

기말 과제

게임 - 맷 속의 수 맞추기



소프트웨어 프로젝트
4분기

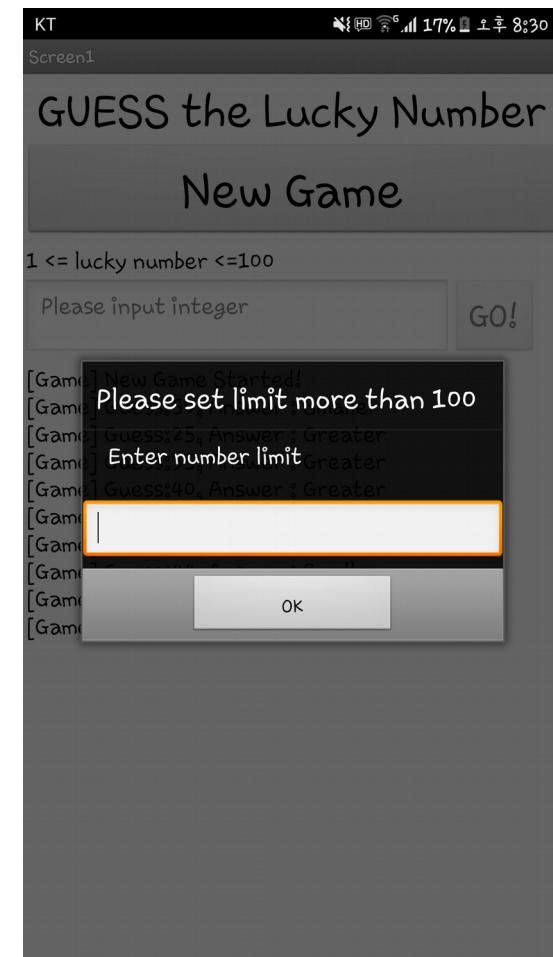
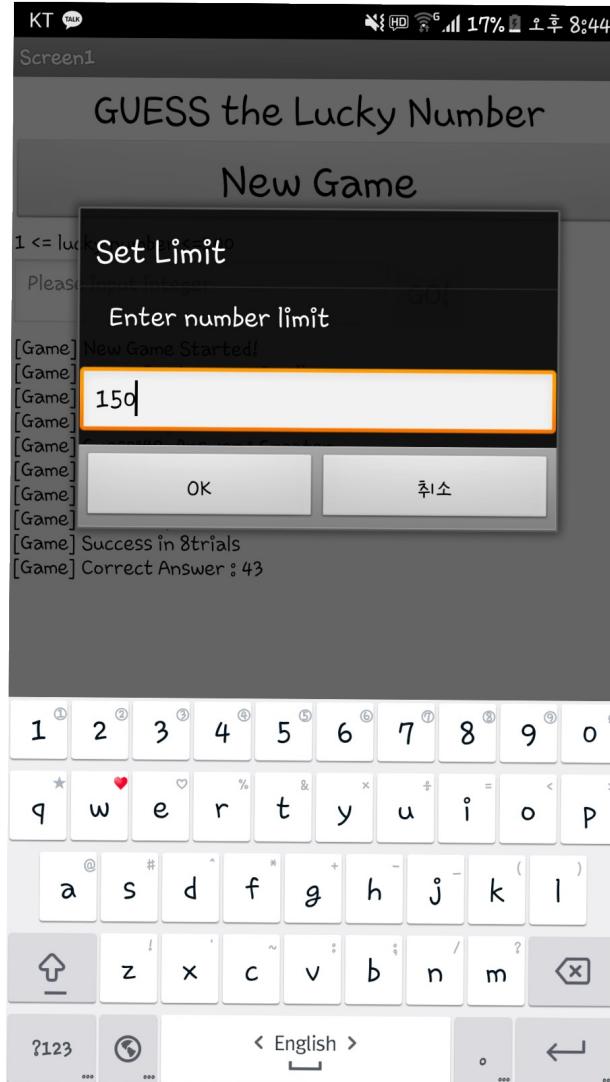
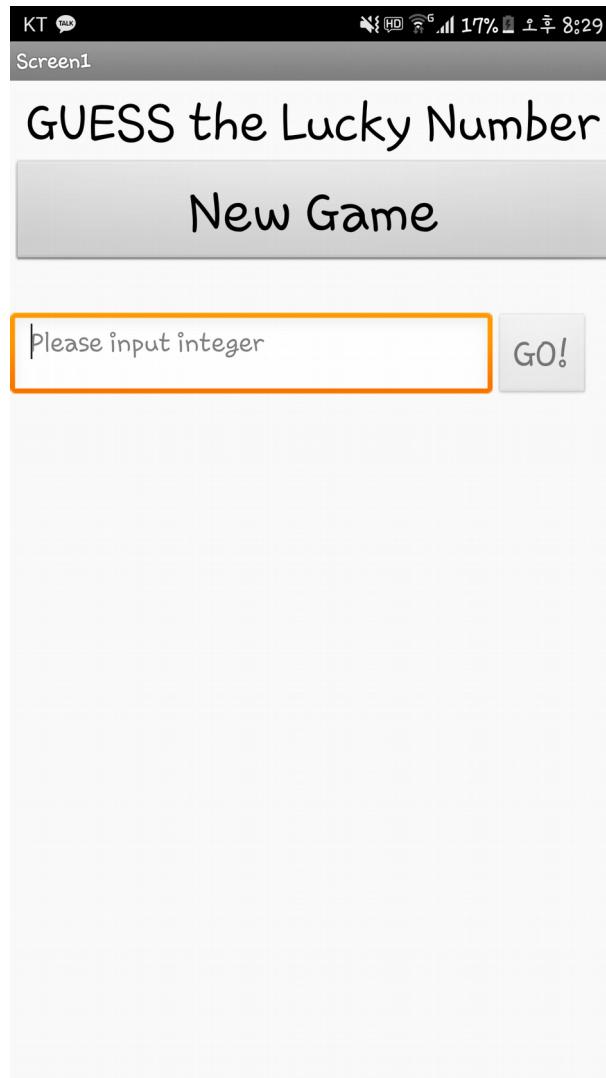
20181676 이지선

목차

- 요구 사항
- 설계 과정
 - 1) API 설계
 - 2) 게임의 핵심 logic 설계
 - 3) API 구현 – WSGI Application
 - 4) 클라이언트 개발 – App Inventor
- 테스트 과정
 - 1) 게임 핵심 logic 이용한 테스트
 - 2) localhost의 터미널에서 curl을 이용한 테스트
 - 3) 시스템 통합 테스트
- 고찰 및 토의

요구 사항

마음 속으로 정한 최대 숫자 범위를 지정해준다!!
(1부터 입력한 수 까지 - 단, 100이상)



만약 100보다 작은 수를
입력했을 경우

요구 사항

* 게임 중에는 GO버튼을 활성화 시킨다.

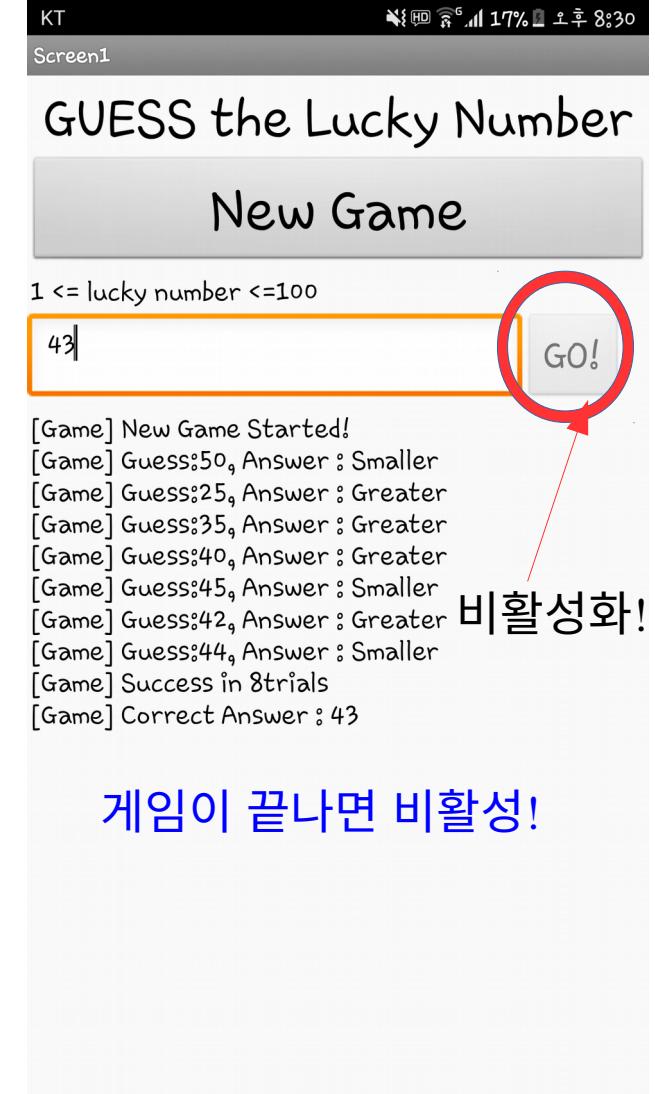
* 게임이 끝나면 비활성화 시킨다.



활성화!



활성화!



게임이 끝나면 비활성!

게임 시작과 게임 중에는 활성화!

클라이언트와 서버의 인터페이스 정의

- 1부터 입력한 수 사이에 있는 Lucky Number 맞추기
→ <http://server/game/guess>
- 새 게임 시작
→ <http://server/game/new>

설계 과정

API 설계 – NEW GAME START

1. URL

→ `http://server/game/new`

2. Input(입력) : POST request body 를 이용한다.

→ `max1 = variable`

`variable` : guess number의 최대 숫자 범위를 지정해주는 입력 인자

3. Output (json 응답) : json 형태로 출력한다.

→ 성공의 경우 : { ‘code’ : ‘success’ }

→ 실패의 경우 : { ‘code’ : ‘error’, ‘msg’ : ‘**max is not given**’ }

설계 과정

API 설계 - Lucky Number 맞추기 시작

1. URL

→ `http://server/game/guess`

2. Input(입력) : POST request body를 이용한다.

→ `guess = guess_num`

`guess_num` : 게임 사용자가 숫자를 추측한 후, 입력하여 맞추기 시도(문자열)

3. Output (json 응답) : json 형태로 출력한다.

→ 성공 : { ‘code’ : ‘success’, ‘Answer’ : ss, ‘trials’ : trials }

→ 실패 : { ‘code’ : ‘error’, ‘msg’ : **MSG** }

MSG : 어떤 에러가 발생했는지 보여준다.

설계 과정

API 설계 - 코드

```
1 from number import Number
2
3 sb = Number()
4
5 def new_game(d):
6     try:
7         max1 = int(d.get('max1', [''])[0])
8     except:
9         return {'code' : 'error', 'msg' : 'max is not given'}
10
11     sb.newGame(max1)
12
13     return {'code': 'success'}
14
15
16 def guess(d):
17     try:
18         guess = d.get('guess', [''])[0]
19     except:
20         return {'code': 'error', 'msg': 'wrong guess parameter'}
21
22     ss = sb.guess(guess)
23     trials = sb.getGuessCount()
24
25     return {'code': 'success', 'Answer':ss, 'trials': trials}
```

설계 과정

게임의 핵심 Logic 설계

class Number:

```
def __init__(self):  
    self.secret = 0  
    self.trials = 0
```

```
def newGame(self, max1):  
    self.secret = random.randint(1,max1)  
    self.trials = 0
```

```
def guess(self, userGuess):  
    self.trials += 1  
    h="Nothing"  
    if(int(userGuess)<self.secret):  
        h="Greater"  
    elif (int(userGuess) > self.secret):  
        h="Smaller"  
    else:  
        h="Correct"  
    return h
```

```
def getGuessCount(self):  
    return self.trials
```

- 1) Class 정의(id)를 생성한다.
 - 2) 생성자 method를 생성 후, 객체를 초기화한다.
 - 3) newGame method
→ lucky number 값을 random함수로 구해준다.
 - 4) guess method
시도 횟수 증가
 - 1) lucky number 보다 클 경우
 - 2) lucky number 보다 작을 경우
 - 3) lucky number 와 같을 경우로 나눈다.
- * getGuessCount method
-> 시도 횟수를 반환

```
if __name__ == '__main__':
    s = Number()
    max1=int(input("max1 >> "))
    while(max1<100):
        print("max is greater than 100")
        max1=int(input("max = "))

    * import되어 사용되지 않을 경우!!
    * instance 생성
    * max1값을 지정
    ->max1값이 100이상이 아닐 경우 다시 지
```

```
s.newGame(max1)                                * 새로운 게임 시작
ss="Nothing"                                     * 추측 값이 맞지 않을 경우 실행
while (ss!="Correct"):
    inputString = input("Your guess: ")
    while (int(inputString)<1) | (int(inputString)>max1):    * 추측 값이 범위를 벗어난 경우
        print("Input between 1 and %d !" %max1)                  →추측 값을 입력
    inputString = input("Your guess: ")

    ss=s.guess(inputString)                                    * 추측 값이 lucky number보다
    print("%c %%(ss))                                         큰지 작은지 판단한 후 출력
```

```
guessCount = s.getGuessCount()                   * 추측의 종료 지금까지 시
print("SUCCESS in %d trials" % guessCount)      도한 횟수를 출력
```

설계 과정

API 구현 – WSGI Application

```
def application(environ, start_response):
    error = False
    if environ['REQUEST_METHOD'] != 'POST': → POST로 요청을 받지 못한 경우
        response = {'code': 'error', 'msg': 'wrong HTTP method'} → Error 메시지를 response에 저장
        error = True
    if not error: → 요청 받은 API 파악
        try:
            path = environ['PATH_INFO'].split('/') → environ에 주어진 환경변수에서
            if len(path) == 2:                     request path를 검사 한 후, 어떤 API가
                method = path[1]                  호출됐는지 판단
            else:
                response = {'code': 'error', 'msg': 'wrong API path'} → API경로가 잘못된 경우
                error = True
        except:
            response = {'code': 'error', 'msg': 'wrong API path'} → API경로가 잘못된 경우
            error = True
```

설계 과정

try:

```
    request_body_size = int(environ.get('CONTENT_LENGTH', '0'))
```

except ValueError:

→ request_body_size가 형태 변환될 수 없을 경우에 Error 발생

```
    request_body_size = 0
```

```
request_body = environ['wsgi.input'].read(request_body_size)
```

```
d = parse_qs(request_body)
```

if not error:

if method == 'new':

```
        response = new_game(d)
```

elif method == 'guess':

```
        response = guess(d)
```

else:

```
        response = {'code': 'error', 'msg': 'non-existent API method'}
```

* http://server/game/zzz의
zzz자리에 들어간 내용이

1. “new” 인 경우

2. “guess” 인 경우

3. 1,2 모두 아닌 경우

→ 3가지 경우로 나누어서 실행

설계 과정

status = '200 OK' → 서버에 성공적으로 접속한 경우

response_body = json.dumps(response) → json형태로 값을 넘겨준다.

response_headers = [

 ('Content-Type', 'application/json'), → json형태로 값을 넘겨준다.

 ('Content-Length', str(len(response_body))) → int형 길이를 문자형으로 바꿔준다.

]

start_response(status, response_headers)

return [response_body] → json형태로 결과 값을 넘겨준다.

설계 과정

```
from cgi import parse_qs
import json
from game import new_game, guess

def application(environ, start_response):
    error = False

    if environ['REQUEST_METHOD'] != 'POST':
        response = {'code': 'error', 'msg': 'wrong HTTP method'}
        error = True

    if not error:
        try:
            path = environ['PATH_INFO'].split('/')
            if len(path) == 2:
                method = path[1]
            else:
                response = {'code': 'error', 'msg': 'wrong API path'}
                error = True
        except:
            response = {'code': 'error', 'msg': 'wrong API path'}
            error = True

        try:
            request_body_size = int(environ.get('CONTENT_LENGTH', '0'))
        except ValueError:
            request_body_size = 0

        request_body = environ['wsgi.input'].read(request_body_size)
        d = parse_qs(request_body)

        if not error:
            if method == 'new':
                response = new_game(d)
            elif method == 'guess':
                response = guess(d)
            else:
                response = {'code': 'error', 'msg': 'non-existent API method'}

    status = '200 OK'
    response_body = json.dumps(response)

    response_headers = [
        ('Content-Type', 'application/json'),
        ('Content-Length', str(len(response_body)))
    ]

    start_response(status, response_headers)

    return [response_body]
```

← API의 구현
- WSGI 코드 -

설계 과정

App inventor

- Design

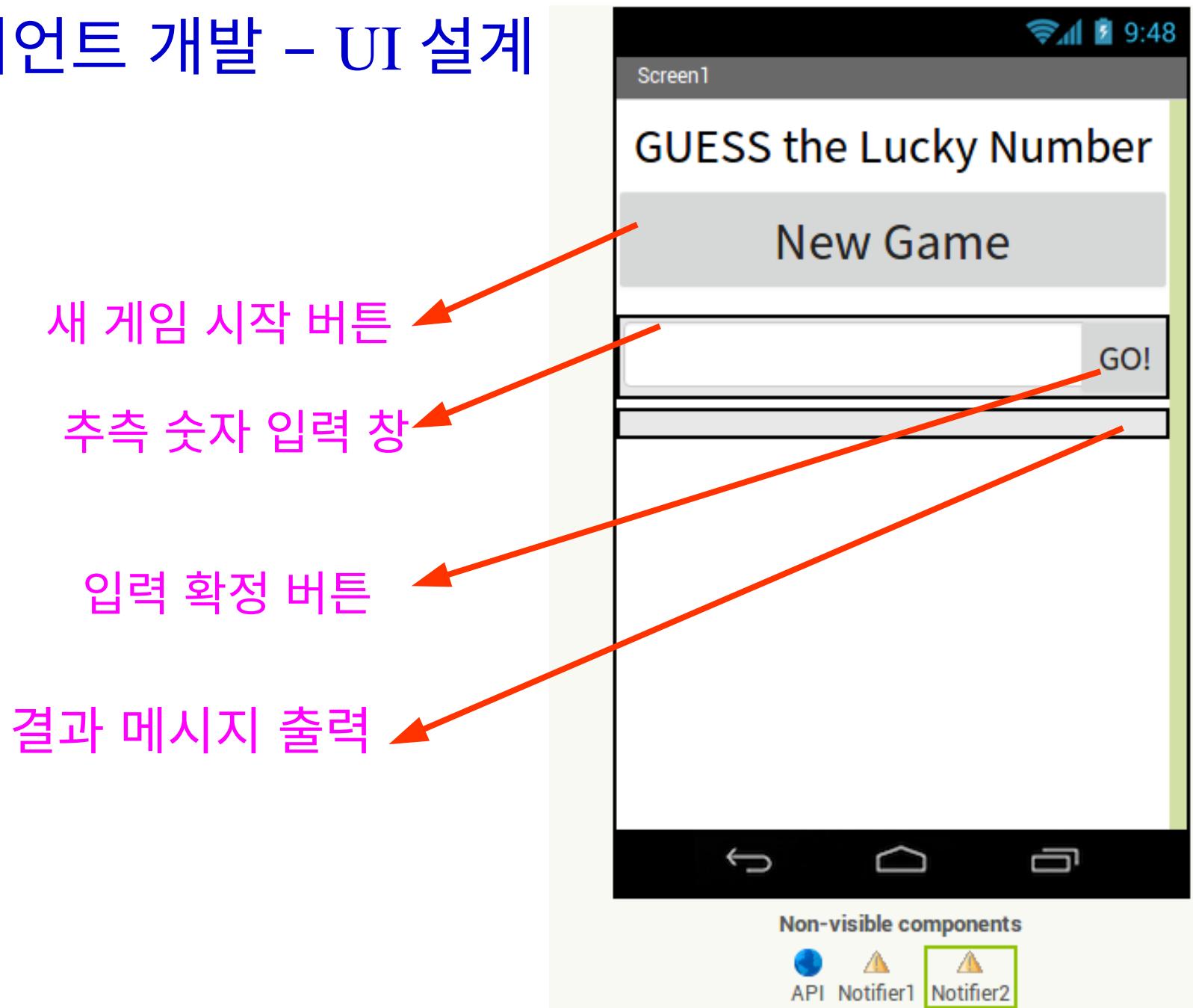
- 새로운 게임을 시작할 수 있는 버튼을 만들고 최댓값을 지정해준다.
- 추측 값을 입력할 수 있는 칸을 만든다.
- GO버튼을 눌렀을 결과가 보이게 한다.
- 결과를 보이게 하는 label을 하나 넣는다.

- Block coding

- 서버 url을 API를 통해 연결한다.
- 모든 값들이 숫자 이외에 다른 것이 들어오면 에러 메시지가 뜨도록 블록 코딩한다.
- 추측 값을 입력할 때, 범위를 벗어나면 에러 메시지가 뜨도록 블록 코딩한다.
- 추측 값이 luck number보다 큰지, 작은지 아니면 똑같은지를 보여주도록 블록 코딩한다.
- 맞췄을 경우 시도 횟수와 luck number를 보여주도록 블록 코딩한다.

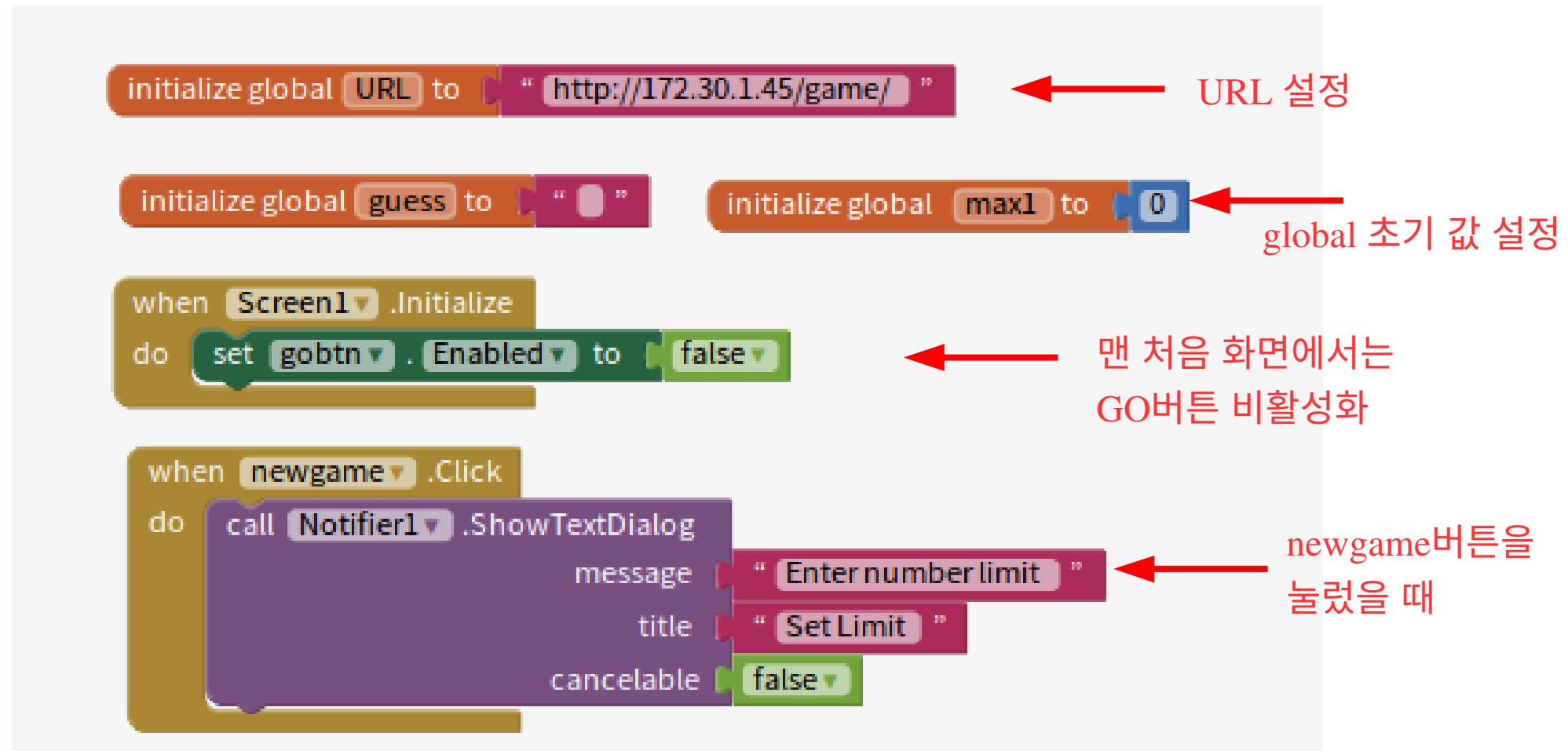
설계 과정

클라이언트 개발 - UI 설계

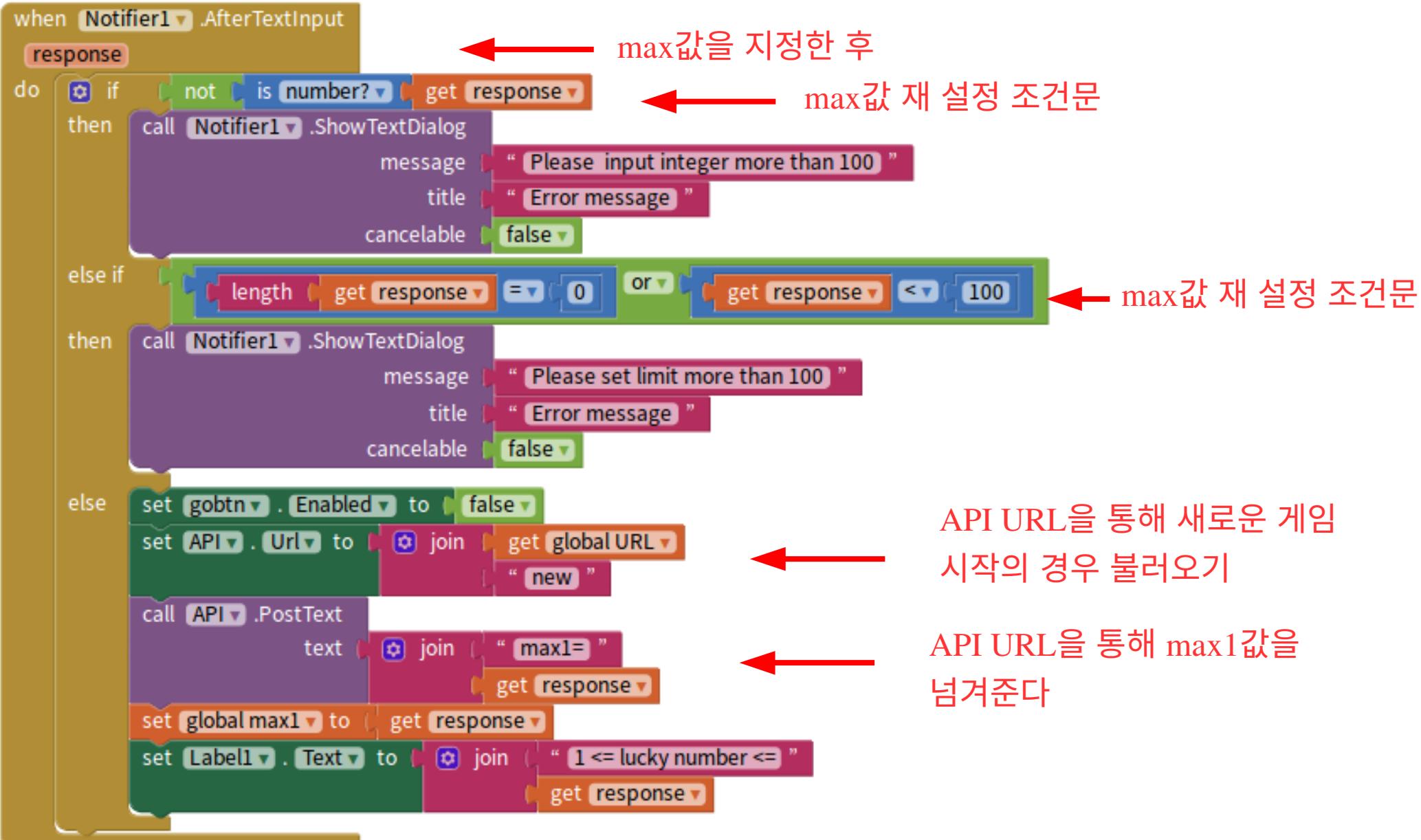


설계 과정

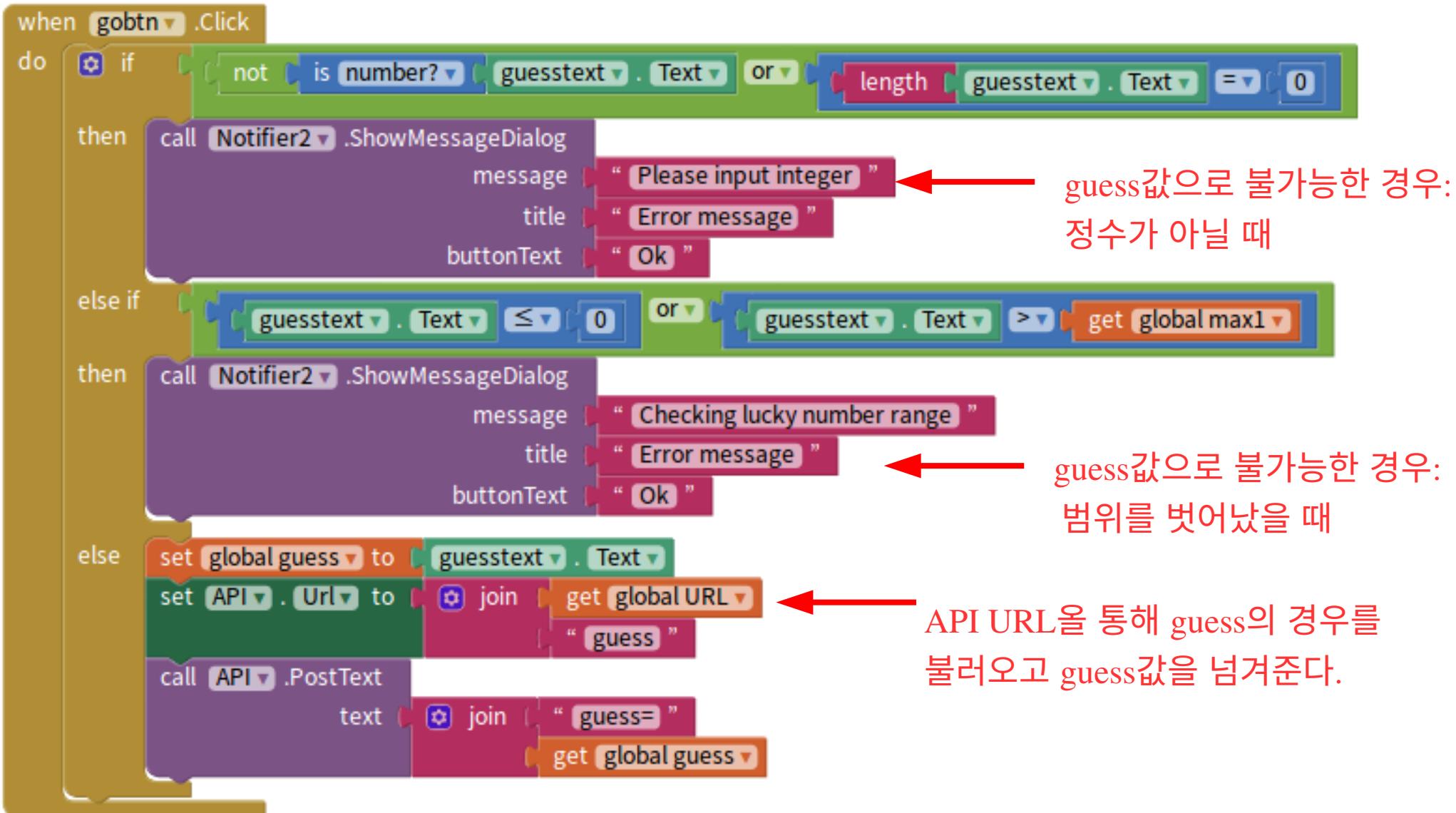
클라이언트 개발 – block coding



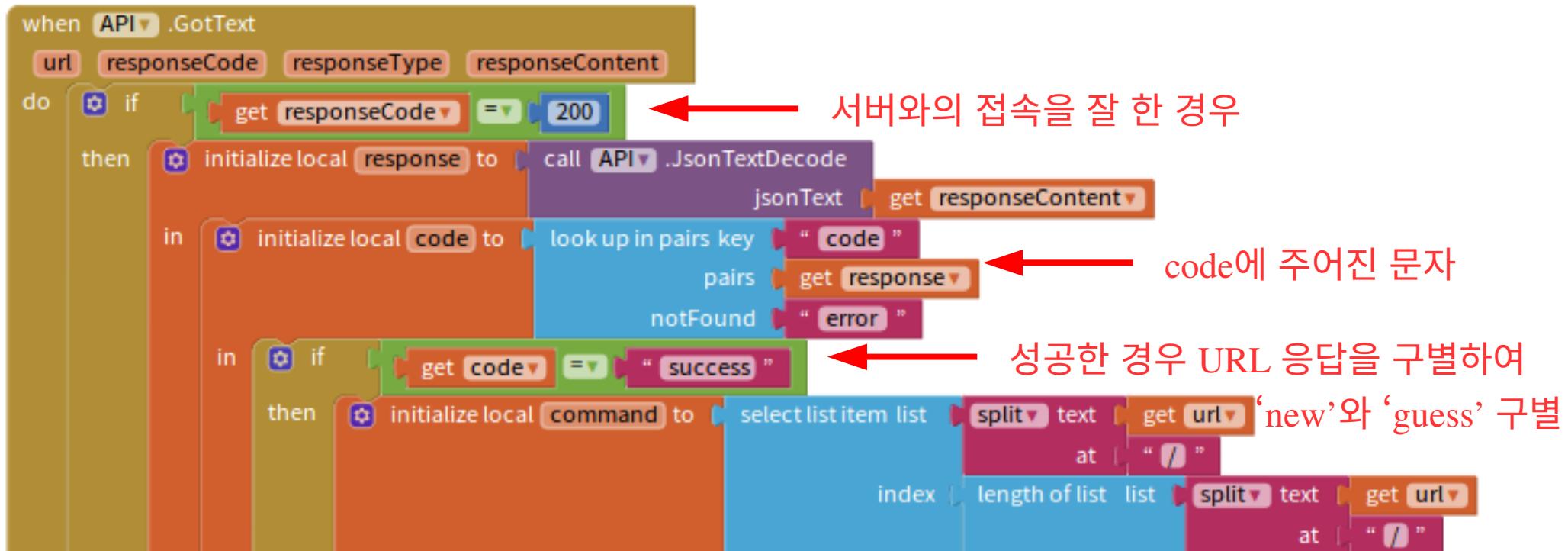
설계 과정



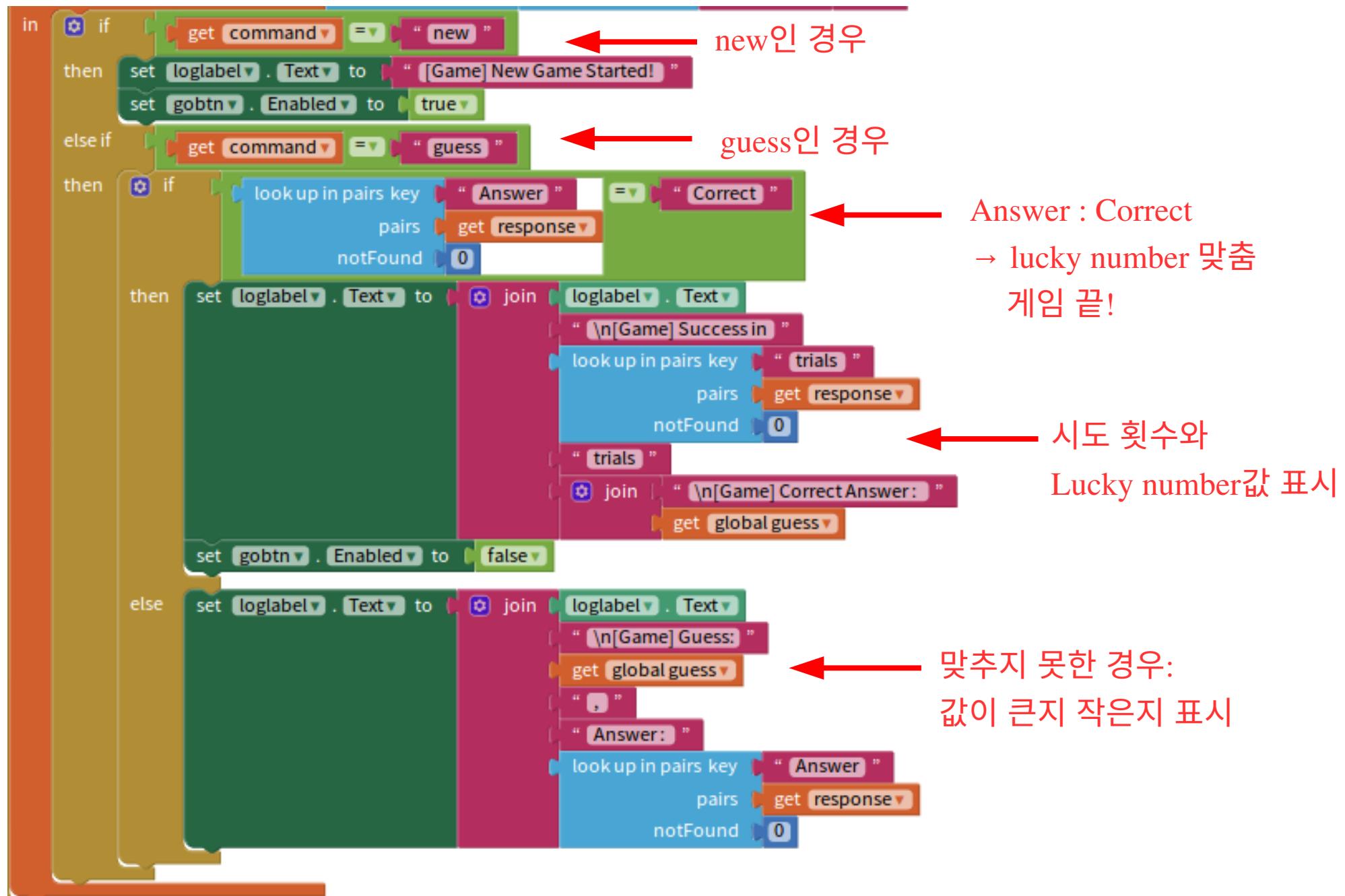
설계 과정



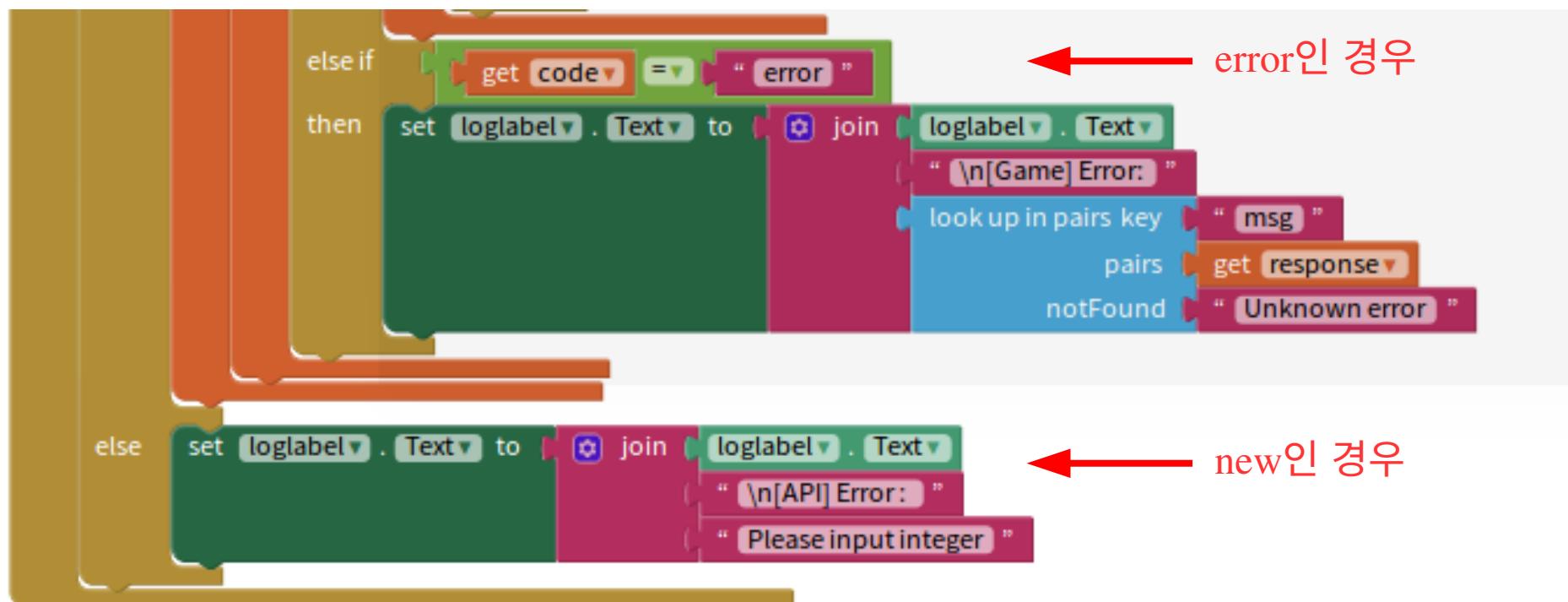
설계 과정



설계 과정



설계 과정



error인 경우

else

set loglabel . Text to



new인 경우

테스트 과정

게임의 핵심 Logic을 이용한 테스트

```
max = 500
Your guess: 250
Greater
Your guess: 300
Greater
Your guess: 350
Greater
Your guess: 400
Greater
Your guess: 450
Greater
Your guess: 470
Smaller
Your guess: 465
Smaller
Your guess: 460
Smaller
Your guess: 455
Greater
Your guess: 457
Correct
SUCCESS in 10 trials
```

```
max = 800
Your guess: 400
Smaller
Your guess: 200
Smaller
Your guess: 100
Smaller
Your guess: 50
Greater
Your guess: 75
Greater
Your guess: 86
Smaller
Your guess: 83
Smaller
Your guess: 80
Greater
Your guess: 82
Correct
SUCCESS in 9 trials
```

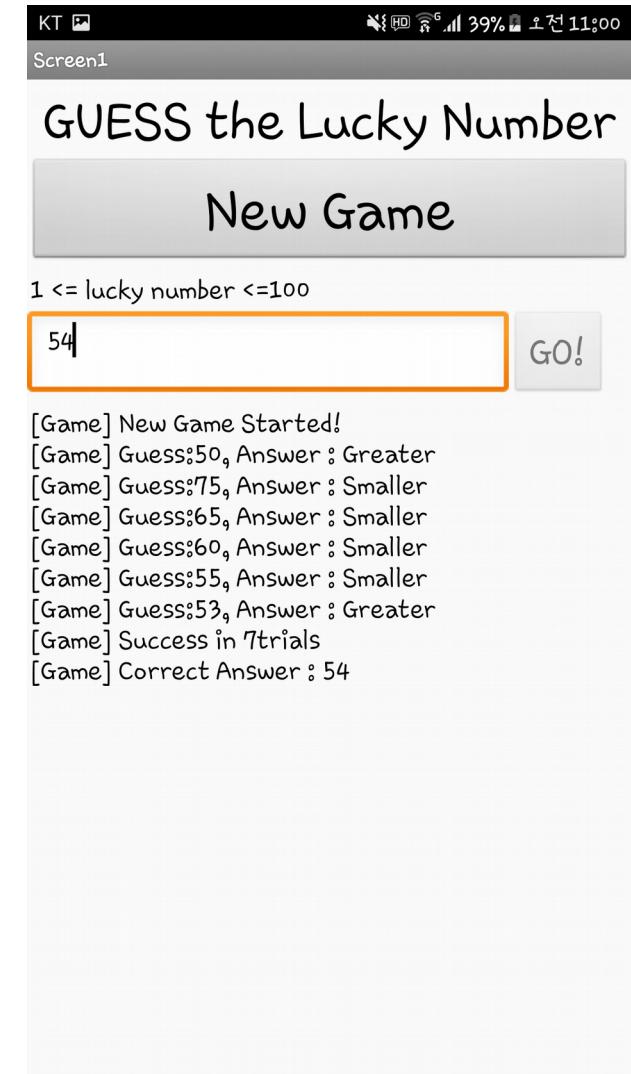
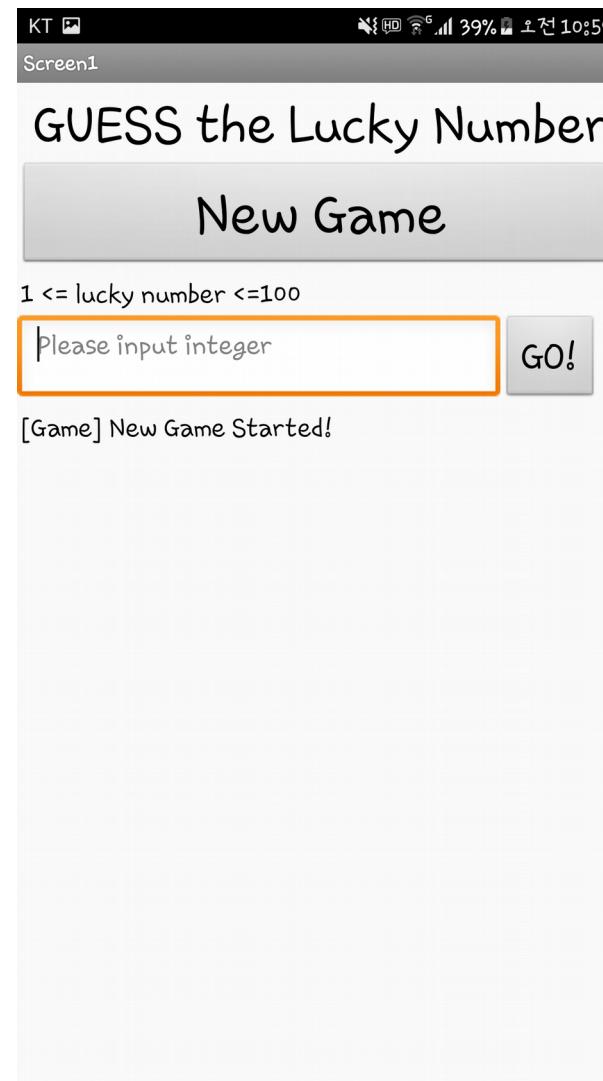
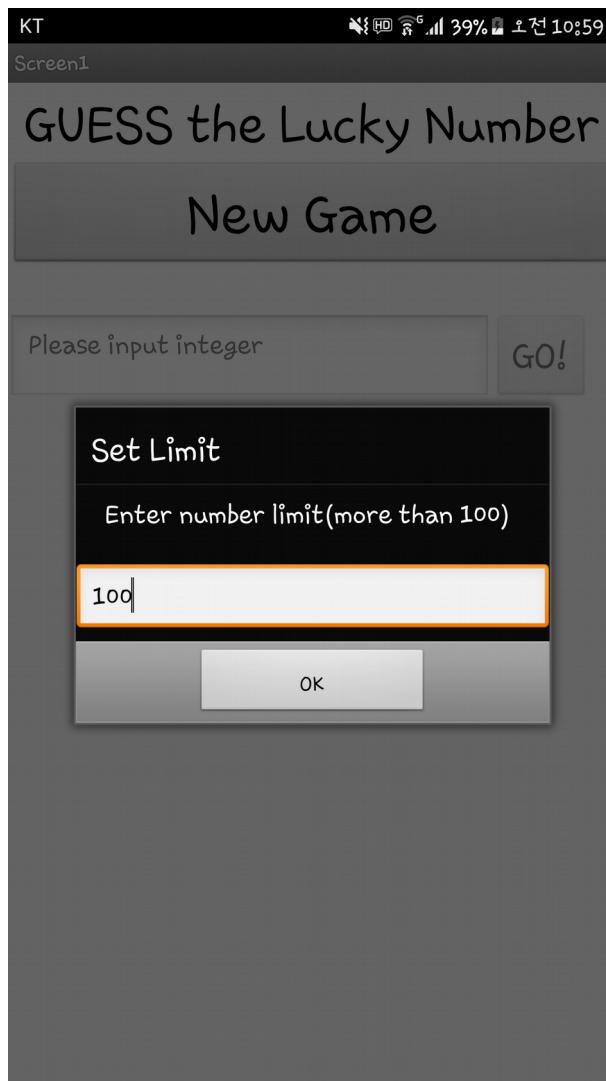
테스트 과정

```
curl -d "max1=1000" http://172.30.1.45/game/new
{"code": "success"}  
  
curl -d "guess=500" http://172.30.1.45/game/guess
{"Answer": "Greater", "trials": 1, "code": "success"}  
  
curl -d "guess=830" http://172.30.1.45/game/guess
{"Answer": "Smaller", "trials": 5, "code": "success"}  
  
curl -d "guess=850" http://172.30.1.45/game/guess
{"Answer": "Smaller", "trials": 3, "code": "success"}  
  
curl -d "guess=820" http://172.30.1.45/game/guess
{"Answer": "Smaller", "trials": 6, "code": "success"}  
  
curl -d "guess=760" http://172.30.1.45/game/guess
{"Answer": "Greater", "trials": 13, "code": "success"}  
  
curl -d "guess=774" http://172.30.1.45/game/guess
{"Answer": "Correct", "trials": 18, "code": "success"}
```

테스트 과정

시스템 통합 테스트(성공의 경우)

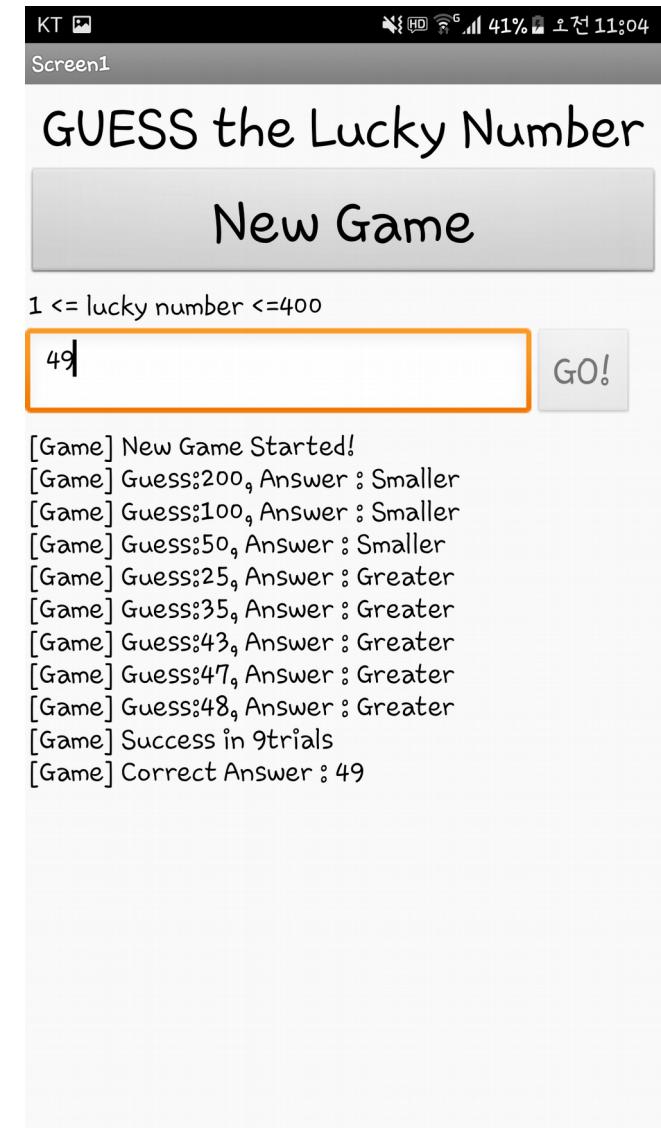
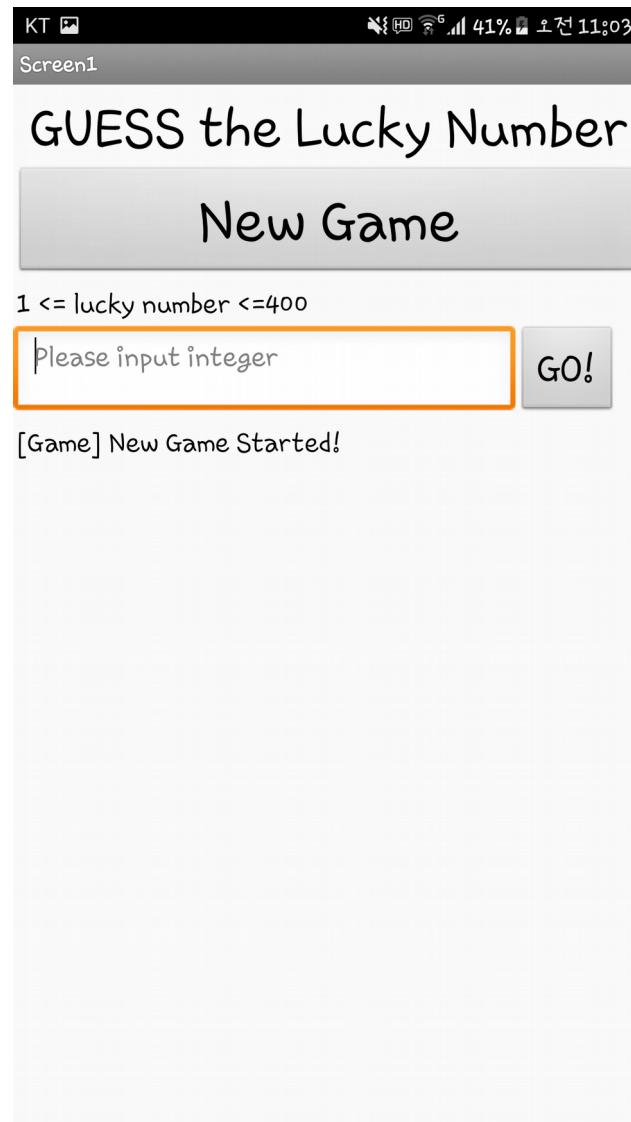
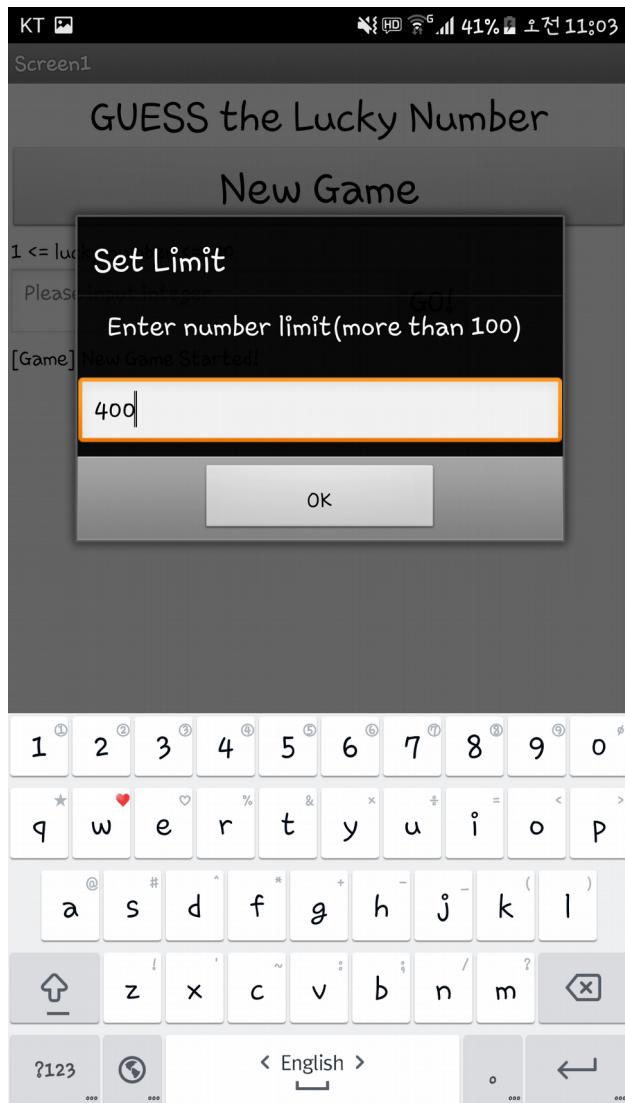
Number limit= 100



테스트 과정

시스템 통합 테스트(성공의 경우)

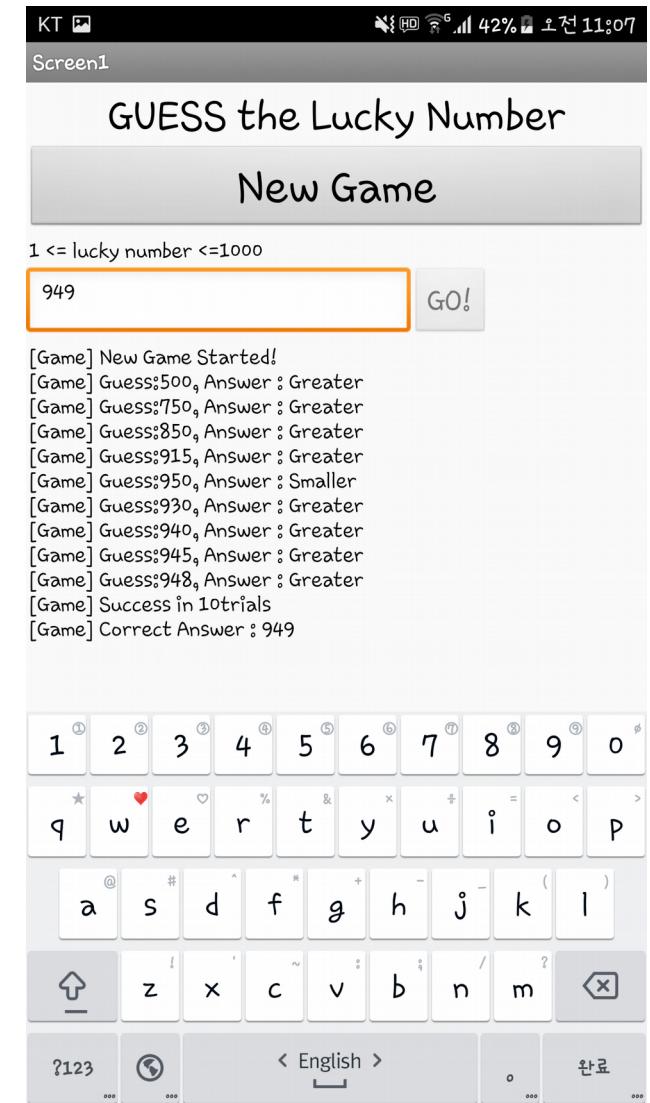
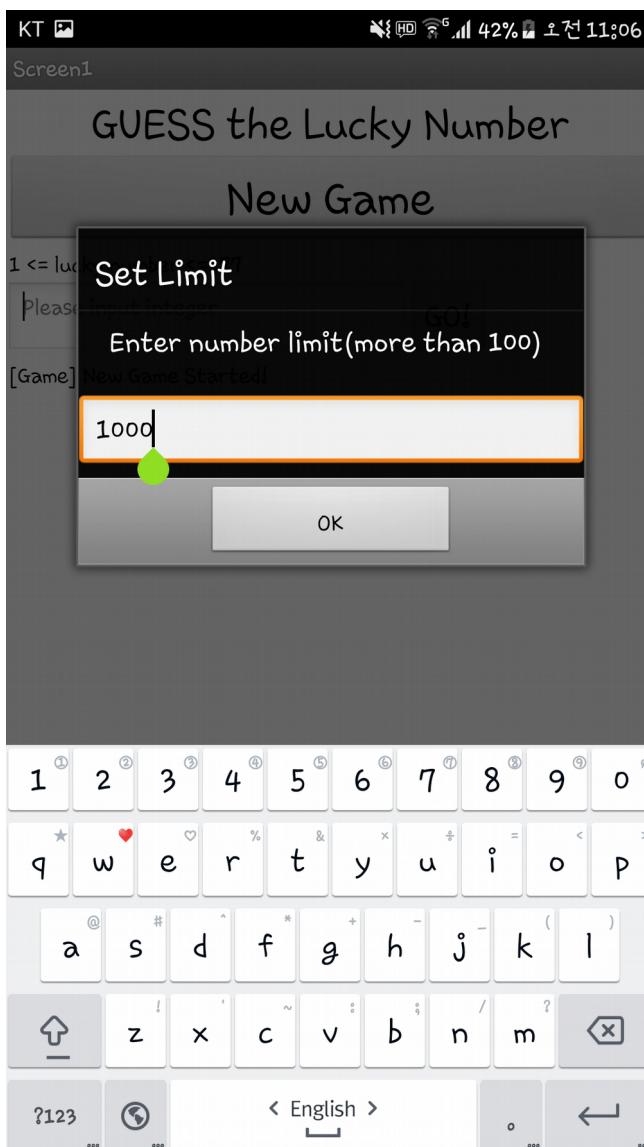
Number limit= 400



테스트 과정

시스템 통합 테스트(성공의 경우)

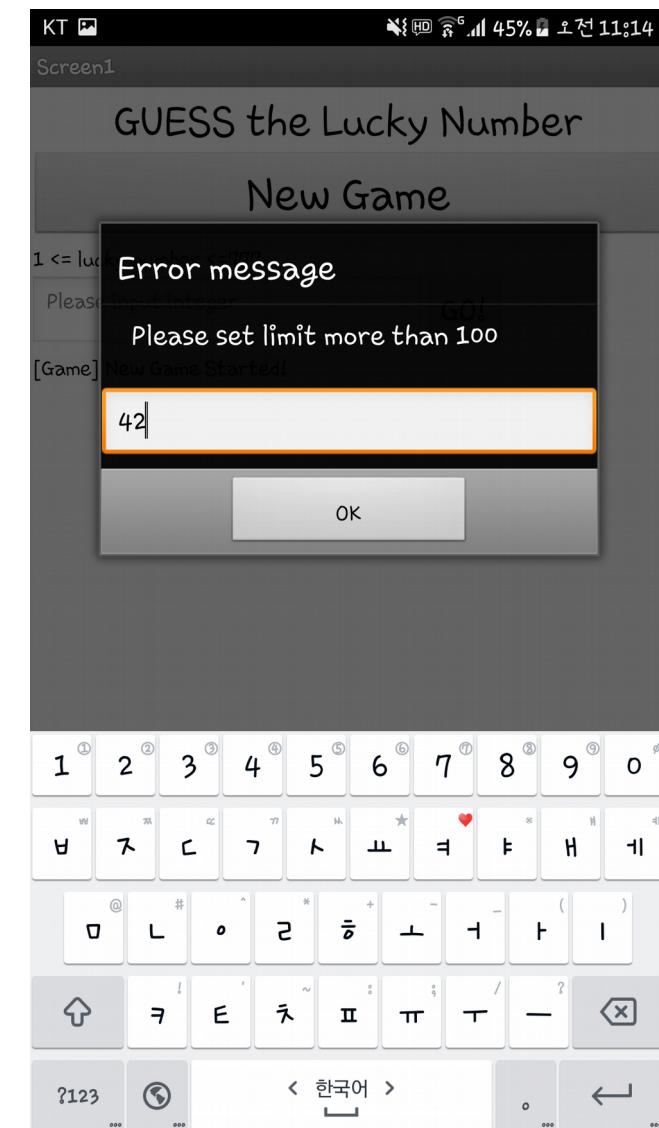
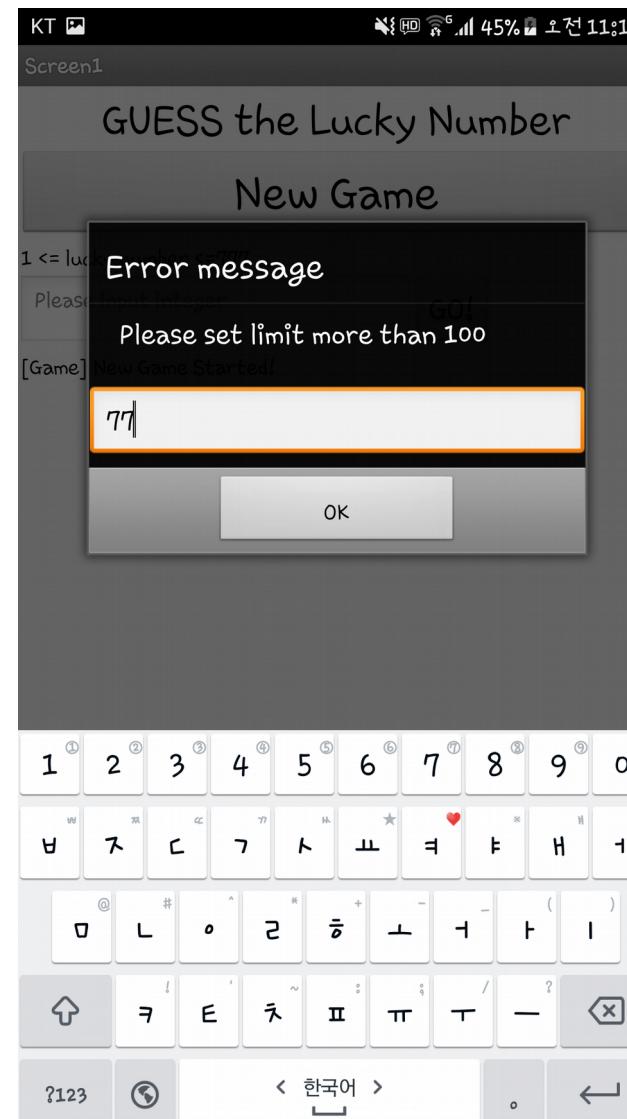
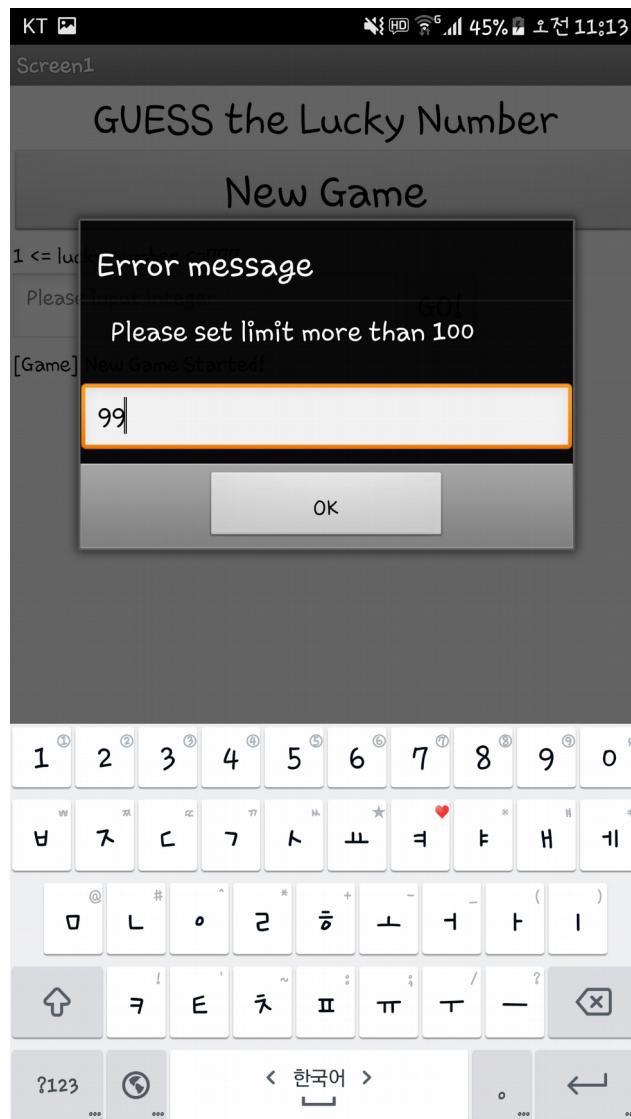
Number limit= 1000



테스트 과정

시스템 통합 테스트 – (Error 처리)

Number limit이 100보다 큰 수가 아니면 에러 메시지를 띄운다.



테스트 과정

시스템 통합 테스트 – (Error 처리)

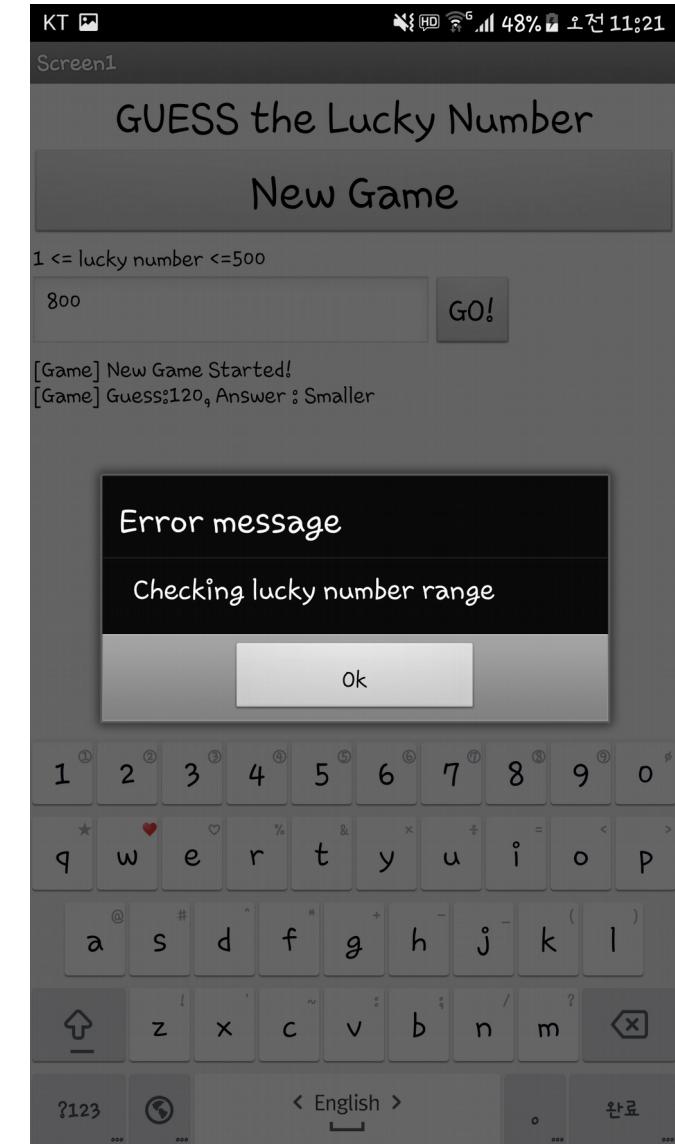
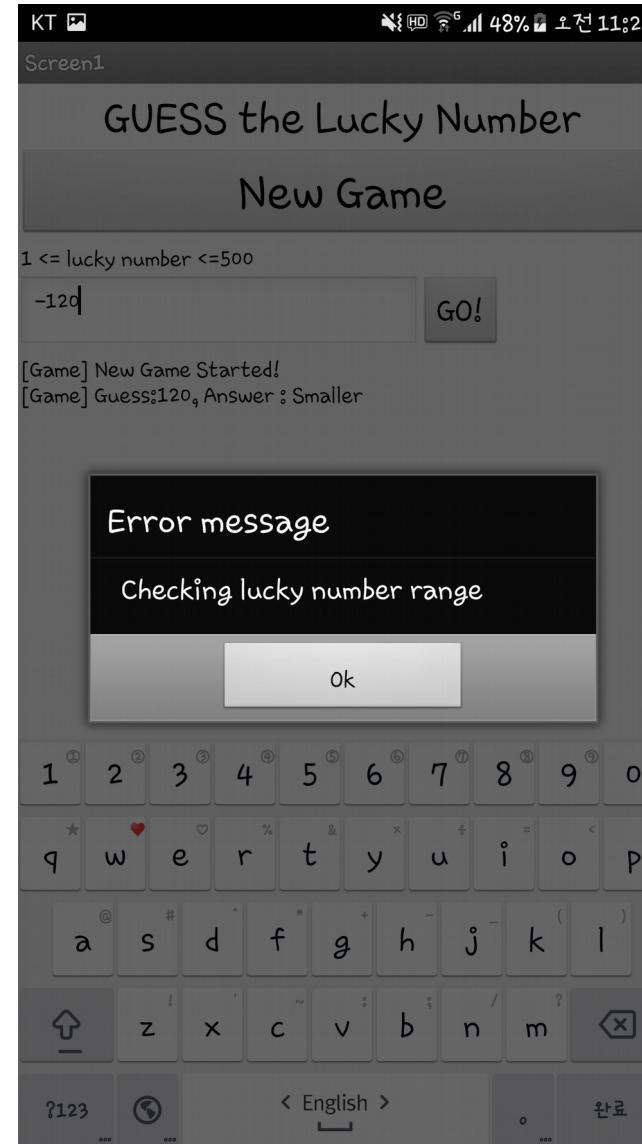
Guess number가 숫자가
아닌 문자인 경우.



테스트 과정

시스템 통합 테스트 – Error 처리

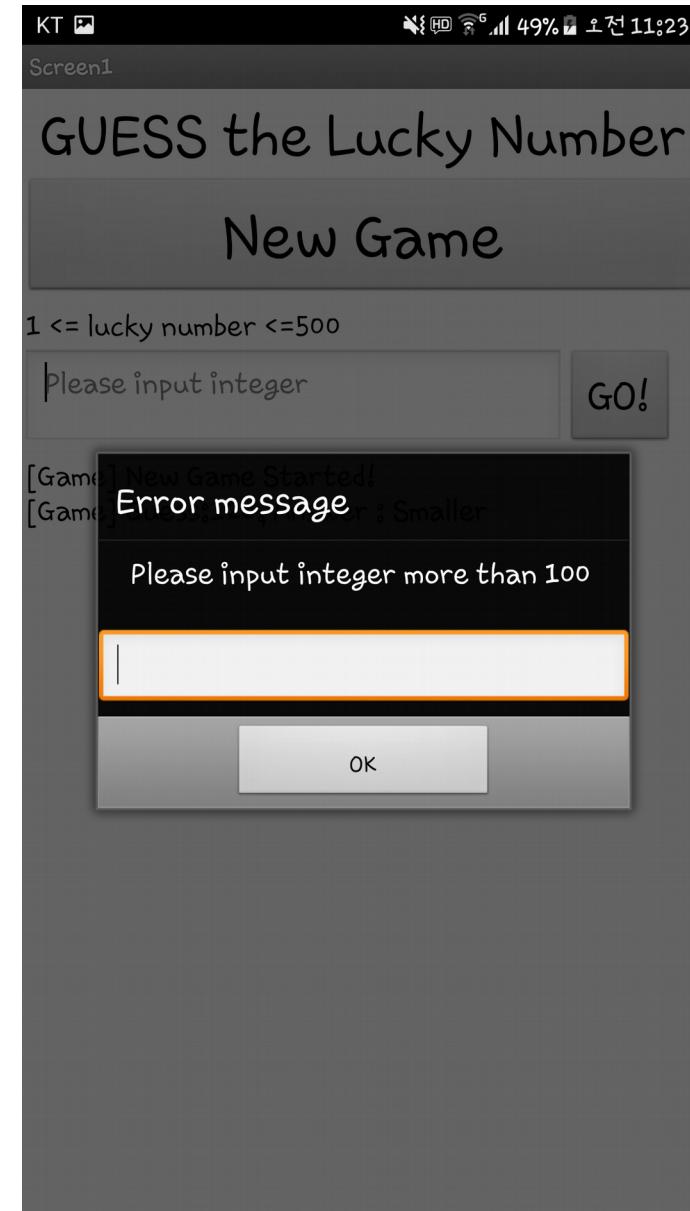
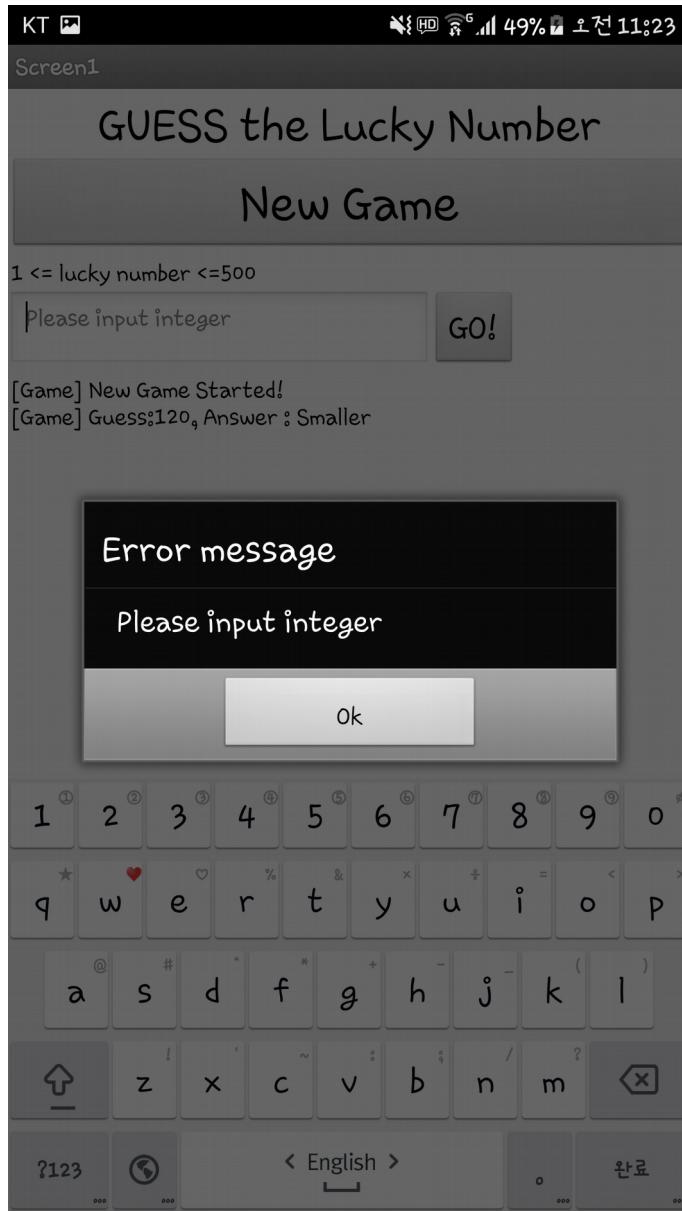
Guess number가 범위를 벗어난 경우



테스트 과정

빈칸을 제출한 경우

시스템 통합 테스트 – (Error 처리)

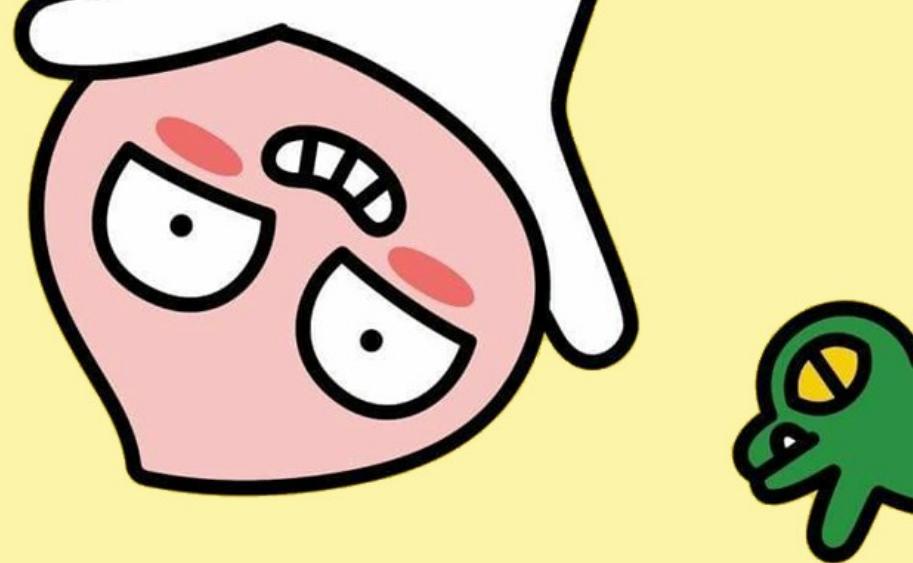


- 고찰

1. 숫자의 범위를 정해주거나 최댓값을 정해줄 때, 빈칸인 경우를 어떻게 하면 좋을까 라고 생각하다가 len블록을 이용하여 0인 경우 일 때를 처리해주니 여러 처리를 할 수 있었다.
2. 입력한 것이 숫자가 아니고 문자일 경우 문자 인지를 판별하는 것보다 숫자가 아닌지 판단해 블록 코딩을 하는 것이 효율적이다.
3. 범위를 벗어난 숫자의 입력을 server에서 처리하지 못하고 block 코딩으로 잡은 점이 아쉽다.

- 토의 : 오류 내용을 잘 파악 했는가?

1. 게임의 규칙을 통해 오류의 종류를 파악했다.
2. 오류 처리 후에 재시작을 구현하였다.
3. 서버 에러 처리를 다양하게 하지 못한 부분이 아쉬운 점이다.



감사합니다 ...★

