }
        }
        System.out.println();
    }

    if (!check) {
        System.out.println("검색 결과가 없습니다.");
    }
```

[자율 검색 실행화면]

```
>> 1
(1) 검색
(2) 추천 검색
(3) 전체 출력

(0) 뒤로가기

>> 1
검색 예시: 국립, 사립, 서울시, 미술관, 테마파크...
>> 테마파크
[사립] [테마파크]                    제주유리의성   제주특별자치도 제주시 한경면 녹차분재로 462
[사립] [테마파크]                 테마공원선녀와나무꾼   제주특별자치도 제주시 조천읍 선교로 267
[사립] [테마파크]                    돌하르방공원   제주특별자치도 제주시 조천읍 북촌서1길 70
[사립] [테마파크]                      파파퀼드   제주특별자치도 제주시 애월읍 평화로 2157
[사립] [테마파크]                      제주민속촌   제주특별자치도 서귀포시 표선면 민속해안로 631-34
[사립] [테마파크]                   박물관은살아있다   제주특별자치도 서귀포시 중문관광로 42
[사립] [테마파크]                   헬로키티아일랜드   제주특별자치도 서귀포시 안덕면 한창로 340
[사립] [테마파크]                   포레스트판타지아   제주특별자치도 서귀포시 안덕면 일주서로 1836
[사립] [테마파크]          세계오지민속박물관(베니스랜드)      제주특별자치도 서귀포시 성산읍 서성일로 474
[사립] [테마파크]                   아쿠아플라넷여수   전라남도 여수시 오동도로 61-11
[사립] [테마파크]                   서울그레뱅뮤지엄   서울특별시 중구 을지로 23(을지로1가)
[사립] [테마파크]                   63아쿠아플라넷   서울특별시 영등포구 63로 50
[사립] [테마파크]                  롯데월드아쿠아리움   서울특별시 송파구 올림픽로300
[공립] [테마파크]                     서울상상나라   서울특별시 광진구 능동로 216
```

- 2. 추천 검색

[추천 검색 메서드 실행 순서]

```
se 1:
  int n2 = 0;
  System.out.printf(
    "(1) 검색%n"
  + "(2) 추천 검색%n"
  + "(3) 전체 출력%n%n"
  + "(0) 뒤로가기%n%n");
  System.out.print(">> ");
  n2 = keyin.nextInt();
  if (n2 == 0)
      break;

  if (n2 == 1) {
      userTaste.kwdSearch(this, user,
      break;
  }

  if (n2 == 2) {
      userTaste.localCheck(keyin);
      userTaste.styleCheck(keyin);
      userTaste.printTaste(this);
      break;
  }
```

추천 검색은 버킷리스트 검색 기능의 진입 장벽을 낮추기 위해서 선호 지역과 키워드를 입력받아 추천 데이터를 출력합니다.

localCheck, styleCheck printTaste 순으로 메서드가 진행됩니다. 차례대로 지역, 키워드를 입력받아 임시 리스트에 저장한 뒤에 전체 뮤지엄 리스트와 for loop를 돌려서 리스트에 존재하는 모든 키워드와 데이터와 매치되는 경우 데이터를 출력하는 구조입니다.

[추천 검색 실행화면]

```
원하는 지역들을 입력하세요
예시: 서울시, 수원시, 부산시

종료버튼: 0

>> 수원시
>> 0
키워드 번호를 입력하세요.
(1) 박물관
(2) 미술관
(3) 갤러리
(4) 기념관
(5) 역사관
(6) 과학관
(7) 테마파크
(8) 문화관
(0) 종료

>> 1
>> 2
>> 3
>> 0
[공립] [ 박물관]                    수원화성박물관   경기도 수원시 팔달구 창룡대로 21
[공립] [ 박물관]                      수원박물관   경기도 수원시 영통구 창룡대로 265
[공립] [ 박물관]                    수원광교박물관   경기도 수원시 영통구 광교로 182
[공립] [ 미술관]                  수원시립아이파크미술관   경기도 수원시 팔달구 정조로 833
```

- 3. 전체 출력

모든 뮤지엄 데이터를 출력합니다. (주요 기능이 아니라 생략했습니다.)

(3) 버킷리스트 기능 – 1.버킷리스트 출력/ 2.데이터 추가, 삭제/ 3.방문 체크, 삭제

- 1. 버킷리스트 출력
유저 버킷리스트에 존재하는 모든 데이터를 출력합니다. user.id 필드를 이용하여 유저 개인 데이터 파일(user.txt)을 읽고 출력합니다. (로그인을 하면 user.txt, user_search.txt 파일을 오픈합니다.)

- 2. 데이터 추가, 삭제
버킷리스트에 데이터를 추가, 삭제하는 기능입니다. (유저 데이터 파일에 데이터를 추가, 삭제 합니다.)

[addPlace(데이터 추가) 코드 일부]

버킷리스트에 인풋 데이터가 존재하는지 확인을 거치고 중복 데이터가 없는 경우 인풋 데이터를 뮤지엄을 반환하는 메서드에 매개변수로 넣습니다. 일치하는 뮤지엄 객체를 찾고 그 데이터를 user.txt 파일에 객체의 필드인 스타일과 네임을 저장합니다. 아래 코드를 보면 유저 데이터 파일에만 저장하는 것이 아니라 user.bw_all통해 MUSEUMALLDATA.txt 파일에도 같이 저장합니다. 이 데이터는 뮤지엄 랭킹 기능에 활용하기 위해 저장하는 데이터입니다.

```
if (!check) {
    for (Museum a : main.museumList) {
        if (a.name.equals(input)) {
            System.out.printf(
                    "'%s' 버킷리스트에 추가 되었습니다.%n",
                    input);
            user.bw.write(a.category.style + " ");
            user.bw.write(a.name);
            user.bw.newLine();
            user.refresh();
            user.openAllData();
            user.bw_all.write(a.category.style + " ");
            user.bw_all.write(a.name);
            user.bw_all.newLine();
            user.refreshAllData();
            check = true;
            break;
```

기본적으로 모든 메서드들은 데이터베이스 혹은 유저 데이터 파일에 이미 데이터가 존재하는 경우 혹은 존재하지 않는 경우를 체크하고 조건을 만족하는 경우에만 실행되도록 설계되어있습니다.

*버킷리스트에 없는 데이터를 삭제하려 할시 안내 메시지 출력

```
제거할 장소를 입력해주세요.
종료 버튼: 0
>> 수원화성박물관
'수원화성박물관'이 버킷리스트에 존재하지 않습니다.
>>
```

*버킷리스트에 이미 존재하는 데이터 추가 입력할시 이미 존재한다는 메시지 출력

```
[O] [사립] [ 박물관]                    가천박물관   인천광역시 연수구 청량로102번길 40-9
[O] [공립] [ 미술관]                 수원시립아이파크미술관   경기도 수원시 팔달구 정조로 833
[O] [사립] [ 박물관]                    박물관수   대구광역시 수성구 국채보상로186길 79
[O] [사립] [ 미술관]                     보임쉔   전라남도 담양군 대전면 신룡길 79-3
```

```
>> 보임쉔
이미 버킷리스트에 보임쉔이 존재합니다.
>>
```

[데이터 추가, 삭제 실행화면]

```
추가할 장소를 입력해주세요.
종료버튼: 0

>> 일현미술관
'일현미술관' 버킷리스트에 추가 되었습니다.
>>
```

```
>> 3
제거할 장소를 입력해주세요.
종료 버튼: 0
>> 일현미술관
삭제 완료
>>
```

- 3. 방문 체크, 삭제

방문한 뮤지엄을 기록하는 기능입니다. user_check.txt에 데이터 입력되며 버킷리스트를 출력하면 방문한 곳이 O로 표시됩니다.

[방문 체크 코드 일부]

```
void visitCheck(Scanner scan, Main main, User user) throws IOException {
    boolean check = false;
    boolean check2 = false;
    String ch = null;
    String muse = null;
    String muse2 = null;
    ArrayList<String> bucketList = new ArrayList<>();
    ArrayList<String> bucketList2 = new ArrayList<>();
    Scanner scan2 = main.openFile("c:/MUSEUM_BUCKET/data/" + user.id +".txt");
    Scanner scan3 = main.openFile("c:/MUSEUM_BUCKET/data/" + user.id +"___check.txt");
```

user.id.txt 파일과 user.id_check.txt 파일을 오픈하여 리스트에 저장한 뒤 이미 방문 체크가 됐거나 버킷리스트에 존재하지 않는 경우를 걸러내는 작업을 가장 먼저 실시하게 됩니다. 위 조건을 만족하는 경우 다음 코드로 넘어가게 됩니다.

```
        if(!check) {
            for (String a : bucketList) {
                if(a.equals(ch)) {
                    user.openCheck();
                    user.che.write(ch);
                    user.che.newLine();
                    user.refreshCheck();
                    System.out.printf("'%s' 방문체크 완료 \n\n", a);
                    check2 = true;
                    break;
```

위 점검 과정을 거쳐서 조건을 만족하면 user_check.txt파일에 해당 데이터가 추가됩니다. 버킷리스트를 출력할 때 두 파일(user, check)을 같이 읽고 양쪽에 데이터가 동시에 존재하는 경우 방문 표시(O)가 되는 구조로 구현했습니다. 반대로 삭제의 경우에는 user_check.txt 데이터를 임시 리스트에 받고 삭제하려는 데이터만 리스트에서 제거하고 OverWrite 방식으로 user_check.txt에 덮어쓰는 방식입니다.

[방문 체크 구조 및 실행 화면]


- 방문 체크시 user__check.txt 에 데이터를 추가합니다.
- 버킷리스트 출력시 두 파일을 같이 읽어서 양쪽에 존재하는 경우 방문체크로 표시합니다.
- 방문 삭제의 경우 user__check.txt파일을 Over Write 하는 방법을 채택했습니다.

(4) 통계 기능 – 1. 사용자 통계/ 2. 뮤지엄 랭킹/ 3. 뮤지엄 추천

- 1. 사용자 통계
사용자의 버킷리스트 데이터들을 가지고 지역, 키워드, 국공립 통계를 내는 기능입니다.

[사용자 통계 코드 일부]

```
System.out.println("지역 통계");
int k2 = 1;
for (int a : localRank2) {
    for (String b : localMap.keySet()) {
        if (localMap.get(b) == a) {
            System.out.printf("[%s위] ", k2);
            System.out.printf(
                    "%s %.2f%%%n",
                    b, (double) a / style.size() * 100 );
            k2++;
```

버킷리스트에서 주소, 국공립, 키워드를 가지고 각각 리스트에 받고 빈도수를 내림차순으로 정렬해서 출력합니다. 위 코드는 지역 통계를 출력하는 일부분입니다. 정렬은 Collections 라이브러리의 sort기능을 사용했습니다.

[사용자 통계 실행 화면]

```
====================
<버킷리스트 통계>
====================
지역 통계
[1위]  인천 33.33%
[2위]  서울 28.57%
[3위]  강원 14.29%
[4위]  대구 9.52%
[5위]  경기 4.76%
[6위]  제주 4.76%
[7위]  충청 4.76%
```

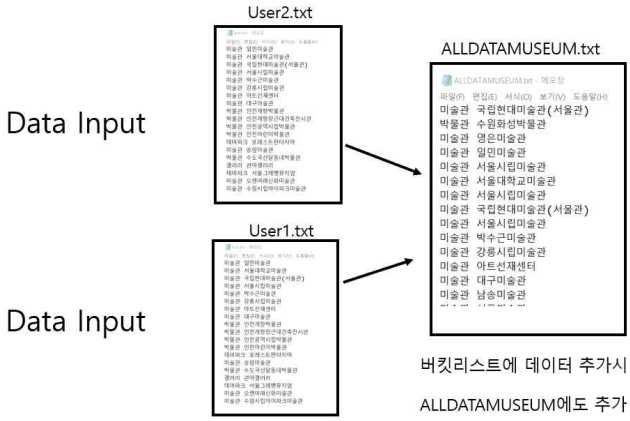
```
키워드 통계
[1위]  미술관 52.38%
[2위]  박물관 33.33%
[3위]  테마파크 9.52%
[4위]  갤러리 4.76%

국공립 통계
[1위]  공립 52.38%
[2위]  사립 38.10%
[3위]  국립 4.76%
[4위]  대학 4.76%
```

```
====================
<유저 검색어 통계>
====================

[ 1위]  미술관
[ 2위]  국립
[ 3위]  공립
[ 4위]  테마파크
[ 5위]  수원
[ 6위]  사립

 *유저 검색 데이터를 기반으로 합니다.
```

- 2. 뮤지엄 랭킹

유저들이 버킷리스트에 데이터를 추가하면 개인 유저 데이터 파일뿐 아니라 ALLDATAMUSEUM.txt 파일에도 해당 데이터가 추가됩니다. 따라서 모든 유저들의 데이터가 축적되어 있는 전체 데이터에서 뮤지엄의 빈도수를 내림차순으로 정렬하여 출력합니다.

[뮤지엄 랭킹 기능 구조]




[뮤지엄 랭킹 기능 코드 일부]

```java
// 1단계 중복 뮤지엄 제거
for (String a : rankMuseum) {
    if (nMuseum.containsKey(a)) {
        nMuseum.put(a, nMuseum.get(a)+1);
    }
}
//2 단계 카운팅만 빼와서 내림차순 정렬하기
for (int a : nMuseum.values()) {
    ranks.add(a);
}

Collections.sort(ranks, Collections.reverseOrder());

// 3단계 카운팅 중복 제거
for(int a : ranks) {
    if(!rankss.contains(a)) {
        rankss.add(a);
```

많이 사용했던 코드 패턴으로 파일을 해쉬맵과 리스트의 형태로 읽어서 중복 뮤지엄의 밸류값을 체크했습니다. **리스트에는 중복 요소가 허용되지만 해쉬맵은 중복 키 값을 갖지 않는** 다는 자료구조를 적극 활용했습니다. 다음으로 Collections 라이브러리의 sort를 이용해서 내림차순으로 정렬하는 코드 패턴입니다.

- 3. 뮤지엄 추천

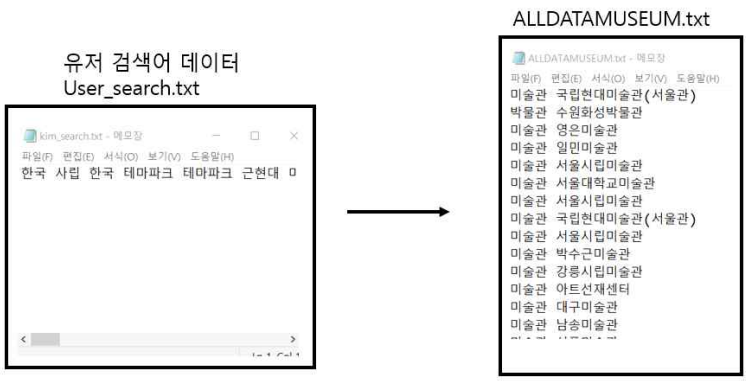
뮤지엄 추천 기능은 사용자 검색 통계와 뮤지엄 랭킹 데이터를 활용한 기능으로 유저가 자율 검색 기능을 이용할 때 키워드를 입력하면 해당 키워드는 user___search.txt 데이터에 입력이 됩니다. 이렇게 누적된 데이터 파일에서 최빈값 데이터를 가지고 뮤지엄 랭크 데이터에 매치되면서 유저의 버킷리스트에 존재하지 않을 경우 추천하는 알고리즘입니다.

[recomed 메서드 코드 일부]

유저 검색어 최빈값 데이터, 뮤지엄 랭크 데이터를 매치시키는 코드 부분입니다.

```
for (String a : rankMuseum) {
    m = museum.returnMuseum(main, a);
    // 해당 뮤지엄의 객체를 반환

    if(m.national.equals(f)) {
        treat.add(m);
    }

    if(m.name.contains(f)) {
        treat.add(m);
    }

    if(m.category.style.equals(f)) {
        treat.add(m);
    }
}

for (Museum a : treat) {
    if(!resultTreat.contains(a) && !checkList.contains(a.name)) {
        resultTreat.add(a);
    }
```

[recomend 메서드 구조]




- 개인 유저의 최상위 빈도수 검색 키워드를 가지고 뮤지엄 랭킹의 상위권에 존재하는 데이터를 매치시켜서 일치하는 경우 추천
- 단 이미 버킷리스트에 존재할 경우 제외

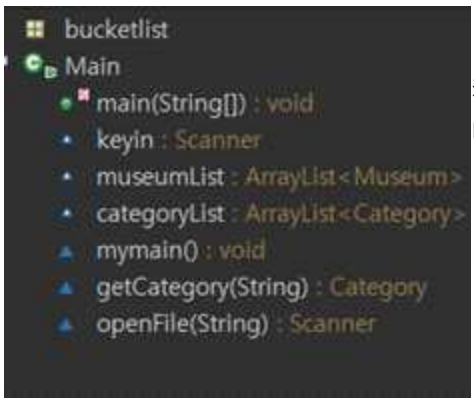
[추천 기능 실행 화면]

검색 최빈값 데이터를 기반으로 뮤지엄 랭킹 데이터와 매치되면서 버킷리스트에 존재하지 않는 데이터들을 출력한 결과 화면입니다.

```
============================

  kim님은 미술관 키워드를 추천합니다.

============================
   <추천 뮤지엄 리스트>

[사립] [ 미술관]                    영은미술관   경기도 광주시 청석로 300
[사립] [ 미술관]                    남송미술관   경기도 가평군 북면 백둔로 322
[사립] [ 미술관]                    신풍미술관   경상북도 예천군 지보면 신풍1리길 50
[공립] [ 미술관]          전라북도도립미술관   전라북도 완주군 구이면 모악산길 111-6
[사립] [ 미술관]                    보임쉔   전라남도 담양군 대전면 신룡길 79-3
[사립] [ 미술관]                    일현미술관   강원도 양양군 손양면 동호리 191-8
[대학] [ 미술관]                    극재미술관   대구광역시 남구 명덕로 104
[사립] [ 미술관]                    고은사진미술관   부산광역시 해운대구 해운대로 452번길 16 [48089]

추천 목록은 사용자 데이터를 기반으로 합니다.
```

## 4. 클래스 및 메서드 설명

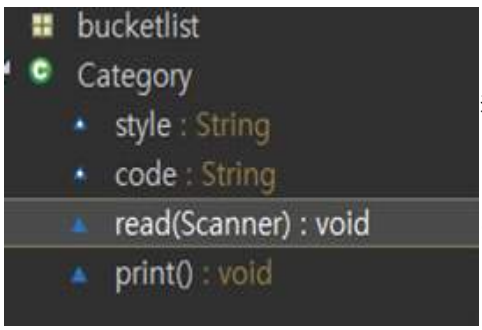
**[Main class]**: 프로그램 실행을 담당하는 메인 클래스




(1) getCategory
카테고리 데이터와 뮤지엄 데이터를 엮기 위해 만든 메서드입니다.

(2) openFile
데이터 파일을 읽기 위해 사용하는 메서드입니다.

**[Category class]**: 카테고리 데이터와 뮤지엄 데이터를 엮는 클래스

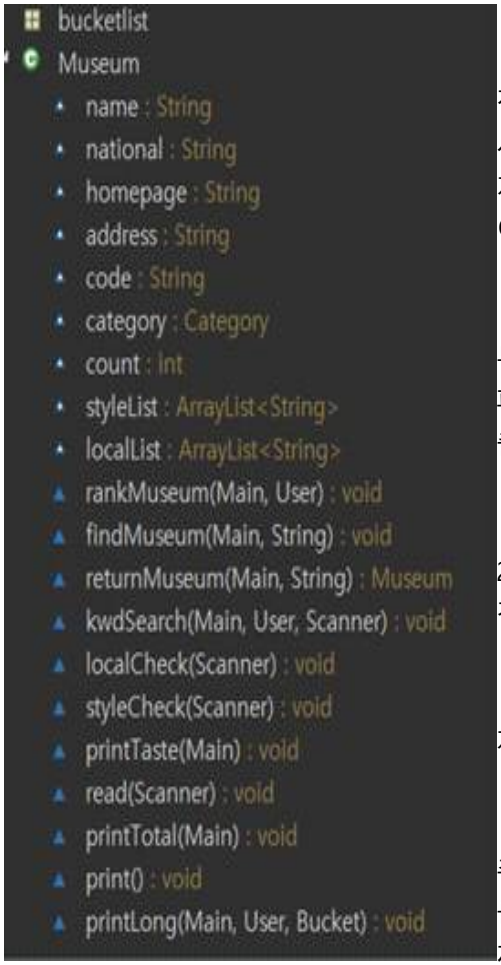



(1) read
카테고리 데이터를 읽는 메서드입니다.

(2) print
입력받은 카테고리 데이터를 출력하는 메서드입니다.

*카테고리 클래스는 뮤지엄 클래스에서만 쓰이며 나머지 4개의 클래스는 카테고리와 자기 자신 클래스를 제외한 나머지 클래스를 대부분 사용하는 구조를 취하고 있습니다.

**[Museum class]**: 뮤지엄 데이터의 입출력을 담당하는 클래스




(1) rankMuseum
전체 사용자들의 버킷리스트 데이터를 통계 내는 메서드입니다. 통계 기능의 경우 따로 클래스를 만들지 않고 관련 있는 클래스에 통계기능을 추가한 것이 특징입니다.

(2) findMuseum
뮤지엄 키워드를 매개변수로 받아서 전체 데이터 파일에서 키워드가 일치하는 경우 해당 뮤지엄 객체 출력해주는 메서드입니다.

(3) returnMuseum
2번 메서드와 비슷하지만 결과 값을 출력하지 않고 객체로 리턴해주는 메서드입니다.

(4) kwdSearch
자율 검색 기능입니다.

(5) localCheck, styleCheck, printTaste
추천 검색에 사용되는 메서드들입니다. 지역, 키워드를 받아 리스트에 저장하게 됩니다. printTaste는 전체 뮤지엄 데이터들 중에서 두 리스트에 전부 매치되는 데이터를 필터링하여 출력합니다.

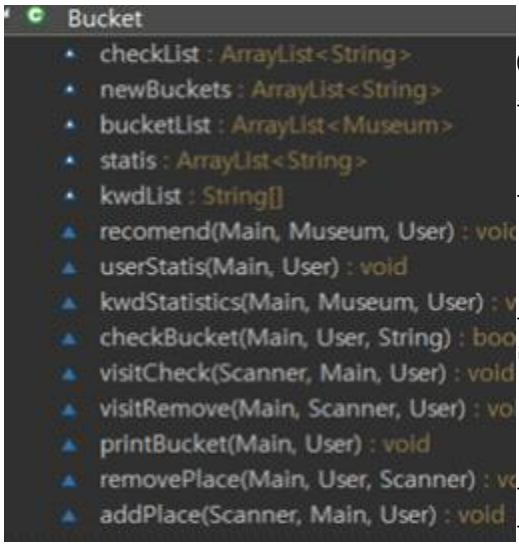
(6) read
Musuem 객체와 관련된 데이터를 읽는 메서드입니다.

(7) printTotal
모든 데이터를 형식에 맞춰 출력해주는 메서드입니다.

(8) print
객체를 형식에 맞춰 출력해주는 메서드입니다.

(9) printLong
버킷리스트 클래스에서 활용하는 데이터로 방문 데이터 파일에 데이터 존재하는 경우 방문 체크가 표시되어 출력되도록 하는 출력 메서드입니다.

**[Bucket class]**: 버킷리스트와 관련된 기능을 담당하는 클래스




(1) recommend
추천기능입니다. 유저 개개인의 검색, 버킷리스트 데이터와 전체 유저들의 버킷리스트 데이터를 지표로 추천합니다.

(2) userStatis
유저의 검색 기록 데이터를 열어서 빈도수 순으로 랭크하여 출력해주는 메서드입니다.

(3) kwdStatis
유저 개인의 버킷리스트 데이터를 오픈하여 지역, 국공립, 키워드 별로 분류해주는 데이터입니다

(4) checkBucket
user.__check.txt 파일(방문 체크 데이터 파일)을 오픈하고 임시 리스트에 데이터들을 추가합니다. 버킷리스트 출력시 임시 리스트와 비교하게 되는데, 개인의 버킷리스트 데이터를 출력할 때 유저의 버킷리스트 데이터 파일과 user___check.txt 데이터 파일 양쪽에 존재하는 경우 방문 체크로 표시되어 출력됩니다.

(5) visitCheck
user___check.txt 파일에 입력받은 데이터를 추가하는 기능을 가진 메서드입니다.

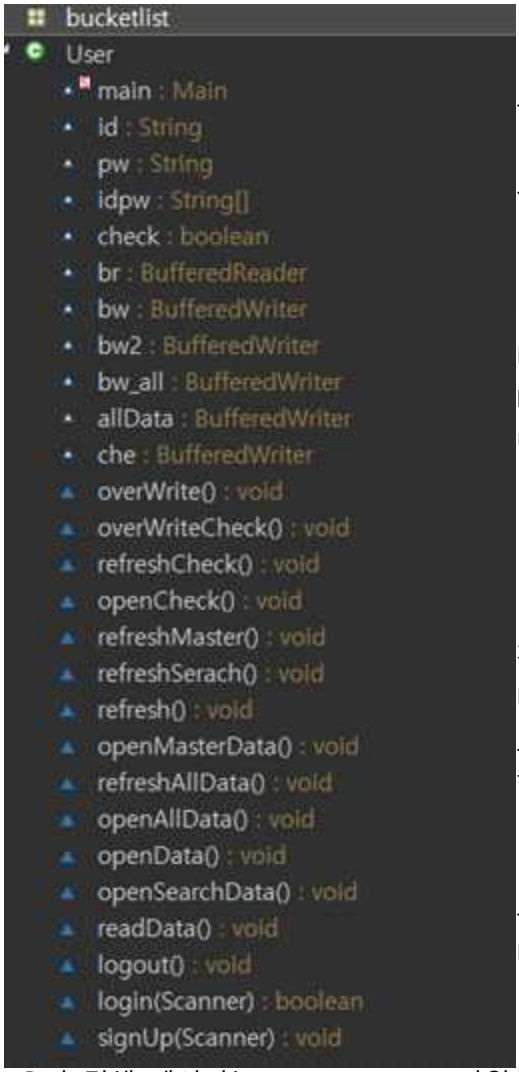
(6) removeCheck
방문 체크 표시한 기능을 삭제하는 기능으로 overwrite 방식을 채택했습니다. 지우고자 하는 데이터가 유저 데이터에 존재하는 경우 해당 데이터를 제외한 나머지 데이터를 리스트에 임시 저장하고 이 리스트를 가지고 파일을 overwrite 했습니다.

(7) printBucket
유저 개인의 버킷리스트 데이터 파일을 불러와 출력해주는 메서드입니다.

(8) removePlace
입력받은 데이터가 버킷리스트에 존재하면 해당 데이터를 삭제합니다. 만약 입력 받은 데이터가 버킷리스트에 존재하지 않는 경우 버킷리스트에 입력 받은 데이터가 없다는 안내 메시지가 출력됩니다.

(9) addBucket
버킷리스트에 데이터를 추가하는 메서드입니다. 만약 유저 버킷리스트 파일에 존재하거나 전체 뮤지엄 목록에 존재하지 않는 경우 해당 데이터가 이미 존재하거나 존재하지 않는 장소라는 경고 메시지가 출력됩니다.

**[User class]**: 로그인, 로그아웃, 회원가입 등 유저 데이터와 관련된 클래스




(1) overWrite, overWriteCheck
유저 버킷리스트 데이터 와 유저 방문 체크 데이터 삭제할 때 데이터 파일을 over writing 하는 메서드입니다.

(2) refreshMaster, refreshSearch, refreshAllData
데이터 파일에 쓰기를 하고 난 뒤에 파일을 닫고 다시 열지 않으면 정상적으로 입력이 되지 않기 때문에 파일을 닫고 다시 닫은 파일을 열기 위해 만든 리프레쉬 메서드들입니다

(3) openCheck, openAllData, openData, openSearchData

전체 유저 버킷리스트 데이터, 유저 개인의 버킷리스트, 검색, 방문체크 데이터 파일을 오픈하는 메서드입니다. 데이터를 이어서 쓰기 형식으로 오픈하는 것이 특징입니다.

(4) readData
유저 개인의 데이터를 읽기 형식으로 읽어들이는 메서드입니다.

*유저 검색 데이터는 user_search.txt 파일에서 따로 관리가 됩니다.

(5) logout
로그인한 유저의 개인 데이터 파일을 클로즈 시키는 메서드입니다.

(6) login
ID와 PW를 입력받아 두 데이터와 모두 일치하는 데이터가 유저 데이터베이스에 있는 경우 해당하는 ID의 데이터를 오픈시키는 로그인 기능의 메서드입니다.

(7) singUp
회원가입 기능입니다. ID와 PW를 입력받아 유저 데이터베이스 ID가 존재하지 않으면서 비밀번호가 5자리 이상일 경우 회원가입이 완료됩니다. 회원가입에 성공하면 유저의 개인 데이터 파일이 생성됩니다.

**4. 본인 평가 & 후기**

입출력, 자율 검색, 추천 검색, 통계 기능, 로그인 기능 등 프로젝트 설계 당시 구현하고자 했던 기능을 모두 구현하는 데에 성공했으며 교수님께서 피드백 주신 내용을 참고하여 데이터 삭제시 OverWrite 방식으로 구현했고, 추천 기능에서 다른 유저의 버킷리스트 데이터를 참고하는 방식을 어떻게 구현할까 고민하던 중 유저가 버킷리스트에 데이터를 추가하는 경우 유저 데이터 파일 뿐만 아니라 전체 유저 데이터 파일을 하나 만들어서 해당 파일에 유저 버킷리스트 데이터를 추가하는 방식을 고안해냈습니다. 그리고 이 데이터를 기반으로 뮤지엄 랭킹까지 구현할 수 있었습니다. 결과적으로 유저들의 입력, 버킷리스트 데이터를 수집하여 추천기능의 일부로 활용했습니다.

추천 기능을 설계하는 과정에서 프로그램 설계시 데이터를 어떤 구조로 수집하고 저장할 것인가에 대하여 꽤 오랜 시간 생각을 해야만 했고 시행착오를 겪으며 기능을 구현하다보니 어떤 방식으로 유저 데이터를 수집하는 방식이 재사용하기 좋은지, 유저가 프로그램을 사용하며 입력 값을 줄 때 어느 타이밍에 유저 데이터 파일을 오픈시켜서 데이터를 입력시키고 닫아야 하는지 등을 생각하는 과정을 거치며 클래스 파일 단위를 넘어서 규모는 작지만 프로그램이 작동하는 원리에 대해서 많은 생각을 하며 프로그램을 설계했고 이 부분이 가장 잘한 점 같습니다. 아쉬운 점으로는 최적화까지 하지 못한 점이 아쉽습니다.

1차 피드백에서 데이터가 심심하니 유저 데이터를 잘 활용해야한다는 피드백을 해주셨습니다. 따라서 해당 피드백 내용에 집중했습니다. 제가 느끼기에도 데이터 자체가 심심했고 다층적인 관계로 엮어서 재미있는 결과물을 내기에는 쉽지 않은 데이터임을 느꼈습니다. 따라서 유저 데이터를 활용하는 로그인 기능과 통계, 추천 기능을 설계하고 구현하는 데에 집중했습니다. 수많은 시행착오를 겪으면서 로그인 기능을 비롯하여 유저 데이터를 불러와서 데이터를 입력하고 삭제하는 기능을 구현하면서 알고리즘과 자료구조는 물론 프로그램을 만드는 데에 있어서 막연히 메서드들을 만들고 붙이는 게 아니라 처음부터 꼼꼼한 설계가 필요하다는 생각이 들었습니다. 이러한 과정을 거치다보니 자연스럽게 객체에 대한 이해도 증가하게 되었습니다. 특히 메서드에 값을 주고 특정 값을 반환해야할 때 객체 형태로 반환하는 형태가 매우 효율적이라고 느꼈습니다.

프로그램에서 뮤지엄 데이터의 빈도수를 기준으로 정렬하는 구조를 많이 사용했는데 컬렉션 라이브러리를 찾아 옵션 등을 찾고, 어떻게 하면 효율적으로 중복 데이터를 삭제하고 통계 기능을 낼 수 있을까 고민하다 보니 일정한 코드 패턴이 생겼고 이 코드 패턴을 적극 활용하여 메서드 소스 코드에 활용했습니다. 이후에 제가 사용한 이 코드 패턴을 다시 보니 불필요한 코드가 있어서 2차 수정을 해보기며 하며 더 최적화된 알고리즘은 없을까 고민도 해볼 수 있었습니다.

이번 프로젝트 경험을 통해서 프로그래밍에 대한 이해는 물론 데이터를 다루는 방법에 대해서 많은 생각을 할 수가 있었습니다. 앞으로의 배움에 있어 튼튼한 기반이 될 시간이었습니다. 많은 시간을 들였지만 많이 헤매기도 해서 기능이 화려하지는 않습니다. 하지만 기본적인 기능들의 완성도 및 유연성을 높이려고 했습니다. 프로그램을 더 확장해서 커뮤니티

기능까지 구현한다면 유저 데이터를 수집할 수 있는 괜찮은 프로그램으로 나아갈 수 있을 것 같습니다.

한 학기 동안 많은 것을 배울 수 있었습니다. 감사합니다.