괄호 회전하기

{} [] () ⇒ 올바른 괄호

{[]}

하나의 스택에서 해야 개수 별 따로 올바른지 확인 가능.

o 12345
[](){}

12345 0 ⇒ 12345 + 0 ⇒ substring(x) + substring(0,x)

1. x번 회전시킨 s가 올바른 괄호 문자열인가
2. 스택에서 꺼낸 여는 괄호와 s의 문자가 닫는 괄호가 같은 종류의 괄호인가.

1. s의 길이가 홀수이면 무조건 올바르지 않음 ⇒ 0반환.

```
if(s.length()%2==1) return 0;
```

2. 올바른 괄호 문자열인지 확인하기 위해 스택, 올바른 괄호 문자열의 개수 값을 변수 선언

```
Stack<Character> stack = new Stack<>();
int count = 0;
```

3. x번 만큼 반복

```
for(int x=0; x< s.length(); x++){
```

4. x번 회전 시킨 str 생성

```
String str = s.substring(x) + s.substring(0, x);
```

boolean isValid(String s)

5. 올바른 괄호 문자열 판별 ; 새로운 함수로 작성

⇒ true이면 count++

```
if (isValid(str)) count++;
}
return count;
```

※ static boolean isValid(String s)

1. 이번 문자가 닫힌 괄호 혹은 아무런 문자가 아닌 괄호면 바로 false 반환

```
int len = s.length();
if (s.charAt(0) == ')' || s.charAt(0) == '}' || s.charAt(0) == ']' || charAt(len-1) == '(' || charAt(len-1) == '{' ||
    s.charAt(len-1) == '[') {
    return false;
}
```

2. s가 올바른지 판별

```
for(int i=0; i<len; i++){
    char ch = s.charAt(i);
```

3. 만약 여는 괄호인 경우 stack에 push

```
    if(ch == '(' || ch == '{' || ch == '['){
        stack.push(ch);
```

4. 닫는 괄호라면 stack에서 꺼내 같은 종류의 괄호인지 확인

```
    }else{
```

5. 이미 비어져 있다면 올바르지 않으므로 false 리턴

```
        if(stack.isEmpty()) return false;

        char open = stack.pop();
        if((open == '(' && ch == ')') || (open == '{' && ch == '}') || (open == '[' && ch == ']')){
            return false;
        }
    }
}
```

6. stack에 있던 괄호와 ch가 올바른 종류의 괄호가 아니면 false

7. stack이 비어있으면 올바른 괄호 문자열이기에 true 반환

```
return stack.isEmpty();
```