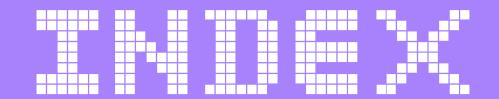


Git Hub 주소

프로젝트 소스: https://github.com/kim798546/gameland

포트폴리오: https://github.com/kim798546/gameland-portfolio



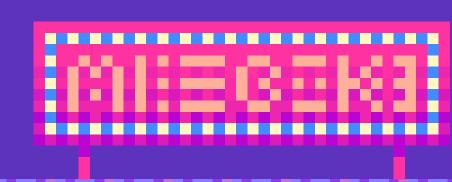








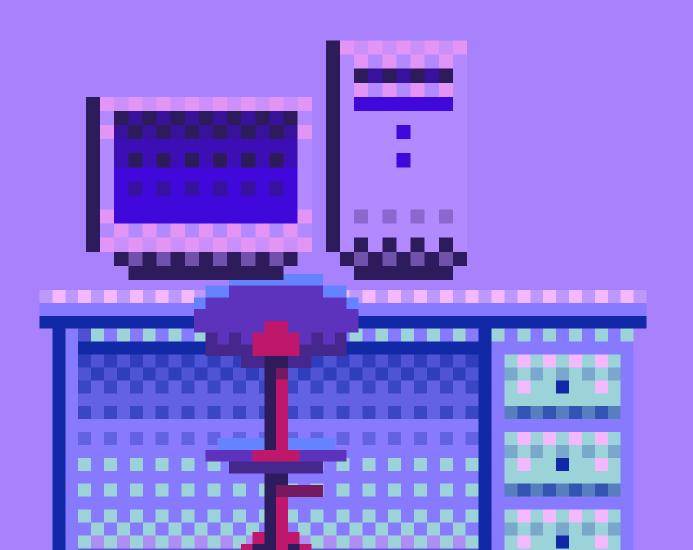


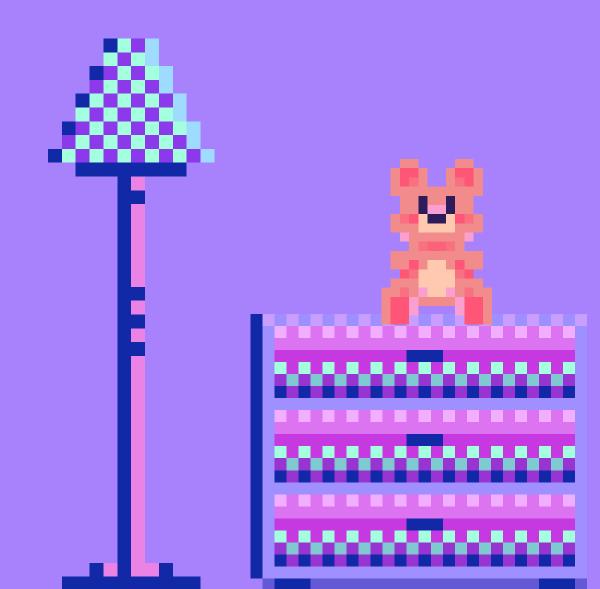




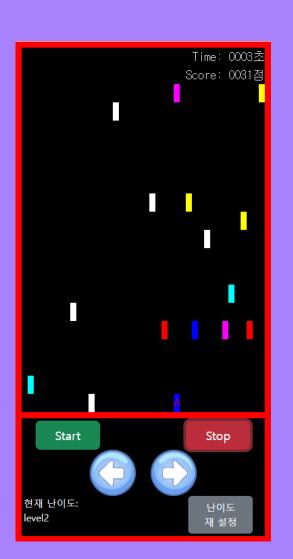
보유한 기술을 기반으로

<u>사용자가 직접 플레이 가능한</u> 게임 홈페이지 구현



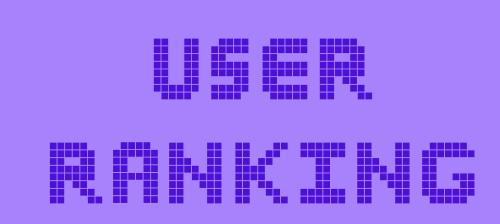








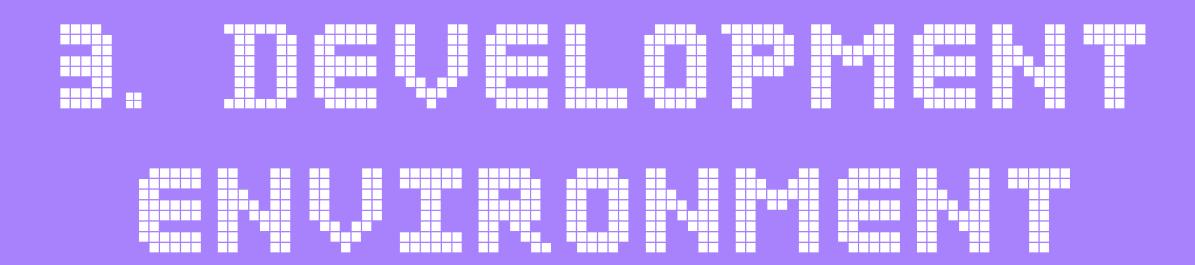






미니 게임 플레이 기능

게임 기록 랭킹화











eclipse(2022-12)

Java(jdk17)

Oracle 21c XE

Spring Boot(3.1.0)





Plug in

- STS(Spring Tool Suite)
- Tern, Node.js

웹 페이지 구현

- html5
- css3 & bootstrap5
- ECMAscript6



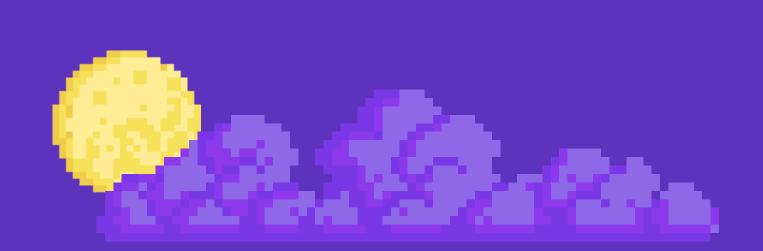
라이브러리 및 프레임 워크

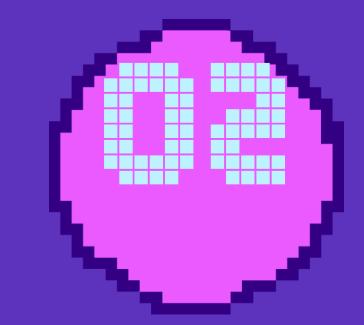
- Gradle(Build Tool)
- ORM: Mybatis
- Spring Security(보안 관련 처리)
- 암호화: BCryptPasswordEncoder
- Spring Web(Tomcat 사용)
- Lombok(@data와 같은 Annotation)
- Validation(사용자 입력 데이터 검증)
- Page Helper(랭킹 페이지 구현)
- JSP





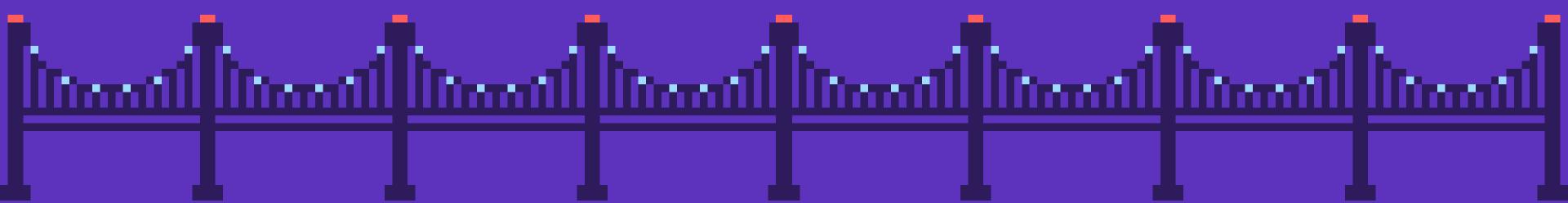
- Scott 유저 생성
- SQL Developer 사용
- =>테이블 생성 및 테스트 진행

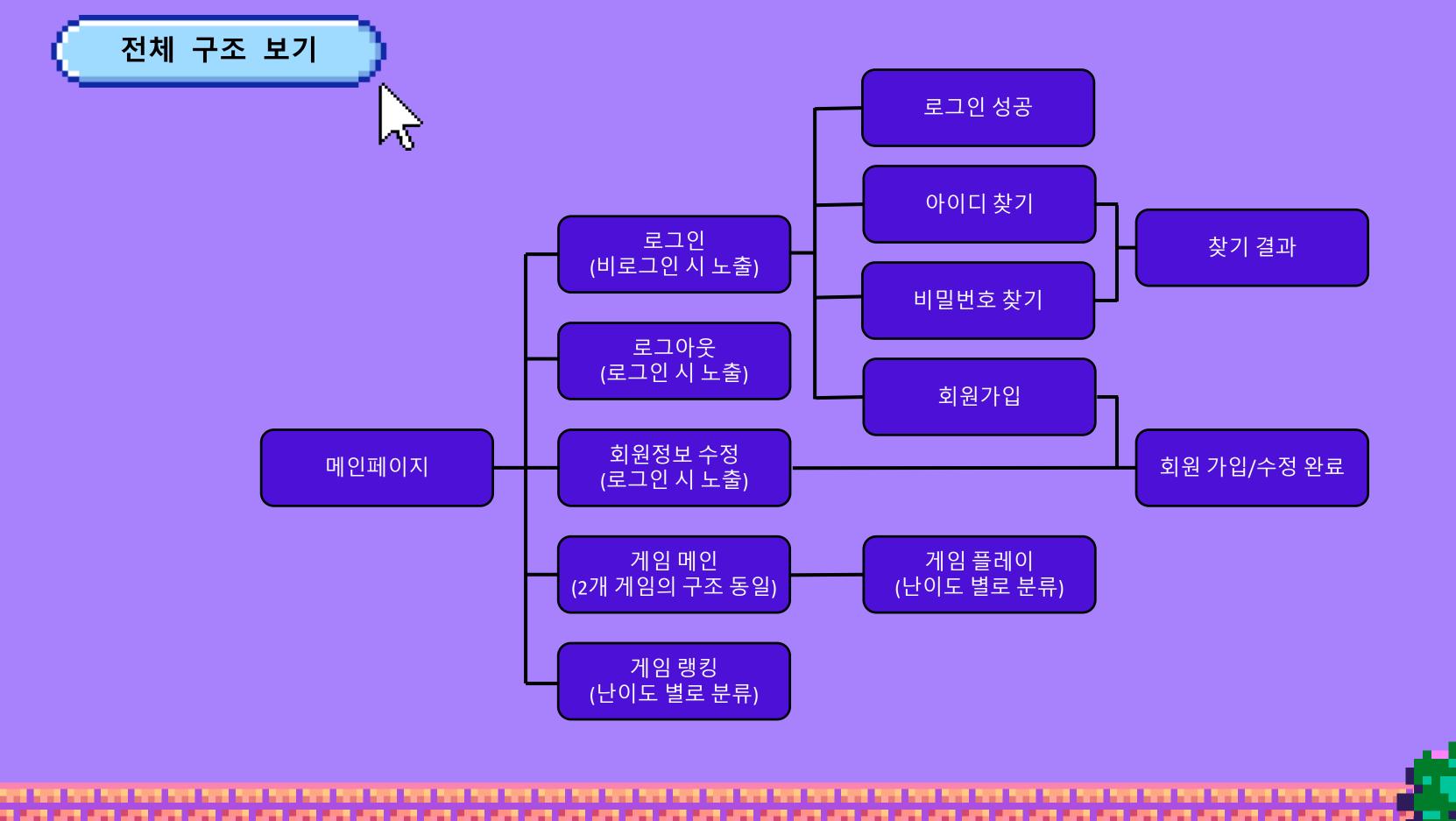




프로젝트 구성도

Let's see a project!





MVC MODEL



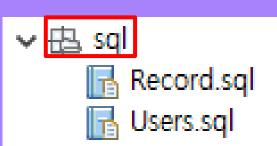


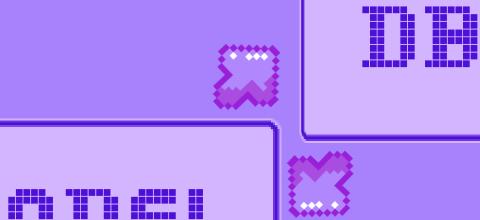


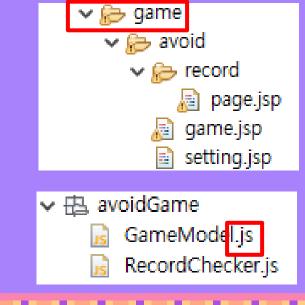
- AvoidGameController.java
- RecordCreateController.java
- RecordPageController.java
- 🗸 🌐 mapper
 - RecordMapper.java
- 🗸 🌐 model
 - > Jì Record.java

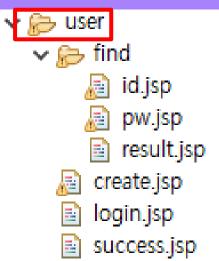




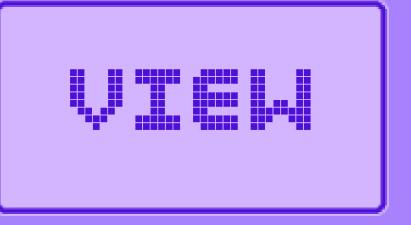






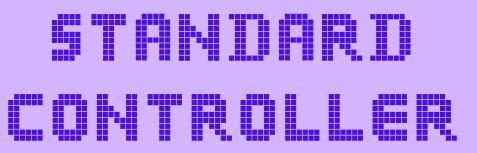


update.jsp





MVC MODEL



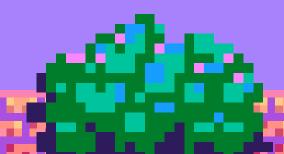


- controller
 - AjaxCreateController.java
 - CreateController.java
 - PageableController.java
 - SuccessController.java
 - > If UpdateController.java

```
13 //Ajax로 받은 데이터 처리 관련 interface
14 public interface AjaxCreateController<T,K,V> {
15
16 @GetMapping("/create")
17 void create(Model model, HttpServletRequest request);
18
19 @PostMapping("/create")
20 ResponseEntity<T> create(@RequestBody Map<K, V> requestBody);
21
22 }
```

```
12 // 계정 생성 관련 interface
13 public interface CreateController<DTO> {
14
15의 @GetMapping("/create")
16 void create(Model model, HttpServletRequest request);
17
18의 @PostMapping("/create")
19 String create(@Valid DTO dto, BindingResult binding, Model model, HttpServletRequest request, RedirectAttributes attr);
20
21 }
```

Standard Controller interface를 구현 => 표준에 맞추어 구현 클래스를 생성함



DATA BASE



USER_NAME VARCHAR2 (10 CHAR)

SCORE NUMBER
ELAPSE_TIME NUMBER
GAME LEVEL NUMBER

RECORD TABLE 블록 피하기 게임의 기록들을 저장

SCOTT.RECORD2

USER_NAME VARCHAR2 (10 CHAR)

SCORE NUMBER
GAME_LEVEL NUMBER

RECORD2 TABLE 스네이크 게임의 기록들을 저장

SCOTT.USERS

* ID VARCHAR2 (20 BYTE)

* PASSWORD VARCHAR2 (100 CHAR)

ROLE VARCHAR2 (20 CHAR)

NAME VARCHAR2 (20 BYTE)

PHONE_NUMBER CHAR (13 CHAR)

■ USERS_ID_PK (ID)

USERS_ID_PK (ID)

USERS TABLE 사용자 계정과 관련된 정보들을 저장






```
//run 메소드 반복 수행
timerChecker[this.blockNumber] = this.timer = setTimeout(() => {
    this.run(object, timerChecker, target, width, height, blockMap,
}, this.cycle);
```

setTimeout으로 run 메소드 반복 호출


```
//반복적으로 수행되면서 캐릭터와 block들을 조작하는 메소드
run object, timerChecker, target, width, height, blockMap, columnChecker, elapseTimeId, recordChecker, setFinalScore, finalScoreTag) {
                             이전 위치의 HTML 태그의 display 속성을 'none'으로 하여 숨김
   this.hide(object);
   this.line++;
  if(blockMap[this.line-1][this.column-1]==null){
                                                              위치 정보는
      blockMap[this.line-2][this.column-1]=null; /
      blockMap[this.line-1][this.column-1]=this;
                                                         20X40의 2차원 배열인
  } else { //캐릭터와 부딪힌 경우 게임 종료
                                                            blockMap에 저장
      for(let i=0; i<timerChecker.length; i++){</pre>
          clearTimeout(timerChecker[i]);
      clearInterval(elapseTimeId):
                                                         이동한 위치에 캐릭터가
      this.gameEnd(recordChecker)
                                                         존재할 경우 게임 종료
      return;
   object.style.left = width*(this.column-1)
   object.stvle.top = height*(
                            현재 위치의 HTML 태그의 display 속성을 'inline'으로 하여 화면에 출력
   this.show(object);
```

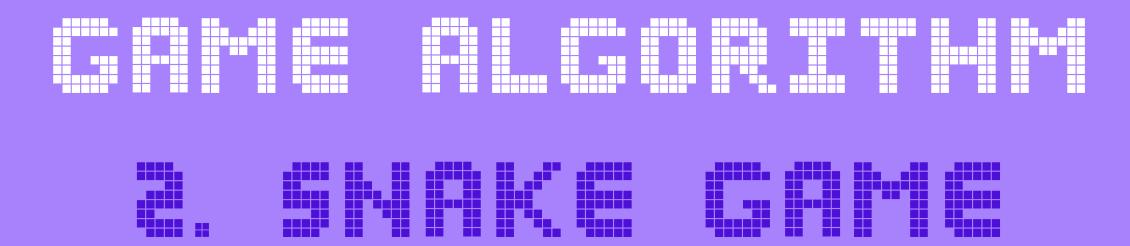


```
//isStart가 true일 때만 run 메소드를 반복 수행
if(this.isStart){
    objects[objects.length-1].timerId = setTimeout(()=>{
        this.run(objectTags, objects, target, t, width, height, blockMap, cycle)
    }, cycle);
}
```

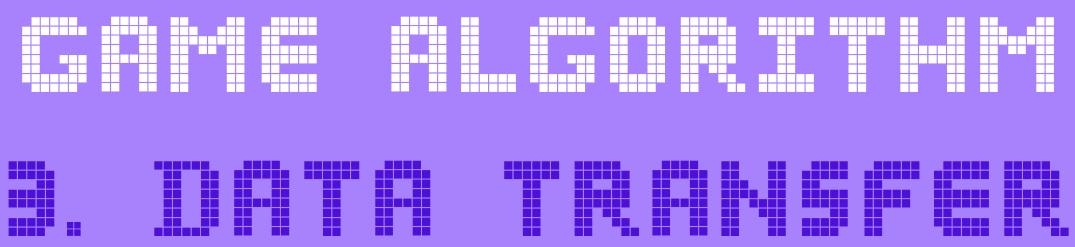
setTimeout으로 run 메소드 반복 호출



```
//반복적으로 수행되면서 캐릭터와 target을 조작하는 메소드
run(objectTags, objects, target, t, width, height, blockMap, cycle) {
   //Game객체들의 line과 column을 순서대로 저장(moveOthers 메소드에 대입하기 위함)
   let previousLine = [];
   let previousColumn = [];
   for(let e of objects){
       previousLine.push(e.line);
       previousColumn.push(e.column);
                                                   캐릭터의 맨 앞은 방향에 따라 1칸 이동,
   //objects배열의 첫 번째 객체인 head를 1칸 이동
   objects[0].moveHead(objectTags[0], objects, target, t, width, height, blockMap);
   //길이가 2이상이면 head가 아닌 나머지 객체들을 자신의 바로 앞의 line, d
                                                         자신의 바로 앞의 블록의 위치로 이동
   if(objects.length > 1)
       for(let i=1; i<objects.length; i++){</pre>
           objects[i].moveOthers(objectTags[i], previousLine[i-1], previousColumn[i-1], width, height, blockMap);
```



Target에 부딪혔을 때 새로운 블록이 생성되는 방식



int insertRecord(@Param("record") Record record);

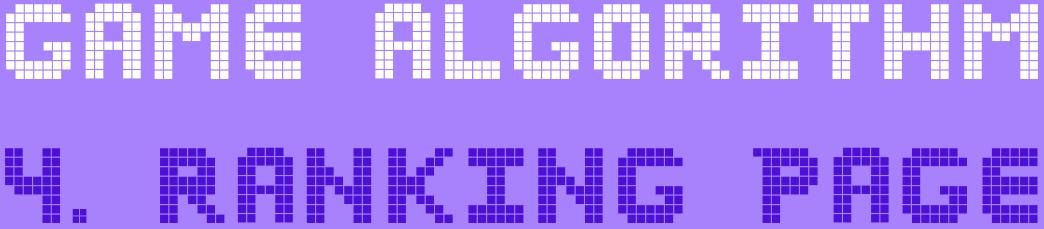
```
//게임이 종료되면 실행되는 메소드
gameEnd(recordChecker) {
   //게임이 종료되었을 때 Start버튼과 Stop버튼 누르지 못하게 막음
   document.querySelector("#btnStart").disabled = "true";
   document.querySelector("#btnStop").disabled = "true";
   //게임 종료 관련 팝업 출력
   const popup = document.querySelector('.popup');
   popup.style.display = 'inline';
   //username, level, score, elapseTime에 랭킹 관련 기록들을 담아서 서버로 전송
   const url = '/game/avoid/record/create';
   const data = {
       userName: recordChecker.userName,
      level: recordChecker.level,
       score: recordChecker.score-1, //부딪힐 때도 점수가 1점 추가되기 때문에 -1(맵핑처리)
       elapseTime: record
                          게임 종료 시 로그인 한 유저의 기록이
   };
                                    Ajax 방식으로 전송
   const options = {
       method: 'POST',
                                     (fetch 메소드 사용)
       headers: {
           'Content-Type': application/js
       body: JSON.stringify(data)
   };
   //로그인해야 정보가 서버로 전송되도록 조건 처리
   if(recordChecker.userName != "anonymousUser")
      fetch(url, options)
          .then(response => console.log(response))
           .catch(error => {
              console.log(error)
          });
```

```
@Override
public ResponseEntity<String> create(Map<String, String> requestBody) {
    var userName = requestBody.get("userName");
    var level = Integer.parseInt(requestBody.get("level"));
    var score = Integer.parseInt(requestBody.get("score"));
    var elapseTime = Integer.parseInt(requestBody.get("elapseTime"));
    var record = Record.builder()
                      .userName(userName)
                      .score(score)
                                                  Model(Record)에 유저의 기록을 담음
                       .elapseTime(elapseTime)
                      .gameLevel(level)
                      .build();
    //조건처리: 각 레벨 당 데이터를 최대 100개로 유지(랭킹 순).
    var minRecord = mapper.selectLastRowOfxLevel(level);
    var cnt = mapper.countAllOfxLevel(level);
    if(cnt<100)
                                               Mapper로 DB에 데이터를 삽입
        mapper.insertRecord(record);
    else {
       if(ninRecord.getScore()<score | minRecord.getScore()==score && minRecord.getElapseTime()<elapseTime) {</pre>
           mapper.deleteMinRecordOfxLevel(level);
            mapper.insertRecord(record);
    return ResponseEntity.ok("Data created successfully");
@Insert("""
       insert into record
       (user_name, score, elapse_time, game_level)
       values
        (#{record.userName}, #{record.score}, #{record.elapseTime}, #{record.gameLevel})
```



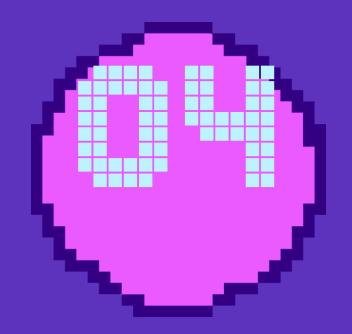
```
- Avoid Game 랭킹 테이블 제거
2 DROP TABLE RECORD;
4 -- Avoid Game 랭킹 테이블 생성
5 CREATE TABLE RECORD(
      USER_NAME VARCHAR2(10 CHAR),
      SCORE
                  NUMBER,
      ELAPSE_TIME NUMBER,
8
      GAME LEVEL NUMBER
10);
11
12 -- Snake Game 랭킹 테이블 제거
13 DROP TABLE RECORD2;
14
15 -- Snake Game 랭킹 테이블 생성
16 CREATE TABLE RECORD2(
      USER_NAME VARCHAR2(10 CHAR),
17
      SCORE
18
                  NUMBER,
19
      GAME LEVEL NUMBER
20 );
21
22 COMMIT;
```

```
1 DROP TABLE USERS;
 2
 3 CREATE TABLE USERS (
 4
                     VARCHAR2(20) CONSTRAINT USERS_ID_PK PRIMARY KEY,
        ID
 5
                     VARCHAR2(100 CHAR) NOT NULL,
        PASSWORD
        ROLE
                     VARCHAR2(20 CHAR),
        NAME
                     VARCHAR2(20 CHAR),
        PHONE_NUMBER CHAR(13 CHAR)
 8
9);
10
11 COMMIT;
```



```
1 package com.example.standard.controller;
3 mport org.springframework.ui.Model; ...
9 //페이지 처리 관련 interface
10 public interface PageableController {
       @GetMapping("/page/{pageNum}/{pageSize}")
       String page(httpservietkequest request, @rathVariable("pageNum") int pageNum, @PathVariable("pageSize") int pageSize, Model model);
13
14 }
15
16 //Avoid Game 랭킹 페이지 처리 Controller
17 @Controller
18 @RequestMapping("/game/avoid/record")
19 public class RecordPageController implements PageableController{
20
21⊖
       @Autowired
22
       RecordMapper mapper;
                                                                                                     @Select("""
23
                                       url 정보로 페이지를 생성
                                                                                                                 select *
24⊖
       @Autowired
                                                                                                                   from record
25
       ObjectMapper json;
                                                                                                                  where game_level = #{gameLevel}
26
                                                                                                                  order by score desc,
27⊝
       @Override
                                                                                                                           elapse_time desc
28
       public String page(HttpServletRequest request, int pageNum, int pageSize, Model model) {
           var level = Integer.parseInt(request.getParameter("level"));
29
                                                                                                     Page<Record> selectPageByGameLevel(@Param("gameLevel") int gameLevel);
           model.addAttribute("level", level);
30
31
32
           PageHelper.startPage(pageNum, pageSize):
           var list = mapper.selectPageByGameLevel(level)
33
           var paging = PageInfo.of(list, 10);
34
                                                               Mapper로 조회한 객체들을 list에 넣음
35
           model.addAttribute("list", list);
           model.addAttribute("paging", paging);
```

return "game/avoid/record/page";





결과물 소개



Let's test a project!

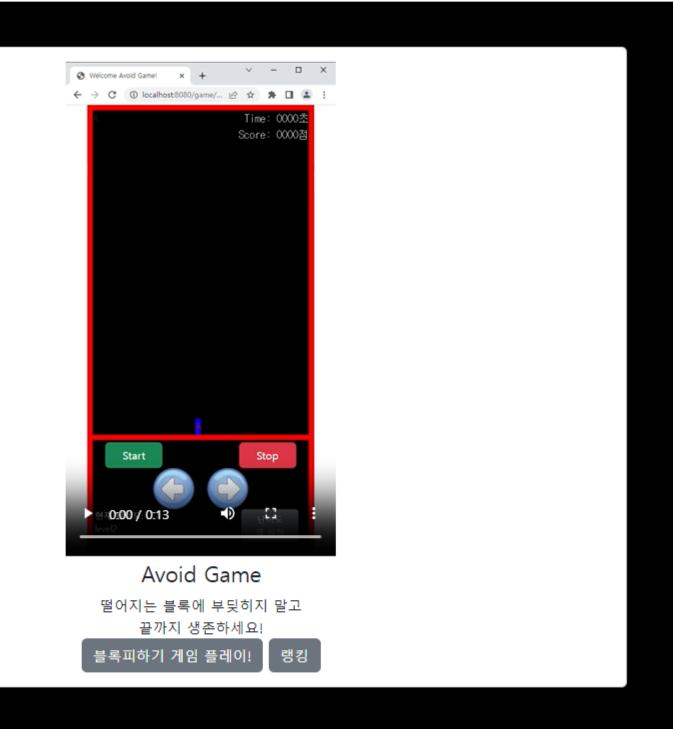


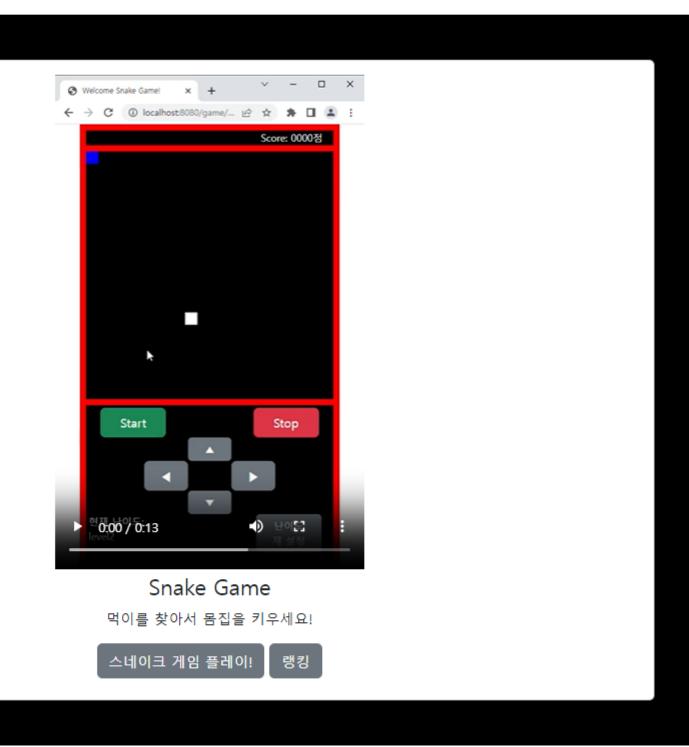
초기화면

Welcome Game Land!

*랭킹 등록을 원하시면 로그인 후 플레이 해주세요!

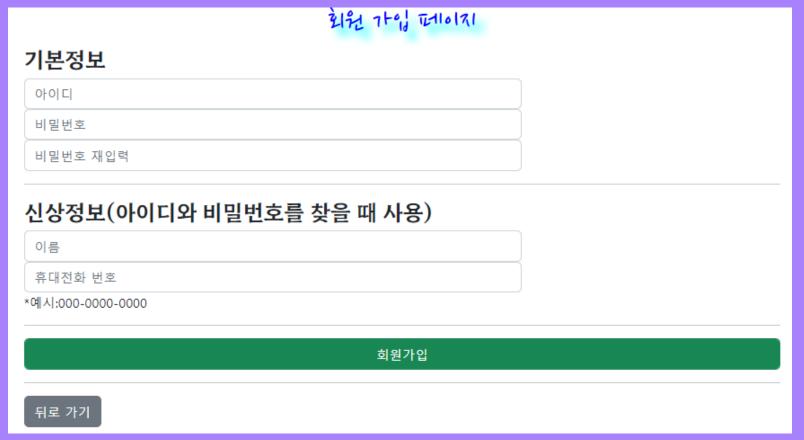
로그인





계정 관련 페이지 1









계정관련페이지글

아이디 찾기

id 조회 결과입니다.

당신의 id는 java입니다.

로그인 페이지로 이동

비밀번호 찾기

임시 비밀번호를 발급해드렸습니다.

임시 비밀번호는 e75a4792 입니다. 로그인 하신 후 꼭 비밀번호를 변경해주시기 바랍니다.

로그인 페이지로 이동



회원 정보 수정이 완료되었습니다.

메인 화면으로

게임소개및셋팅페이지

Block 피하기 게임 세계에 오신 것을 환영합니다.



간단설명:

하늘에서 떨어지는 수 많은 블록들을 방향키를 이용해 피하시면 됩니다.

- 1. 가운데에 A문자가 적혀있는 블록이 캐릭터 입니다.
- 2. 화살표 버튼이나 키보드의 방향키로 캐릭터를 조작합니다.
- 3. 하늘에서 수 많은 블록들이 떨어집니다. 캐릭터와 부딪히면 게임종료.
- 4. 블록이 바닥까지 떨어졌을 때 점수가 1점 추가 됩니다.
- 5. 난이도는 3단계까지 있으며, 난이도에 따라 블록의 속도와 수가 달라집니다.
- 6. 1단계: 쉬움, 2단계: 보통, 3단계: 어려움
- 7. Start 버튼을 눌러 게임을 시작하고, Stop 버튼을 눌러 일시정지 할 수 있습니다.
- 8. 난이도 선택 후 플레이 해주세요!

플레이 할 난이도를 선택해주세요.







메인 화면으로

Snake 게임 세계에 오신 것을 환영합니다.



간단설명:

목표물을 향해 이동하여 캐릭터의 몸집을 키우시면 됩니다.

- 1. 파란색 블록이 캐릭터, 흰색 블록이 목표물입니다.
- 2. 게임이 시작되면 캐릭터는 일정한 속도로 이동합니다.
- 3. 화살표 버튼이나 키보드의 방향키로 캐릭터의 방향을 조절합니다.
- 4. 캐릭터가 목표물을 집어 삼키면 길이가 길어지고, 1점이 추가됩니다.
- 5. 캐릭터가 벽에 부딪히거나, 자기 자신과 만나면 게임 종료.
- 6. 난이도는 3단계까지 있으며, 난이도에 따라 캐릭터의 속도가 달라집니다.
- 7. 1단계: 쉬움, 2단계: 보통, 3단계: 어려움
- 8. Start 버튼을 눌러 게임을 시작하고, Stop 버튼을 눌러 일시정지 할 수 있습니다.
- 9. 난이도 선택 후 플레이 해주세요!

플레이 할 난이도를 선택해주세요.

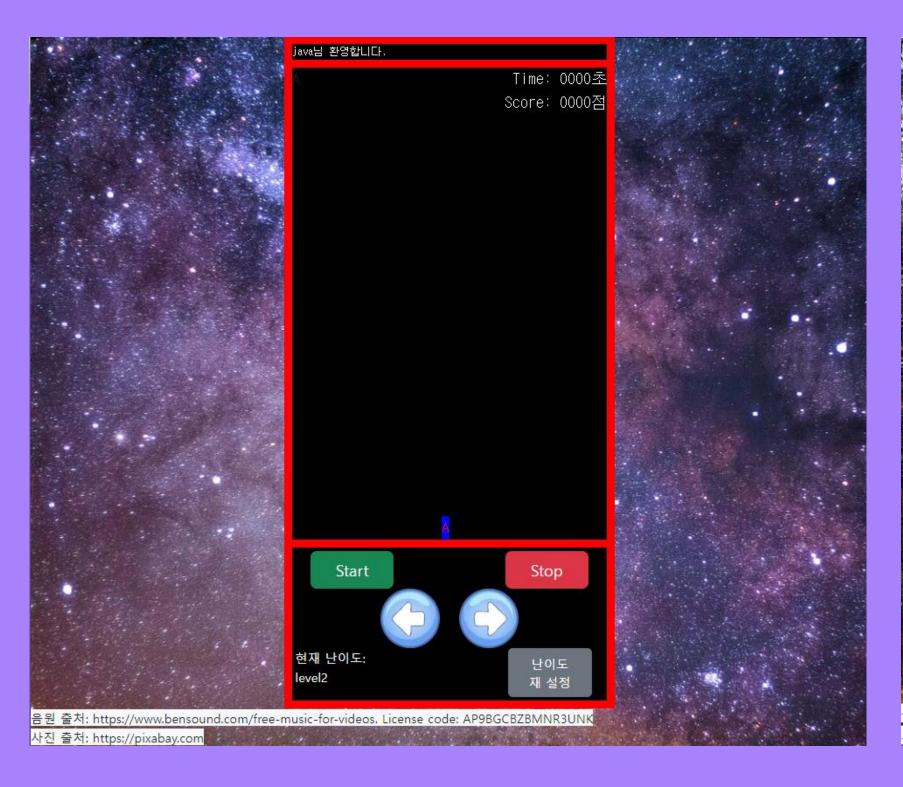


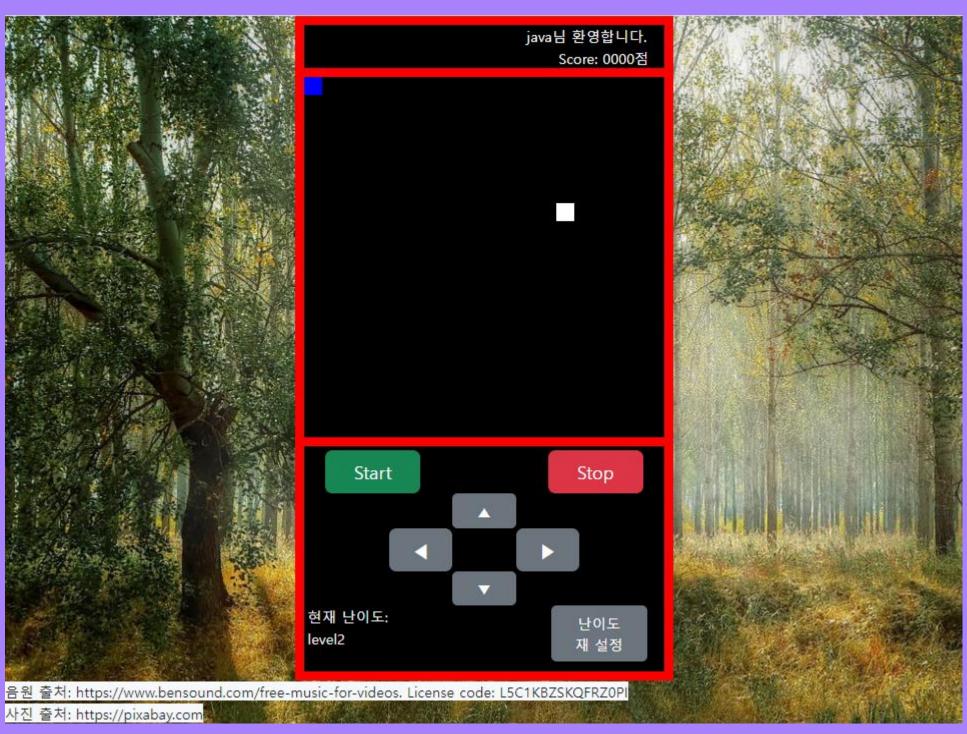




메인 화면으로

게임 플레이 페이지



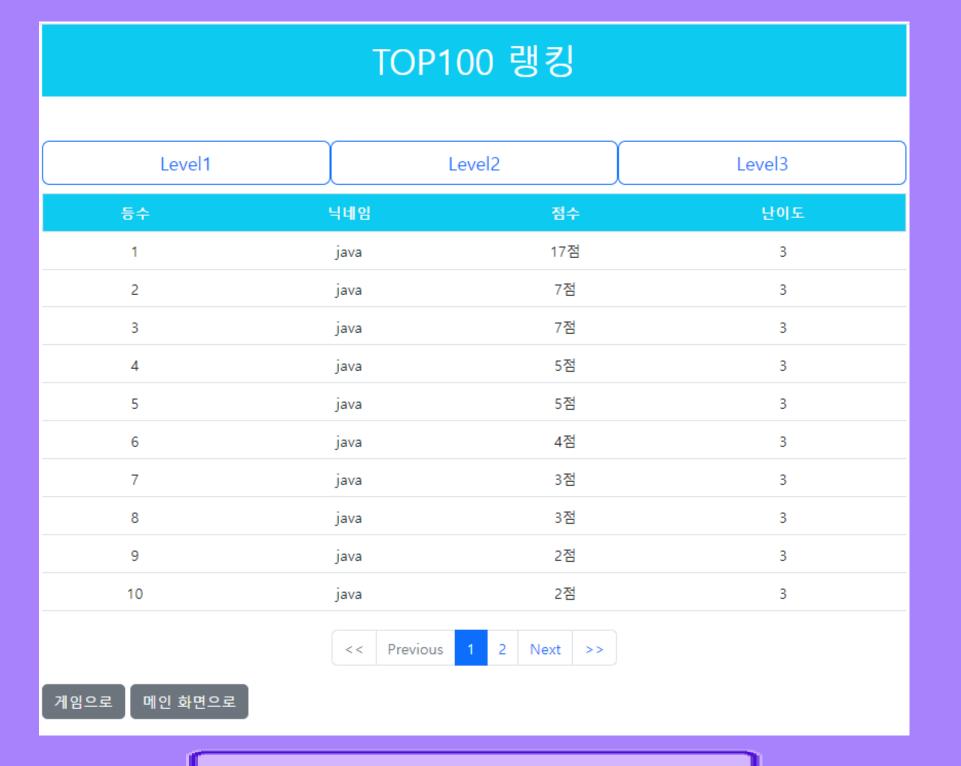


게임랭킹페이지

TOP100 랭킹

*순위 산정 기준: 점수 높은 순, 점수 같을 시 생존 시간 높은 순

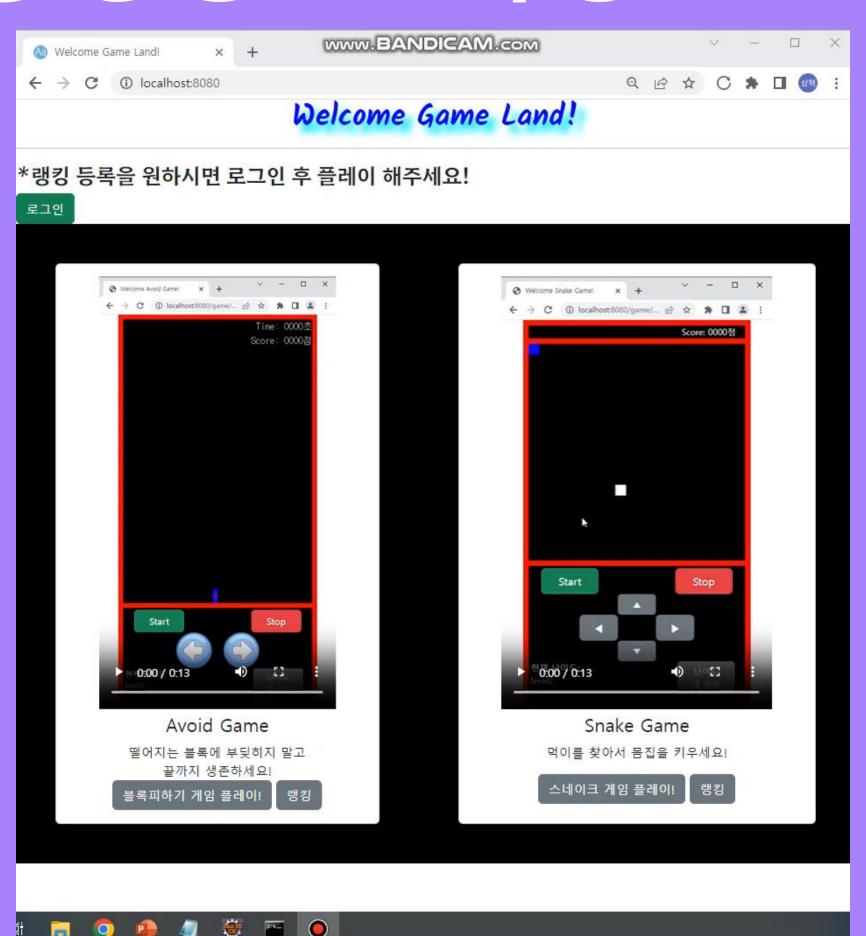
Level1		Level2		Level3	
등수	닉네임	점수	생존 시간	난이도	
1	java	251점	13초	3	
2	java	249점	12초	3	
3	java	209점	10초	3	
4	java	184점	7초	3	
5	java	69점	3초	3	
6	java	50점	2초	3	
7	java	34점	1초	3	
8	java	33점	1초	3	
9	java	31점	1초	3	
10	java	26점	1초	3	
게임으로 메이경	ig O Z	<< Previous 1 2	Next >>		



AVOID GAME



시연 동영상 1 계정 관련 조작



시연 동영상 검게임 플레이

