[ JAVA 프로그래밍 ]

[ 배열활용-주차관리 ]

담당 교수 : 지정희 교수님

학 번 : 202311264

이 름 : 김민홍

제출일 : 2023년 09월 27일

|  |  |
| --- | --- |
| 1 | **문제정의 및 분석** |

2차원 배열을 활용한 주차관리 프로그램을 작성하는 문제이다.

1. 반드시 main 메소드 안에 주차 공간에 관한 2차원 배열을 선언해야 한다, ROW와 COL의 크기는 마음대로 설정가능. 또한 static final을 통해 class안에서 배열의 행, 열을 상수로 선언해야 한다. 또한 전역으로 Scanner 객체를 생성하여 클래스 안에 모든 메소드에서 활용할 수 있도록 구현해준다.
2. 메인메뉴를 구성한다. 메뉴 선택창 위에는 특수문자를 이용한 주차공간 이용 가능여부를 표시해주고, 밑부분에서는 주차하기, 차량검색, 출차하기, 종료를 번호를 통해 선택할 수 있도록 구현한다. 주어진 번호 이외에 다른 번호를 입력했을 시에는 다시 입력해 달라고 출력하고 메뉴를 다시 불러온다. 1~3번 메뉴 수행후 다시 메인 메뉴가 출력될 수 있도록 구현해야 한다.
3. 주차하기 메뉴에서는 주차할 위치를 선택해주고(행과 열 순으로 띄어쓰기를 통해 구분하여 입력 받는다) 그 후 차량 번호를 입력 받고, 그 차량번호가 맞는지 사용자에게 다시 y/n를 통해 확인 할 수 있도록 구현해야 한다. 이후 정상적인 과정이 이루어지면 입력한 차량번호를 출력하면서 주차를 완료했다고 표시한다. 여기서 몇가지 입력 오류 처리를 해주어야 하는데 첫번째로 입력된 주차할 위치가 2차원 배열의 행렬 크기를 벗어나면 위치 번호를 확인해 달라고 다시 요청하고 메인 메뉴로 돌아간다. 또한 사용자가 입력한 위치에 다른 차량이 이미 주차되어 있다면 메인 메뉴로 돌아갈 수 있도록 한다. 차량번호 확인시 n을 입력하였을 때 메인 메뉴로 돌아갈 수 있도록 한다.
4. 차량검색 메뉴에서는 사용자로부터 차량번호를 입력받고, 배열 검사를 통해 해당 번호의 차량이 주차되어 있는 경우, 차량이 어디에 위치하는지를 출력해주는 메뉴이다. 차량 번호를 입력받을 때 주차관리 시스템에 등록되어 있지 않은 경우 차량이 존재하지 않는다고 출력해주고 메인 메뉴로 돌아가게 한다.
5. 출차하기 메뉴는 사용자로부터 차량번호를 입력 받고 해당 차량이 있는 경우 그 위치에서 차량을 빼고 해당 공간을 빈공간으로 출력하게 해야 한다. 만약 입력된 차량번호가 주차관리 시스템에 등록되어 있지 않으면 차량이 존재하지 않는다고 출력해주고 메인 메뉴로 돌아갈 수 있도록 구현한다.
6. 종료시에는 “시스템을 종료합니다” 출력해주고 scanner.close를 해준뒤 프로그램을 종료할 수 있도록 한다.

|  |  |
| --- | --- |
| 2 | **주요 소스코드 설명** |

String[][] parkingSpace = new String[*ROW*][*COL*];  
String[][] parkingCarNum = new String[*ROW*][*COL*];

parkingSpace는 주차 현황을 출력할 때 사용할 특수문자를 담는 2차원 배열이고 parkingCarNum은 따로 차량번호를 저장하기 위해 만든 2차원 배열이다.

for(int i=0;i<*ROW*;i++){  
 for(int j=0;j<*COL*;j++){  
 parkingSpace[i][j] = "♡";  
 }  
}  
for(int i=0;i<*ROW*;i++)  
 for(int j=0;j<*COL*;j++)  
 parkingCarNum[i][j] = "";

이중 for문을 이용해 각각의 배열을 빈공간으로 초기화 시켜주었다.

while(true){  
  
 *printParking*(parkingSpace);  
 System.*out*.println("1) 주차하기 2) 차량검색 3) 출차하기 4) 종료");  
 System.*out*.print("메뉴를 선택하세요 : ");  
 int menuNum = *scanner*.nextInt();  
 System.*out*.println();  
  
 switch (menuNum){  
 case 1->*parking*(parkingSpace, parkingCarNum);  
 case 2->*searching*(parkingCarNum);  
 case 3->*takingOff*(parkingSpace, parkingCarNum);  
 case 4->*sysExit*();  
 default -> System.*out*.println("메뉴 번호를 확인 후 다시 입력해 주세요.");  
 }  
 System.*out*.println();  
}

while문을 통해 사용자가 4를 입력해 sysExit메소드를 불러올 때 까지 프로그램이 종료되지 않게 구현해주었다. printParking이라는 주차현황을 보여주는 메소드를 만들었고 그 밑에서 메뉴를 출력해 사용자 입력을 switch문으로 처리하여 1~4메뉴를 선택하였을 때 각각의 메소드를 호출하여 메뉴를 수행할 수 있도록 구현하였다. 여기서 1~4이외의 다른 숫자값이 들어왔을 경우 default문을 통해 다시 메뉴를 출력하여 입력을 받을 수 있도록 구현하였다.

public static void printParking(String[][] parkingSpace){  
 System.*out*.println("\*\*\*\* 주차 현황 \*\*\*\*");  
 System.*out*.print(" ");  
 for(int i=1;i<=*COL*;i++){  
 System.*out*.printf("%3d", i);  
 }  
 System.*out*.println();  
 for(int i=0;i<*ROW*;i++){  
 System.*out*.printf("%2d", i+1);  
 for(String space : parkingSpace[i])  
 System.*out*.printf("%2s", space);  
 System.*out*.println();  
 }  
 System.*out*.println();  
}

printParking메소드를 살펴보면 열의 개수만큼 위에 숫자를 출력해주고 printf문을 통해 숫자사이의 간격을 맞춰줬다. 다음으로 한 행씩 for문을 이용해 주차현황을 출력할 수 있도록 구현하였다.

1번 메뉴인 주차하기를 살펴보자면

int setRow, setCol;  
String carNum;  
String check;

setRow, setCol : 사용자가 주차할 위치를 받는 변수

carNum : 차량번호를 입력 받는 문자열 변수

check : y/n을 입력 받는 문자열 변수

if(setRow<=0 || setRow>*ROW* || setCol <=0 || setCol>*COL*){  
 System.*out*.println("위치 번호를 확인해 주세요. 처음부터 다시 시작해 주세요.");  
 return;  
}  
if(parkingSpace[setRow-1][setCol-1].equals("♥")){  
 System.*out*.println("다른 차량이 주차되어 있습니다. 처음부터 다시 시작해 주세요.");  
 return;  
}

첫번째 if문은 사용자가 입력한 위치의 크기가 2차원배열을 넘는지 검사하는 코드이고 두번째 if문은 사용자가 입력한 위치가 다른 차량이 주차되어 있다는 것을 확인하는 코드이다. return을 통해 메소드를 끝내 메인 메뉴가 실행될 수 있도록 하였다.

System.*out*.print("차량 번호를 입력해 주세요(입력예 : 20가1234 ) : ");  
carNum = *scanner*.next();  
System.*out*.println("차량 번호 " + carNum+ " 맞습니까(y/n)? ");  
check = *scanner*.next();  
if(check.equals("n")){  
 System.*out*.println("처음부터 다시 진행해 주세요.");  
 return;  
}  
  
parkingSpace[setRow-1][setCol-1] = "♥";  
parkingCarNum[setRow-1][setCol-1] = carNum;  
System.*out*.println(carNum+"차량의 주차를 완료하였습니다.");

차량번호를 입력받고 이 차량번호가 맞는지 확인한 후 만약 사용자가 n을 입력했다면 return을 통해 메소드를 끝내고 그 외의 값이 들어오면 parkingSpace 배열에 사용자가 입력한 위치에 꽉찬 하트 표시를 하고 parkingCarNum에 차량번호를 넣어 준다.

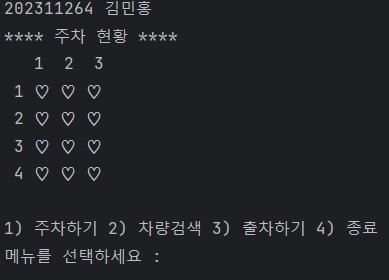
public static void searching(String[][] parkingCarNum){  
 String carNum;  
 System.*out*.println("\*\*\*\* 차량검색 \*\*\*\*");  
 System.*out*.print("차량 번호를 입력해 주세요 : ");  
 carNum = *scanner*.next();  
 for(int i=0;i<*ROW*;i++)  
 for(int j=0;j<*COL*;j++){  
 if(parkingCarNum[i][j].equals(carNum)){  
 System.*out*.println(carNum+"는 ("+(i+1)+", "+(j+1)+")에 위치합니다.");  
 return;  
 }  
 }  
 System.*out*.println("차량이 존재하지 않습니다. 차량번호 확인후 처음부터 다시 진행해 주세요.");  
}

차량검색에서는차량번호를 입력받고 이중for문을 통해 parkingCarNum에 등록된 것 있는지 확인하여 있다면 배열의 위치를 알려주고 return한다 만약 없다면 return이 실행되지 않아 존재하지 않는다고 출력해준다.

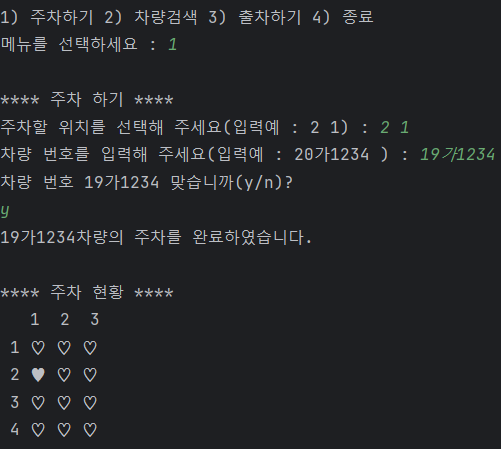
public static void takingOff(String[][] parkingSpace, String[][] parkingCarNum) {  
 String carNum;  
 System.*out*.println("\*\*\*\* 출차하기 \*\*\*\*");  
 System.*out*.print("차량 번호를 입력해 주세요 : ");  
 carNum = *scanner*.next();  
 for (int i = 0; i < *ROW*; i++)  
 for (int j = 0; j < *COL*; j++) {  
 if (parkingCarNum[i][j].equals(carNum)) {  
 parkingSpace[i][j] = "♡";  
 parkingCarNum[i][j] = "";  
 System.*out*.println(carNum+" 차량이 출차되었습니다. 안녕히 가세요.");  
 return;  
 }  
 }  
 System.*out*.println("차량이 존재하지 않습니다. 차량번호 확인후 처음부터 다시 진행해 주세요.");  
}

출차하기도 차량검색과 마찬가지로 사용자로부터 차량 번호를 입력받고 만약 parkingCarNum에 등록된 차량번호와 입력받은 것이 같다면 각각의 배열을 비워주고 출차되었다고 출력해주고 return을 통해 빠져나간다. 그런 차량번호를 찾지 못했다면 return되지 않아 차량이 존재하지 않는다고 출력해준다.

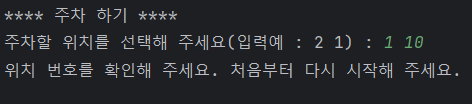
|  |  |
| --- | --- |
| 3 | **실행결과** |



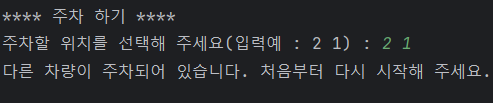
처음 실행화면



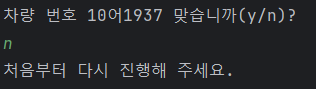
주차하기 정상작동 화면



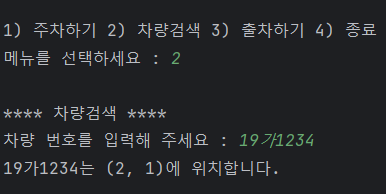
배열의 크기를 벗어났을 때



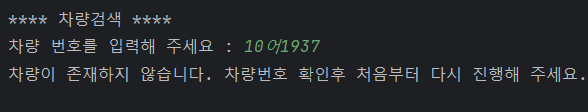
주차하려는 위치에 다른 차량이 이미 존재할 때 뜨는 메시지



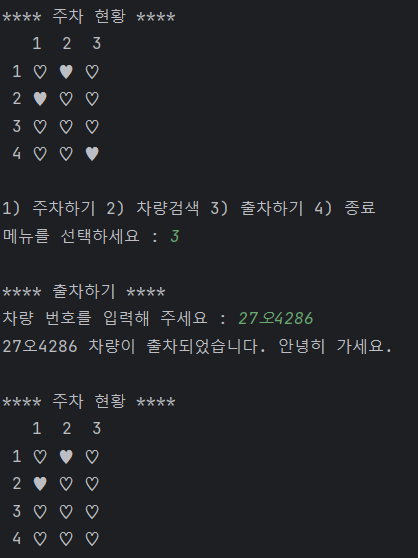
n을 입력했을 때 출력되는 메시지



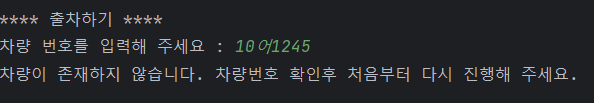
정상적인 차량검색 했을 때



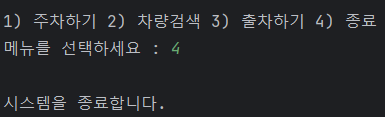
차량검색에서 차량이 존재하지 않을 때



정상적으로 차량이 출차 되었을 때(4,3자리에 빈공간이 됌)



주차된 차량번호가 없을 때.



종료시

|  |  |
| --- | --- |
| 4 | **느낀점 및 토의사항** |

우선 주차현황을 나타낼 때 출력되는 하트와 숫자의 칸을 맞추는 것이 힘들었고, 차량번호를 어떻게 저장할까 고민하다가 따로 배열을 하나 만들어주자 해서 또 하나의 배열을 하나 만들었다. 메인 메소드에서 보기 편하게 각각의 메뉴들을 메소드로 구현해주었는데 메소드에 이차원 배열을 넘겨주면 힙 영역의 주소값을 넘겨주는 것이기 때문에 메소드 안에서도 값이 변경된다.

|  |  |
| --- | --- |
| 5 | **과제 수행 완료 체크 리스트** |

<체크리스트>

**1. 배열 선언관련**

A. 문제에서 요구한 위치에 배열을 선언하고 그 배열을 사용하여 문제를 해결하였는가? **예**

**2. 메뉴 운영 관련**

A. 종료 메뉴를 선택하기 전까지 메뉴가 반복 수행되는가? **예**

B. 1번~3번 메뉴 수행 후 다시 메뉴를 출력하는가? **예**

C. 메뉴 선택 번호 이외의 숫자를 입력한 경우 예외 처리하였는가? **예**

D. 주차된 공간과 빈 공간을 문자를 이용하여 구분해서 예시와 같이 출력하고 있는가? **예**

**3. 주차하기 메뉴 관련**

A. 모든 정보를 정상적으로 입력한 경우 선택한 해당 위치에 차번호를 저장하는가? **예**

B. 이미 주차된 공간을 선택한 경우에 대한 처리를 수행하였는가? **예**

C. 주차 위치 입력할 때 잘 못된 위치의 숫자 입력시 예외처리 하였는가? **예**

D. 차량번호 확인에 대한 y/n에 대한 처리를 요구사항 대로 처리하였는가? **예**

**4. 차량검색 메뉴 관련**

A. 주차된 차량 번호를 올바르게 입력한 경우 주차 위치를 잘 출력하는가? **예**

B. 차량번호가 없는 경우 요구사항대로 처리하였는가? **예**

**5. 출차하기 메뉴 관련**

A. 주차된 차량 번호를 올바르게 입력한 경우 출차 메시지를 출력하는가? **예**

B. 출차된 위치의 주차공간은 다시 빈 공간으로 처리하였는가? **예**

C. 차량번호가 없는 경우 요구사항대로 처리하였는가? **예**

**6. 종료메뉴 선택시 프로그램이 종료되는가?** **예**

**7. 학번이름은 출력되어 있는가?** **예**