

클라이언트 서버 미니 프로젝트



게임 클라이언트
모바일 앱

사용자의 액션을
데이터화하여 전달
→
←
게임 로직을 실행하고
결과를 전달



게임 서버
(API 제공)

학번 : 20181684
이름 : 임현진

목차

1. 요구 사항

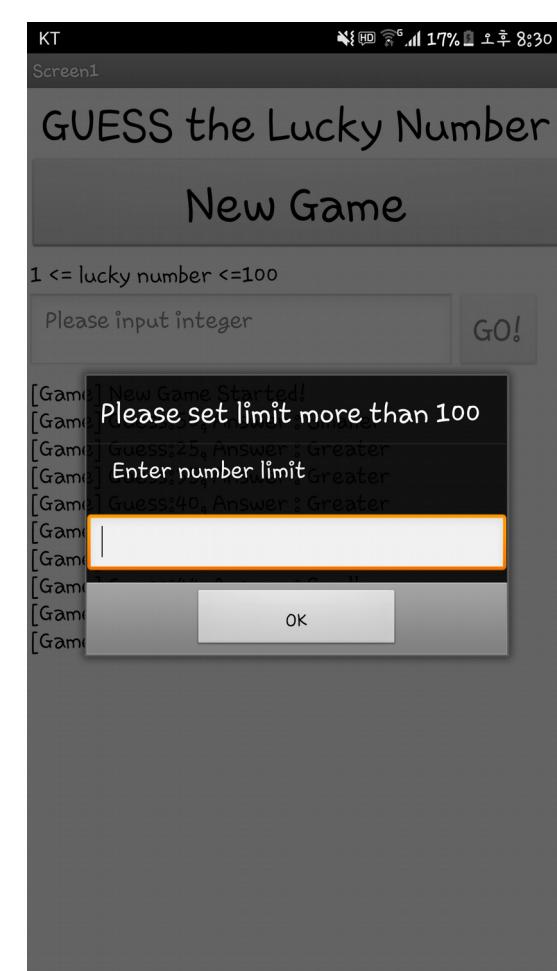
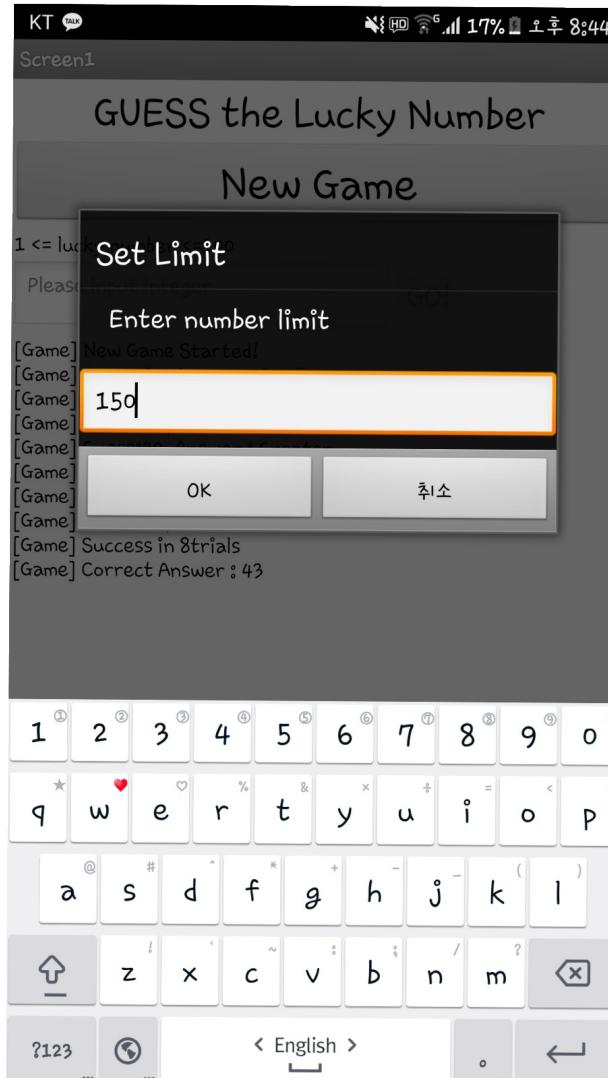
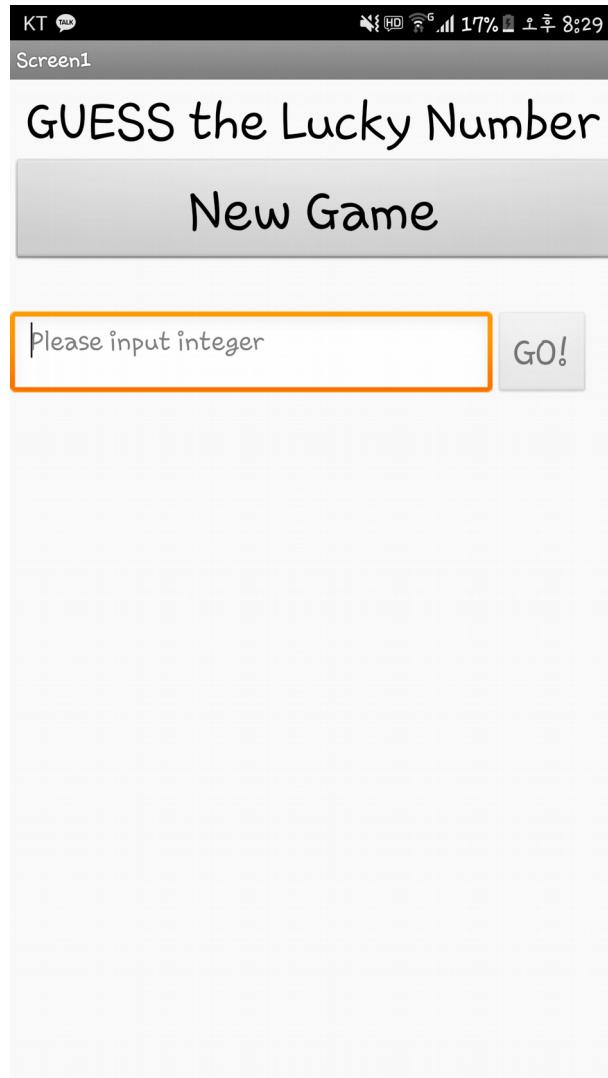
2. 설계 과정

3. 테스트 과정

4. 고찰 및 토의

요구 사항

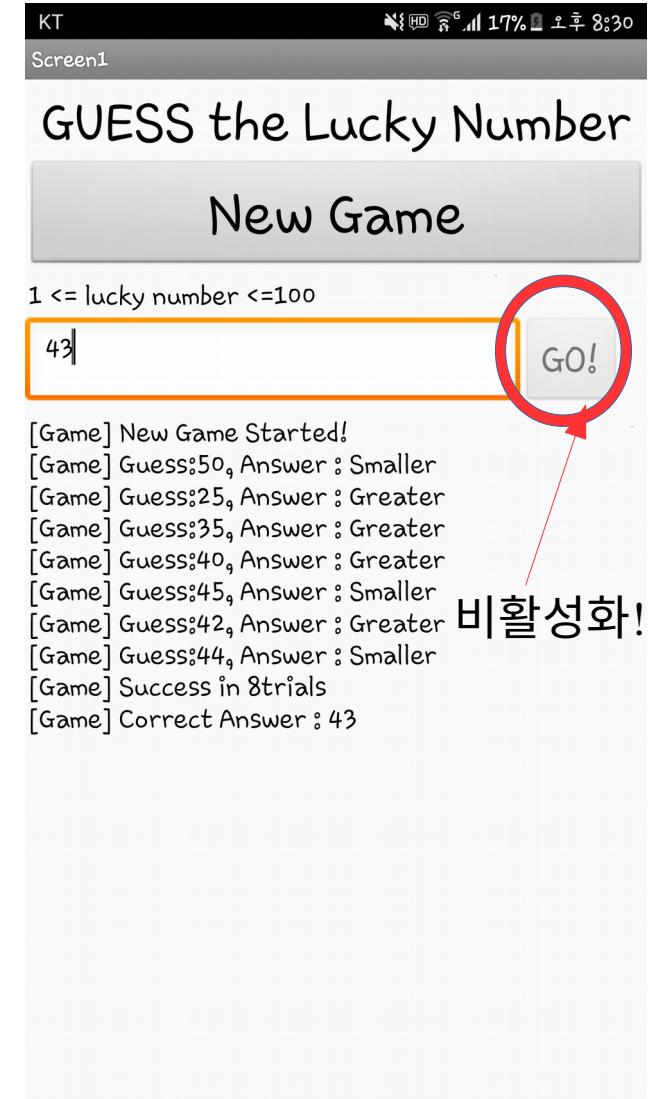
마음 속으로 정한 최대 숫자 범위를 지정!!(1부터 입력한 수 까지...단, 100이상)



만약 100보다 작은 수를
입력했을

요구 사항

게임 중에는 GO버튼을 활성화, 게임이 끝나면 비활성화로 만들기



클라이언트와 서버의 인터페이스 정의

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

설계 과정

API 설계 - 새로운 게임 시작

1. URL

→ `http://server/game/new`

2. Input(입력) : POST request body 를 이용

→ `max1 = n`

`n` : 최대 숫자 범위를 지정해주는 입력 인자

3. Output (json 응답(출력))

→ 성공 : { ‘code’ : ‘success’ }

→ 실패 : { ‘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 body 를 이용

→ `guess = num`

`num` : 게임 사용자가 입력하여 맞추기 시도(문자)

3. Output (json 응답(출력))

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

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

MSG : 어떤 에러가 발생했는지 보여줌

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

API 구현 – WSGI Application

```
def application(environ, start_response):
    error = False
    if environ['REQUEST_METHOD'] != 'POST': * POST 요청이 아닌 경우
        response = {'code': 'error', 'msg': 'wrong HTTP method'} * error 메시지 전송
        error = True
    if not error: * 요청된 API 파악
        try:
            path = environ['PATH_INFO'].split('/')
            if len(path) == 2:
                method = path[1] * environ에 주어진 환경변수에서
                                         request path를 검사 한 후, 어떤
                                         API가 호출됐는지 판단
            else:
                response = {'code': 'error', 'msg': 'wrong API path'} * 잘못된 API 경로일 경우
                error = True
        except:
            response = {'code': 'error', 'msg': 'wrong API path'} * 잘못된 API 경로일 경우
            error = True
```

설계 과정

API 구현 - WSGI Application

try:

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

except ValueError:

```
    request_body_size = 0
```

*request_body_size가 int형으로 형 변환될 수 없을 경우

```
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) 둘 다 아닌 경우”로 나
누어서 처리한다.

설계 과정

API 구현 – WSGI Application

```
status = '200 OK'
```

*서버에 잘 접속했을 때

```
response_body = json.dumps(response) * json형태로 넘겨주기
```

```
response_headers = [
```

```
    ('Content-Type', 'application/json'), * json형태로 넘겨준다.
```

```
    ('Content-Length', str(len(response_body))) * 길이를 문자형으로 바꿔준다.
```

```
]
```

```
start_response(status, response_headers)
```

```
return [response_body]
```

*json형태로 결과 값을 넘겨준다.

설계 과정

API 의 구현 - WSGI 코드 -

```
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]
```

설계 과정

게임의 핵심 Logic 설계

class Number:

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

* Class Number를 생성!

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

* 생성자 method - 객체를 초기화한다.

```
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
```

* newGame method

-> lucky number 값을 구한다.

* guess method

시도 횟수 증가

1) lucky number 보다 클 경우

2) lucky number 보다 작을 경우

3) lucky number 와 같을 경우로 나눈다.

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

* 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 = "))

    s.newGame(max1)                                * 새로운 게임 시작

    ss="Nothing"
    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 다시 입력
        inputString = input("Your guess: ")

        ss=s.guess(inputString)                            * 추측 값이 lucky number보다
        print("%c %%(ss))                                큰지 작은지를 출력

guessCount = s.getGuessCount()                  * 추측이 끝났다면 지금까지
print("SUCCESS in %d trials" % guessCount)    시도한 횟수를 출력
```

설계 과정

App inventor

- Design

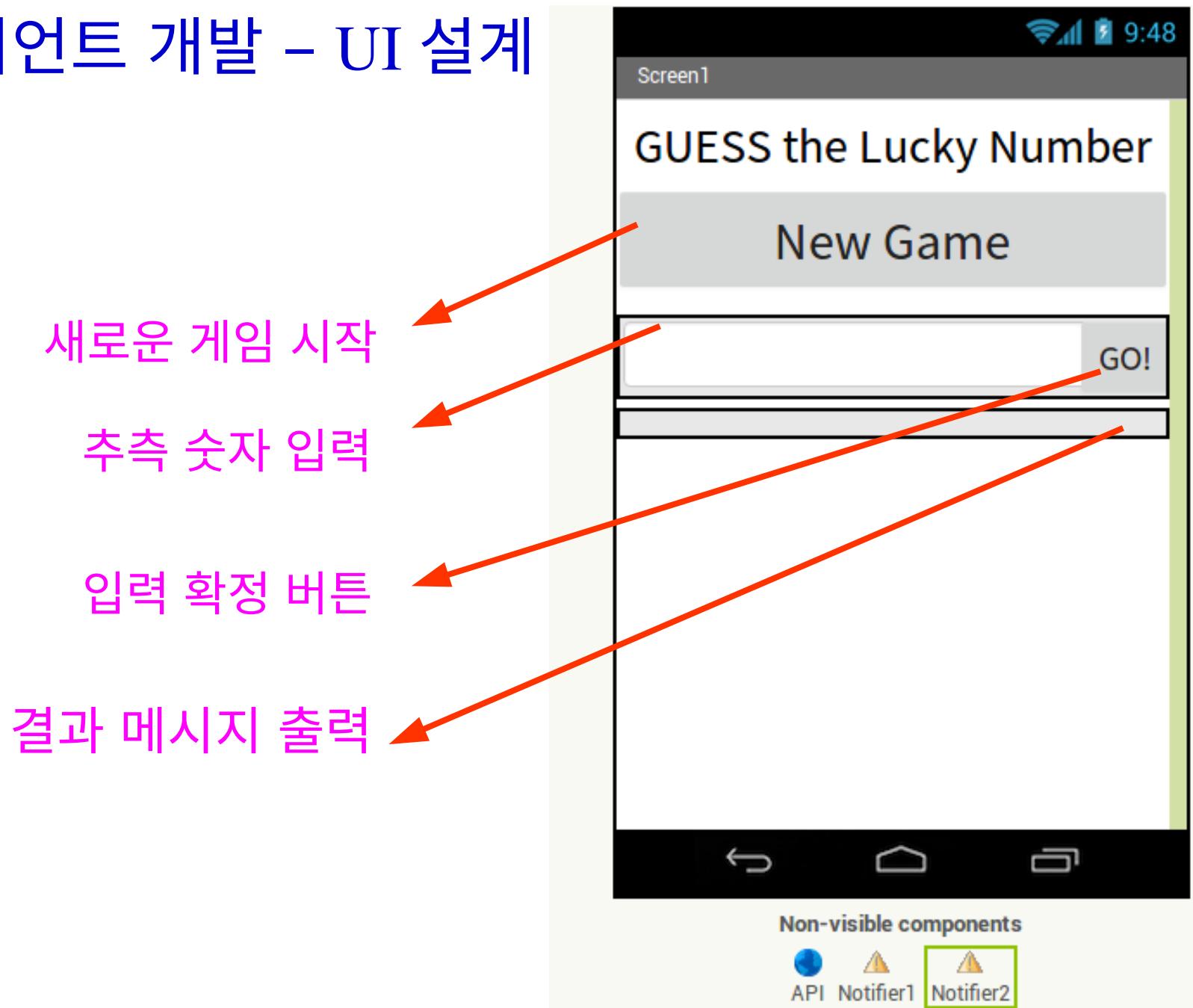
- 새로운 게임을 시작할 수 있는 버튼을 만들고 최댓값을 지정해준다.
- 추측 값을 입력할 수 있는 칸을 만든다.
- GO버튼을 눌렀을 결과가 보이게 한다.
- 결과를 보이게 하는 label을 하나 넣는다.

- 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 딩한다.

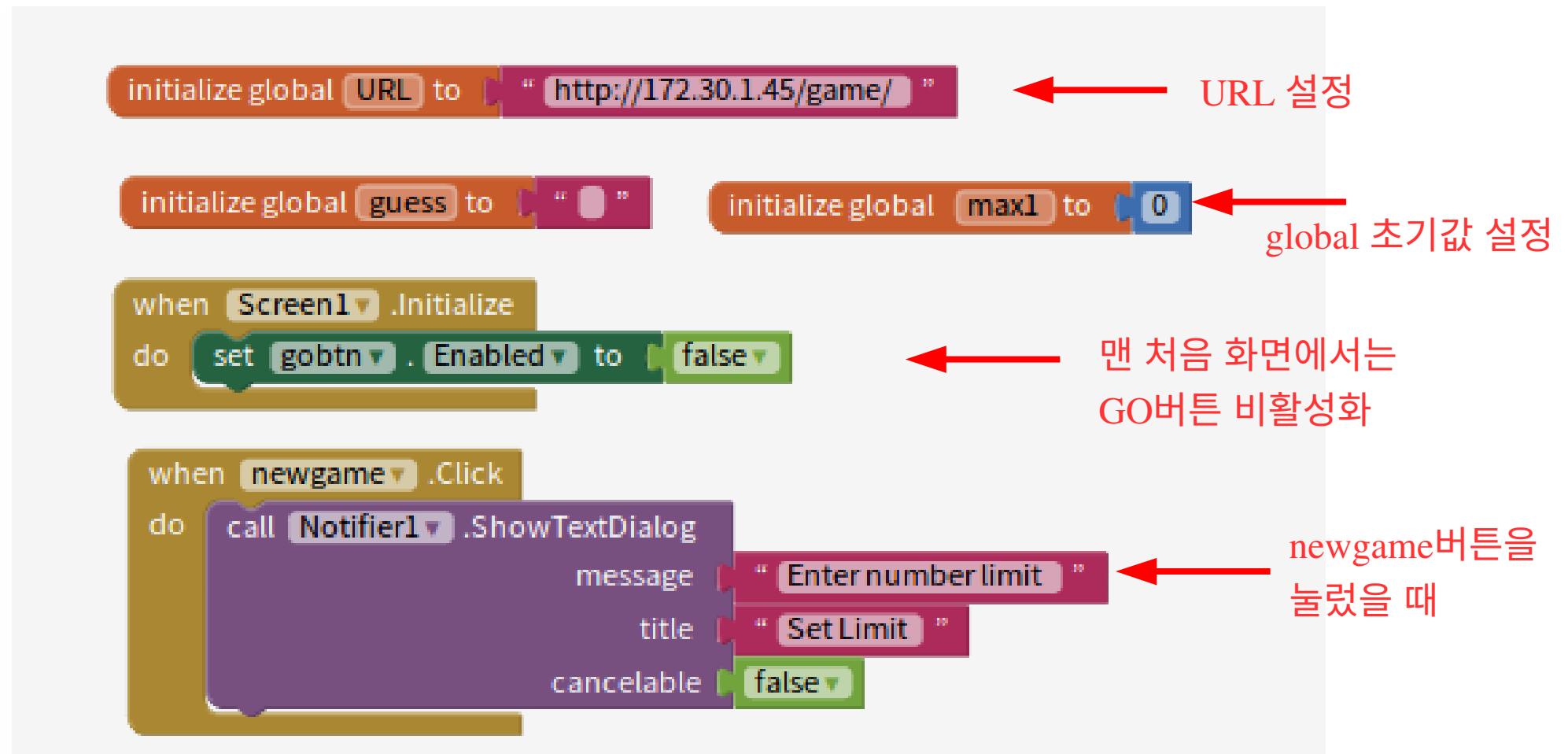
설계 과정

클라이언트 개발 - UI 설계

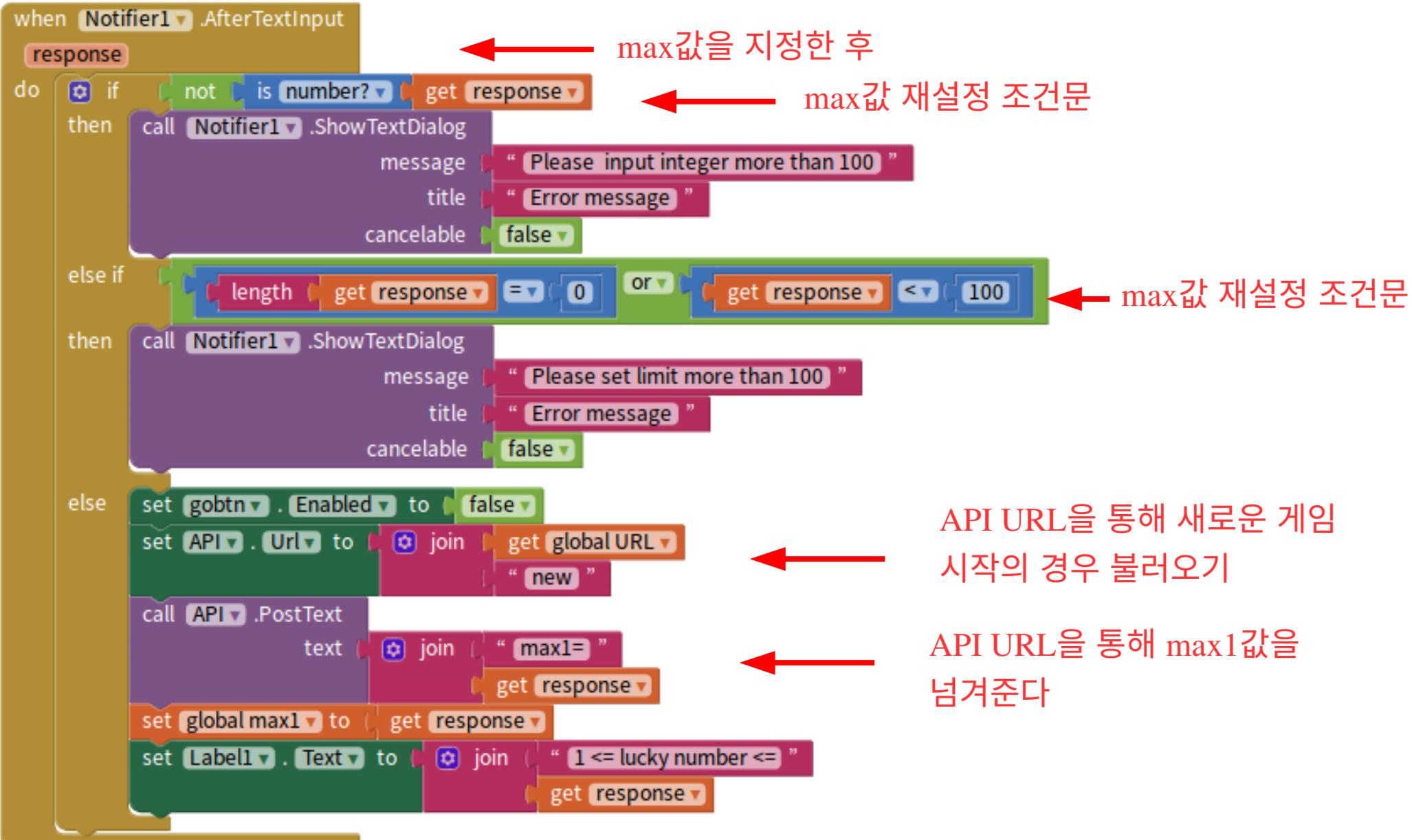


설계 과정

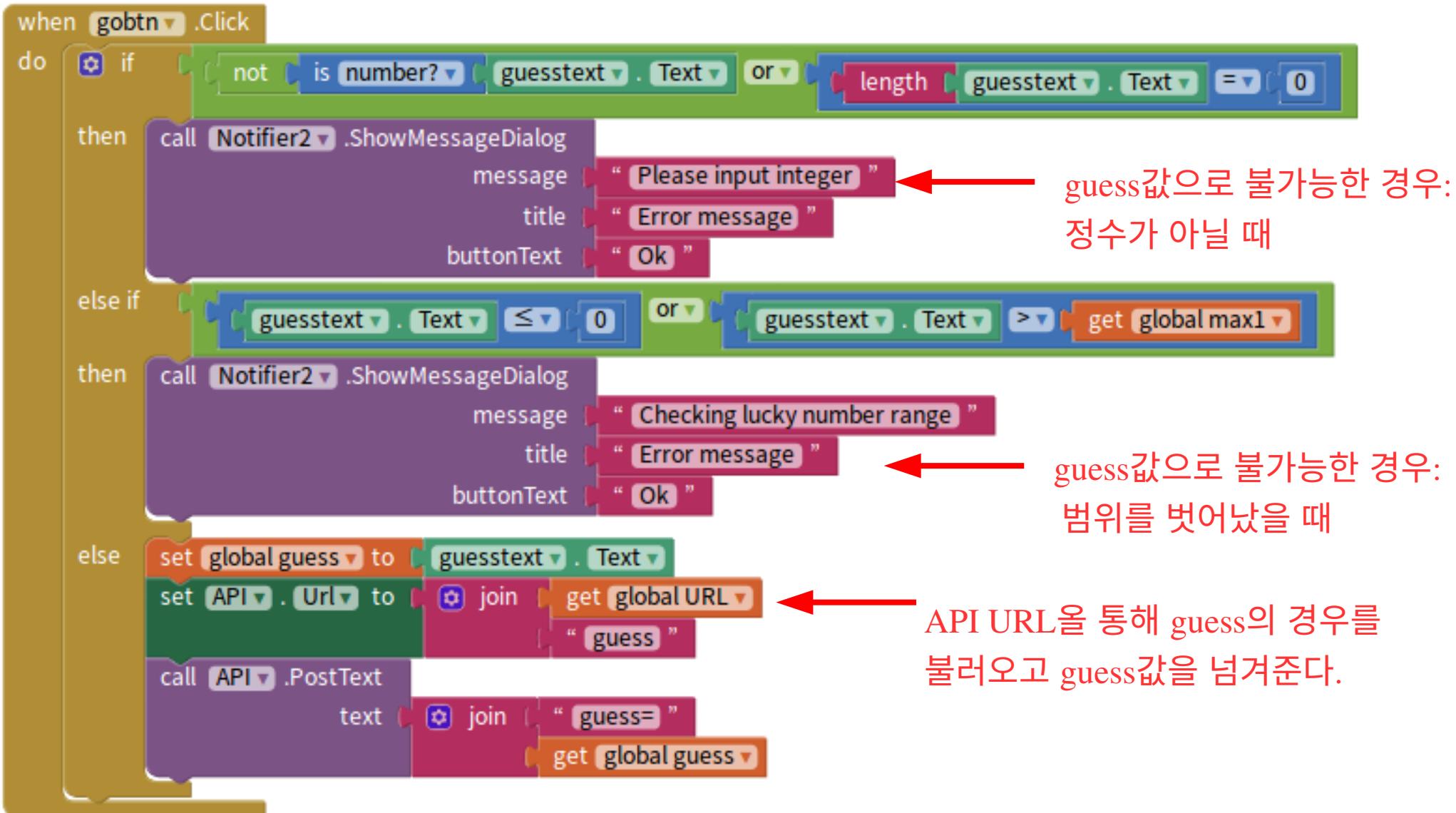
클라이언트 개발 – block coding



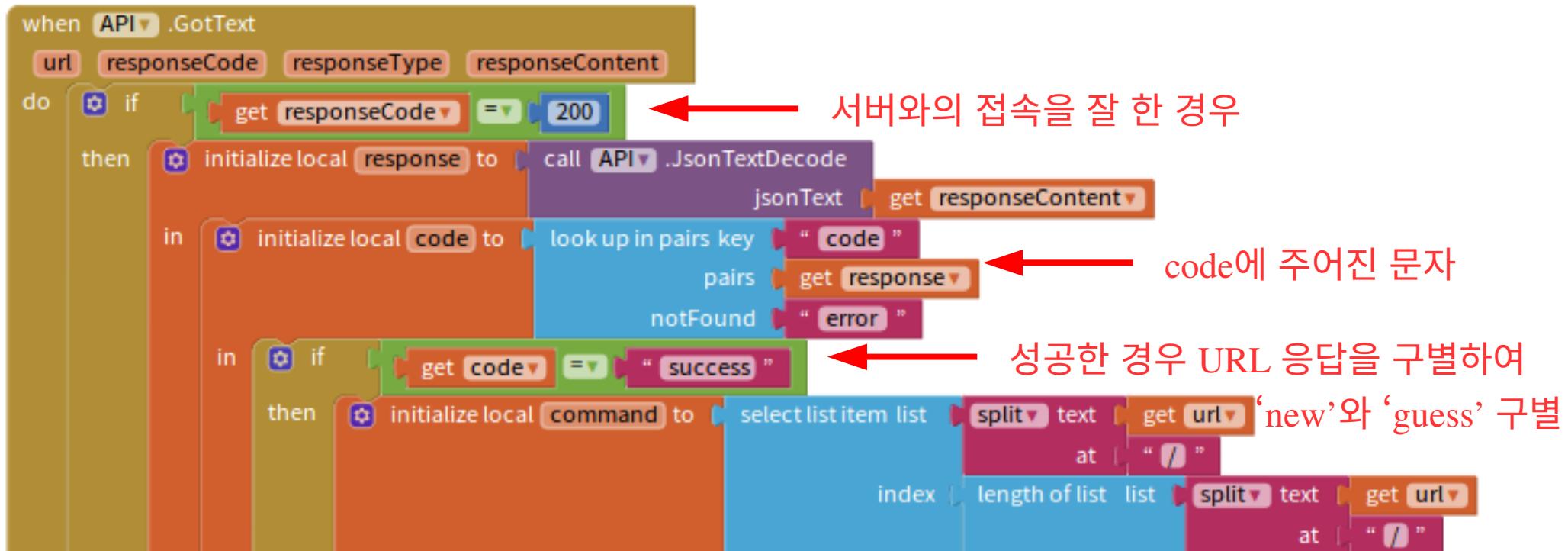
설계 과정



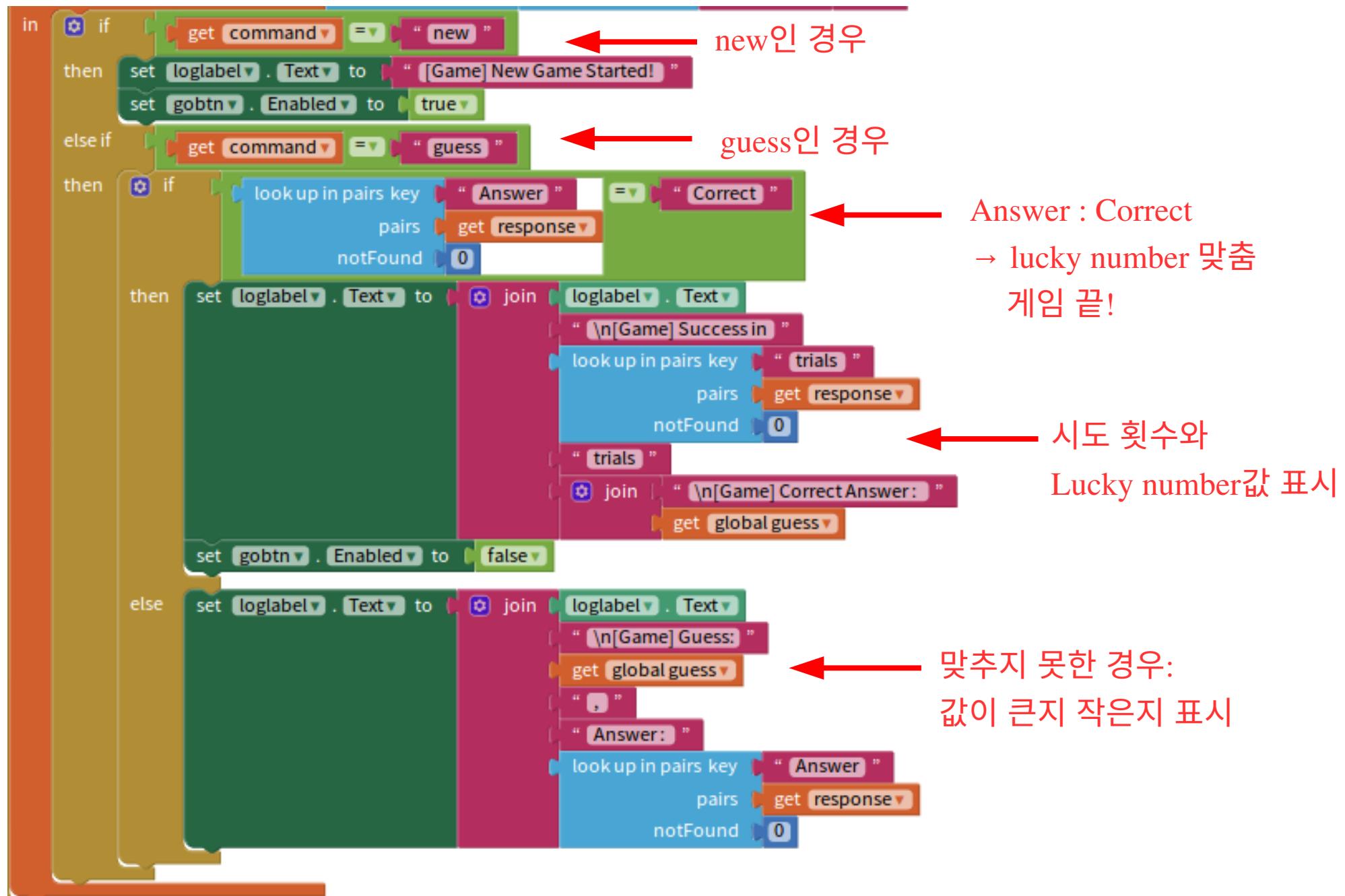
설계 과정



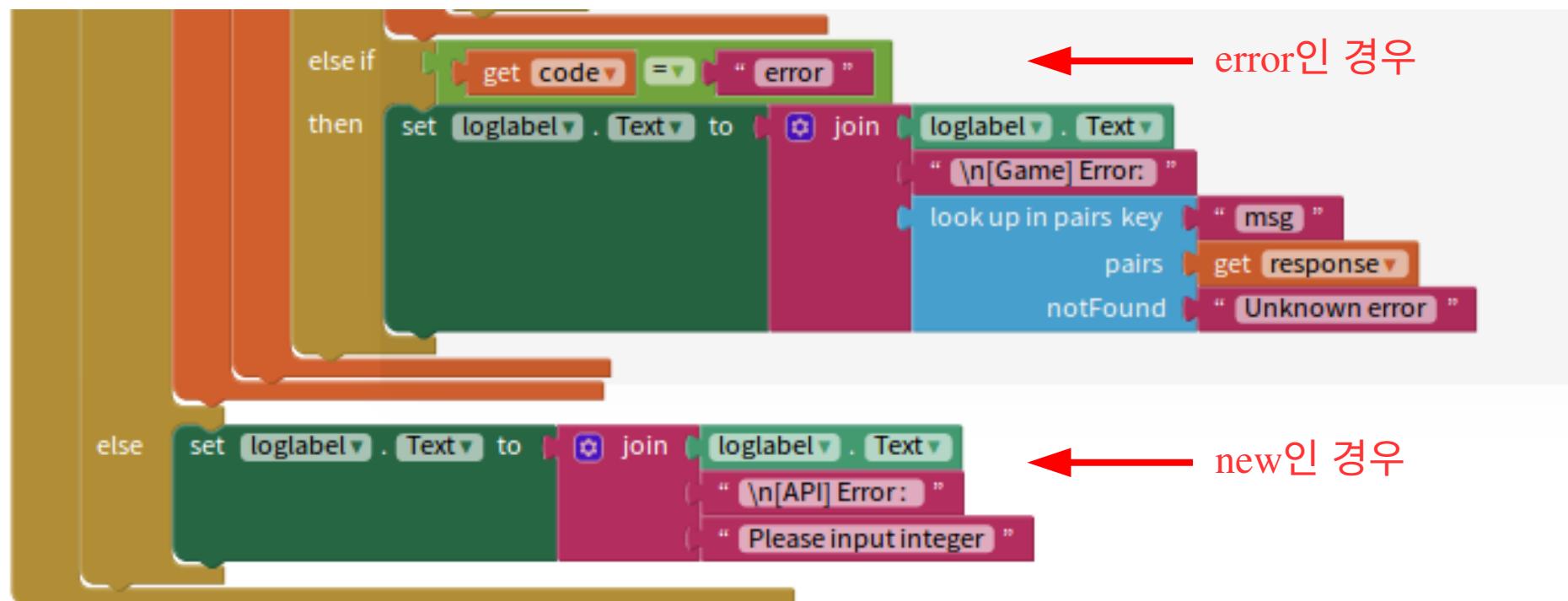
설계 과정



설계 과정



설계 과정



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

localhost의 터미널에서
curl을 이용해서 텍스트 기반으로 테스트

```
curl -d "max1=1000" http://172.30.1.45/game/new
{"code": "success"}hyeondin@hyeondin-ThinkPad-T47

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"}|
```

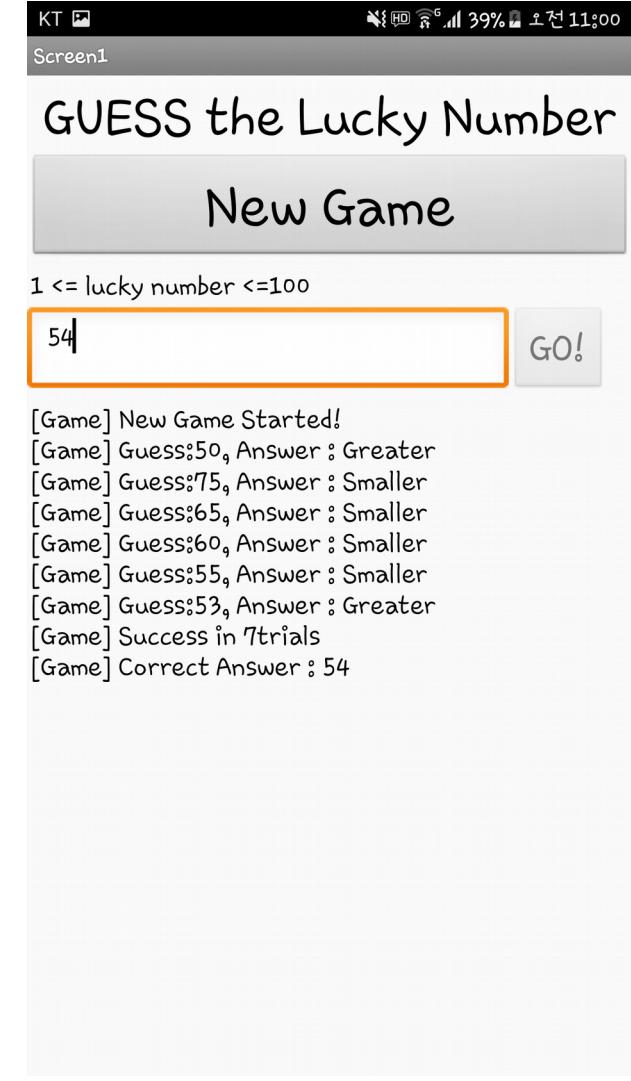
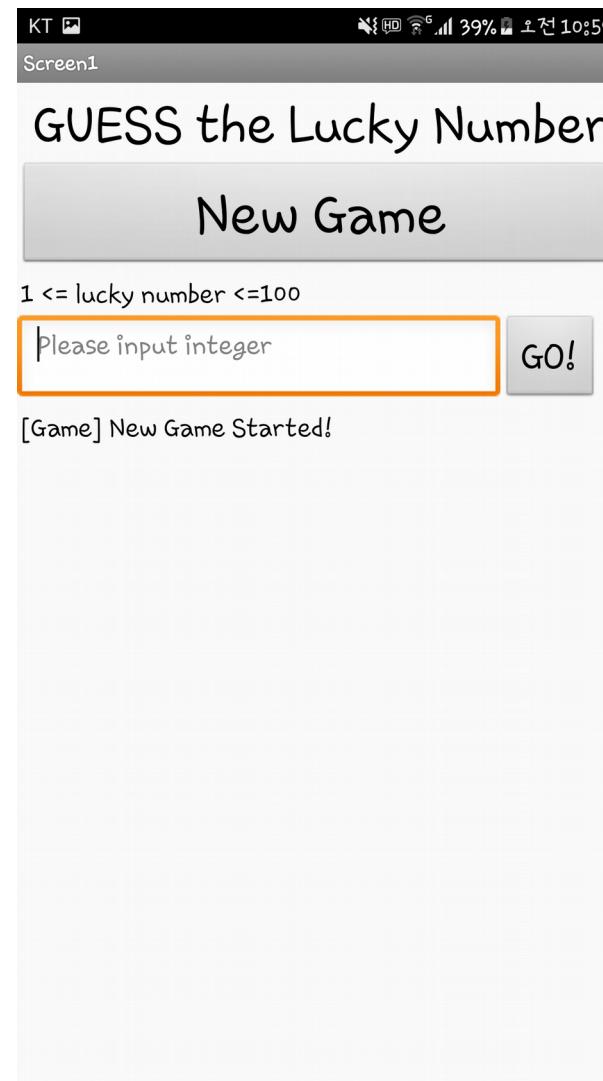
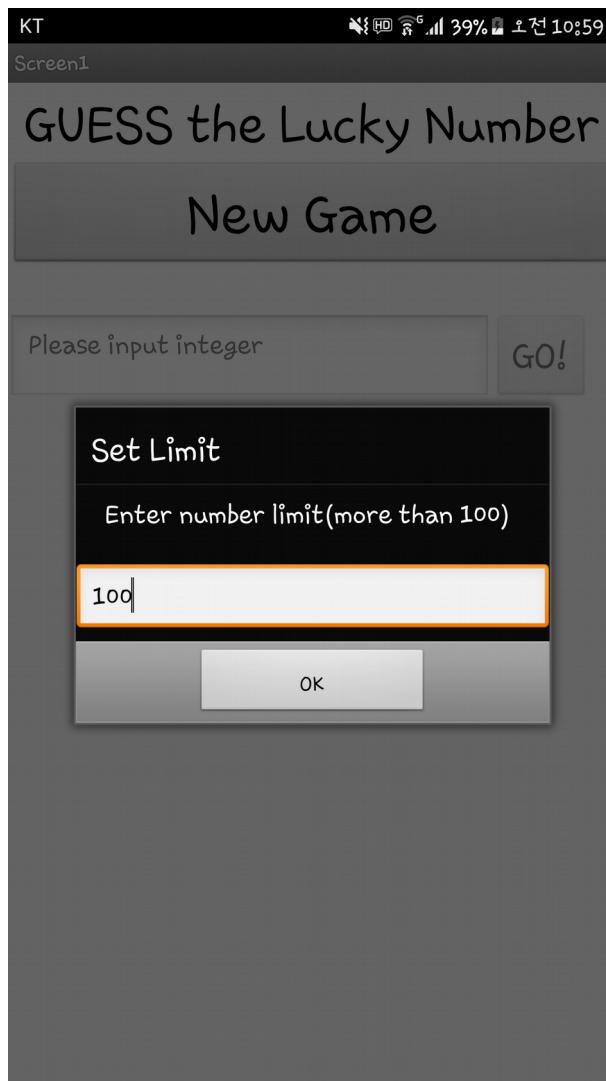
```
curl -d "guess=760" http://172.30.1.45/game/guess
{"Answer": "Greater", "trials": 13, "code": "success"}|
```

```
curl -d "guess=774" http://172.30.1.45/game/guess
{"Answer": "Correct", "trials": 18, "code": "success"}|
```

테스트 과정

시스템 통합 테스트(성공의 경우)

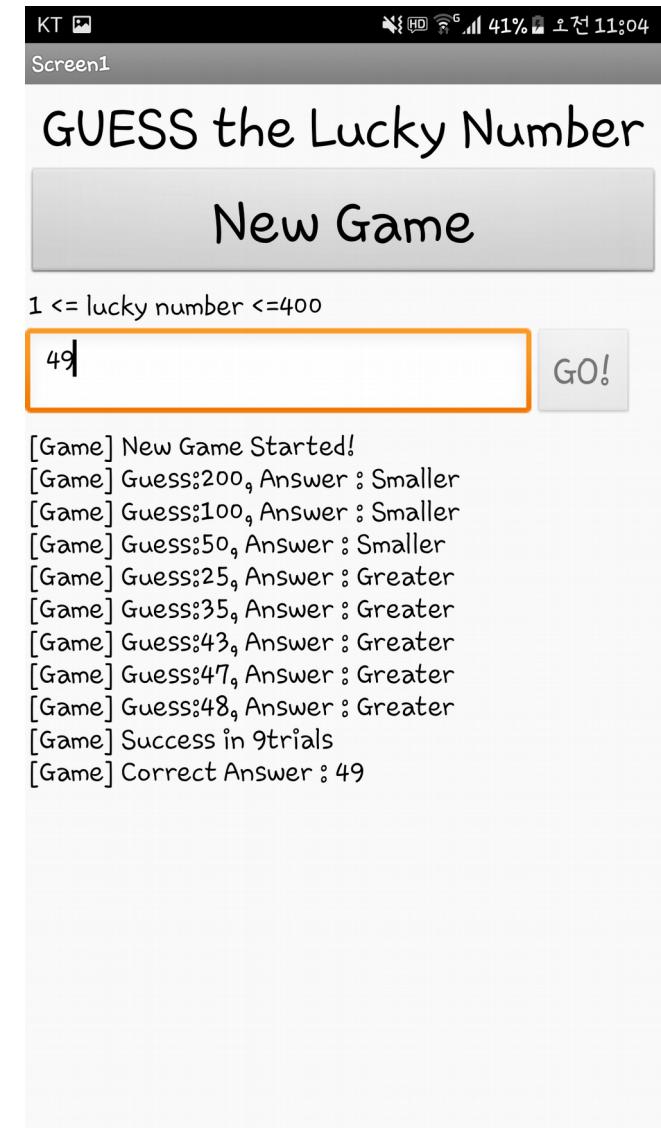
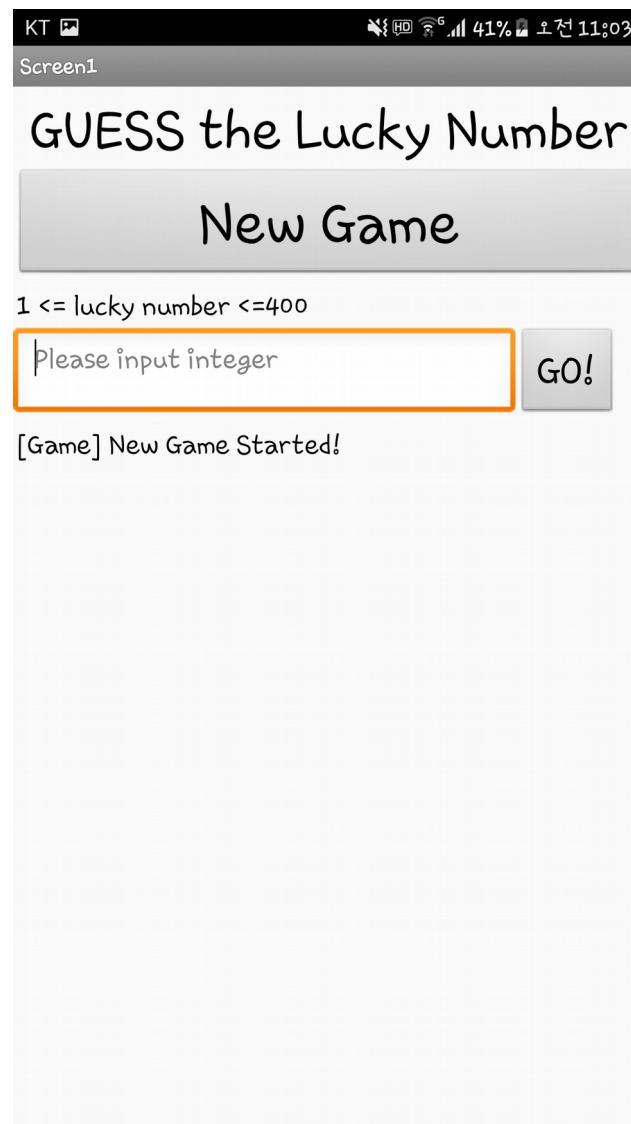
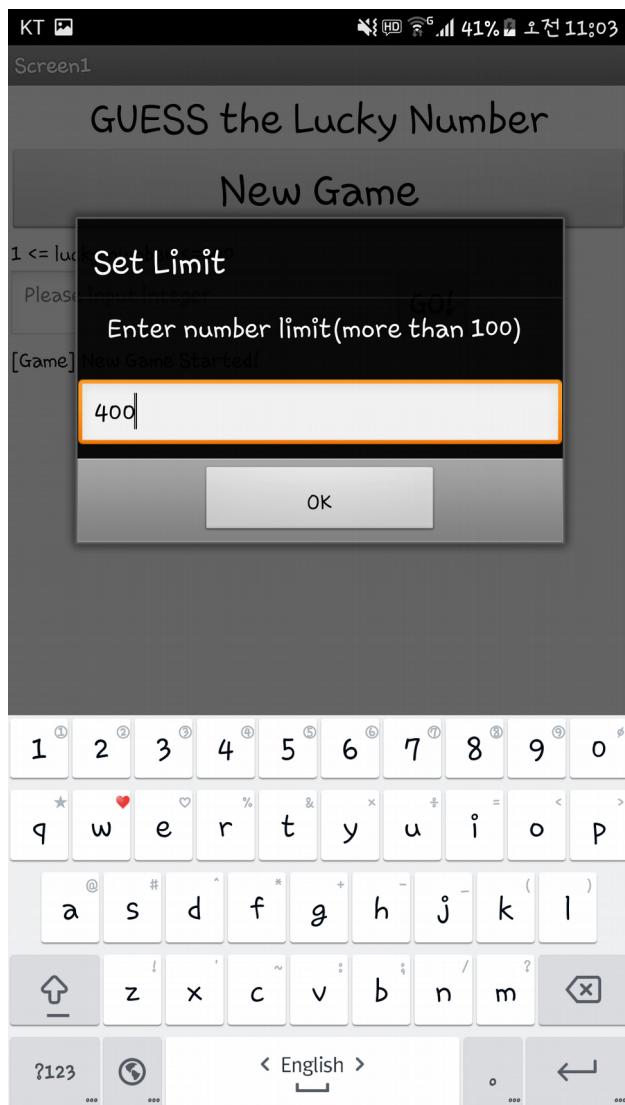
Number limit= 100



테스트 과정

시스템 통합 테스트(성공의 경우)

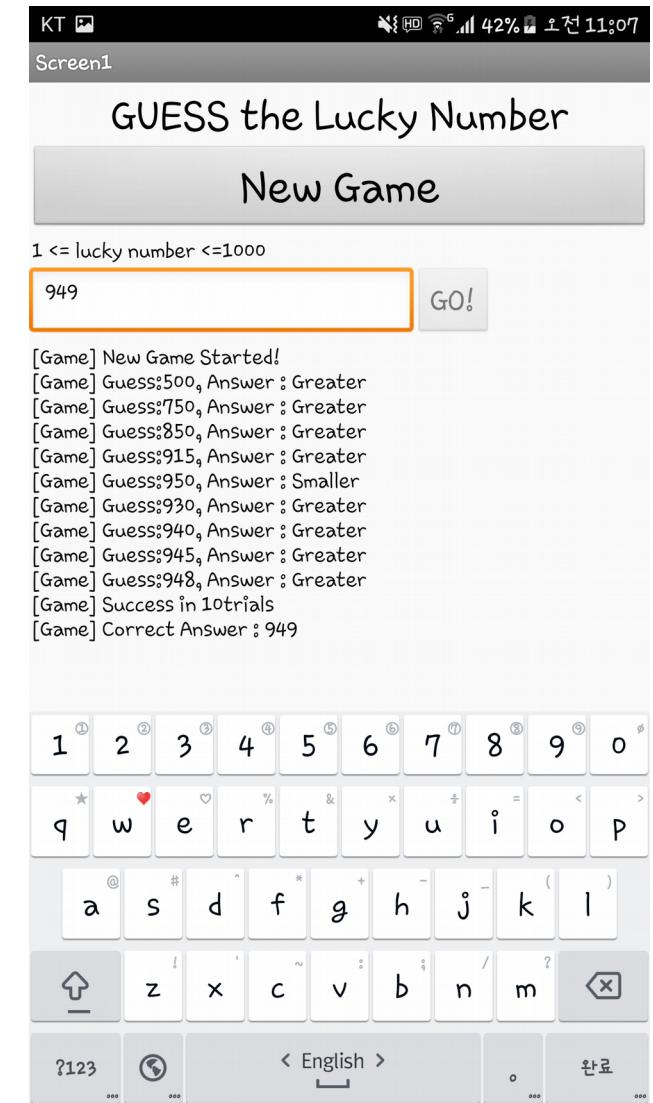
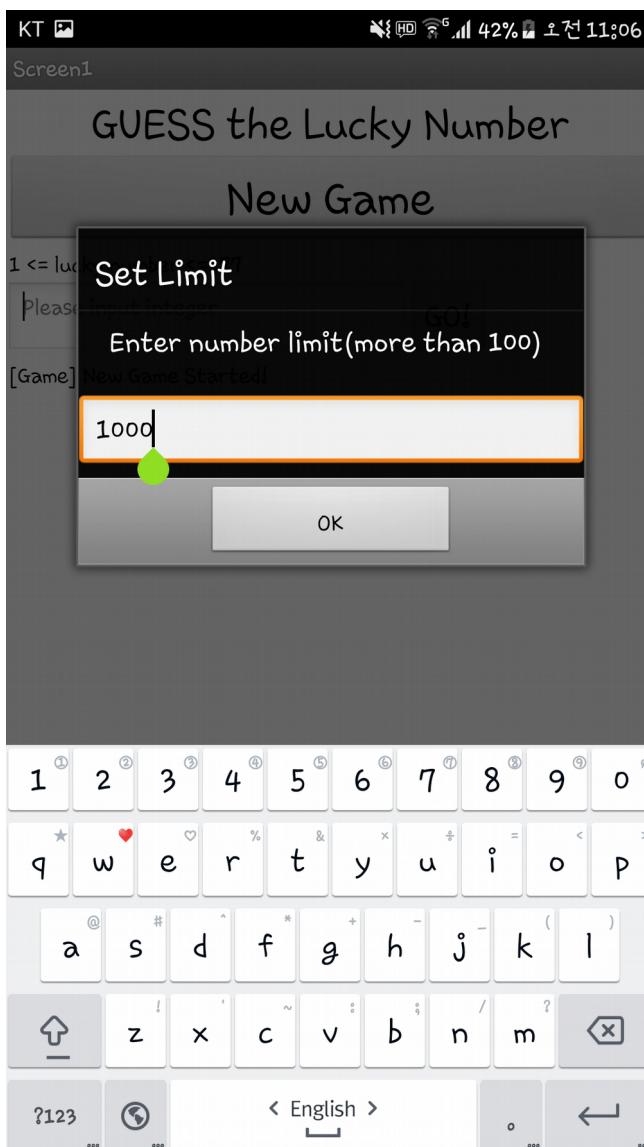
Number limit= 400



테스트 과정

시스템 통합 테스트(성공의 경우)

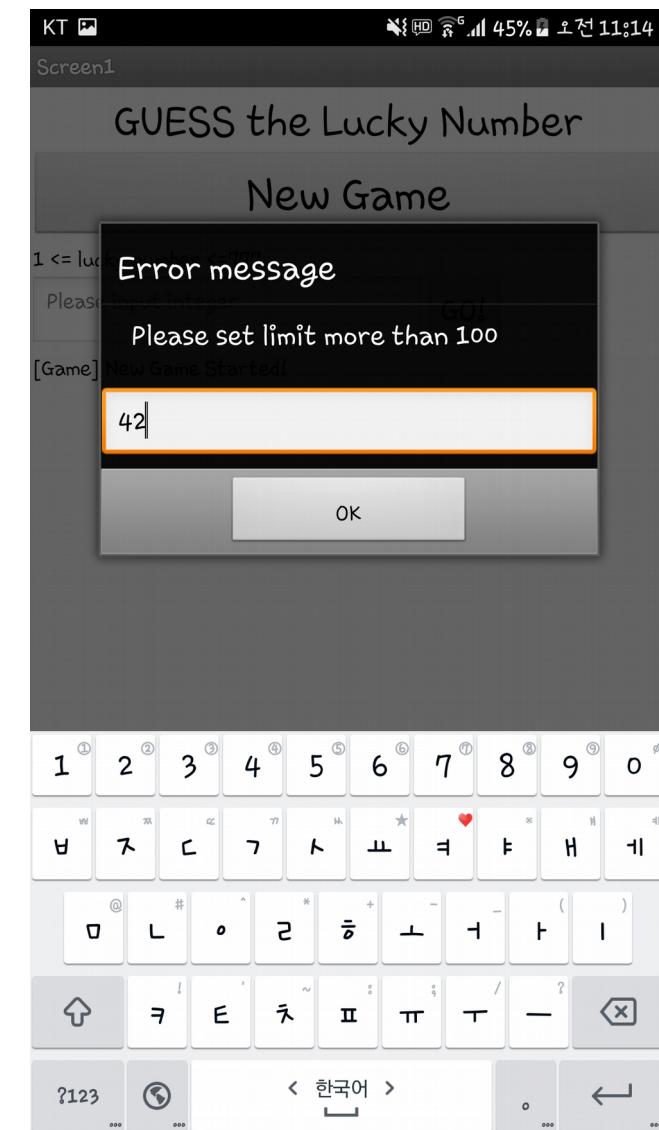
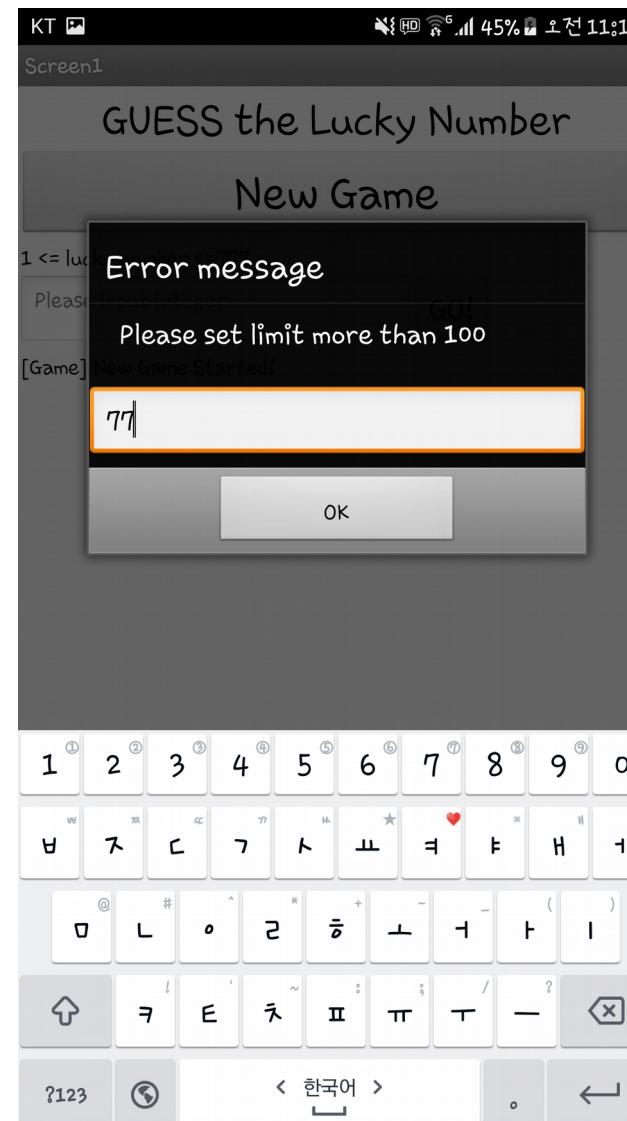
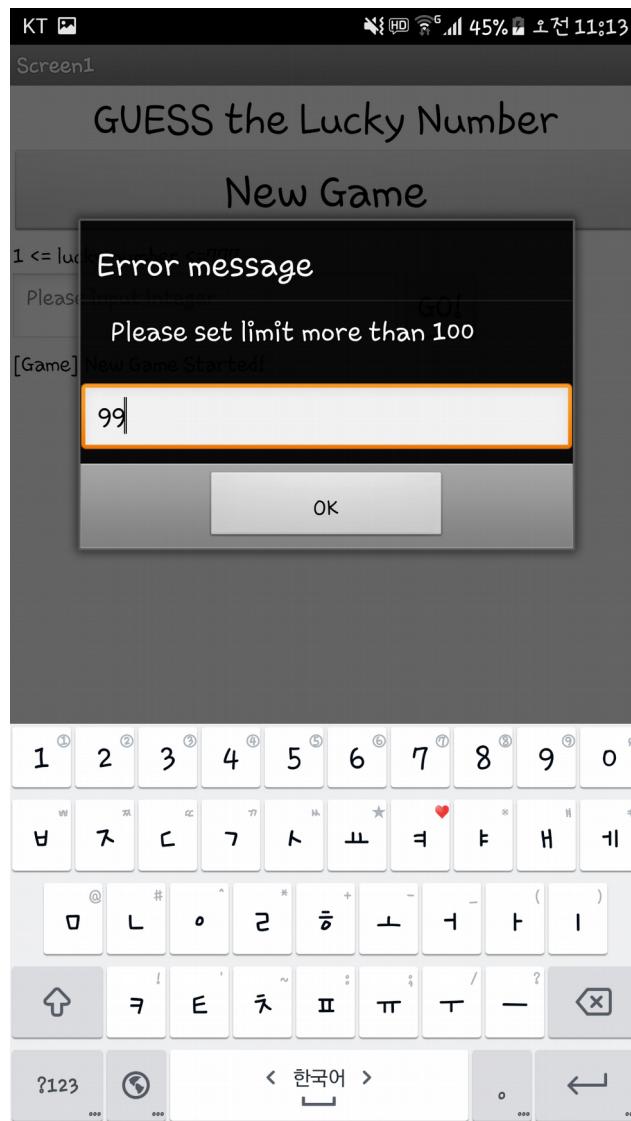
Number limit= 1000



테스트 과정

시스템 통합 테스트 – (Error 처리)

Number limit이 100보다 큰 수가 아니면 에러 메시지를 띄운다.



테스트 과정

시스템 통합 테스트 – (Error 처리)

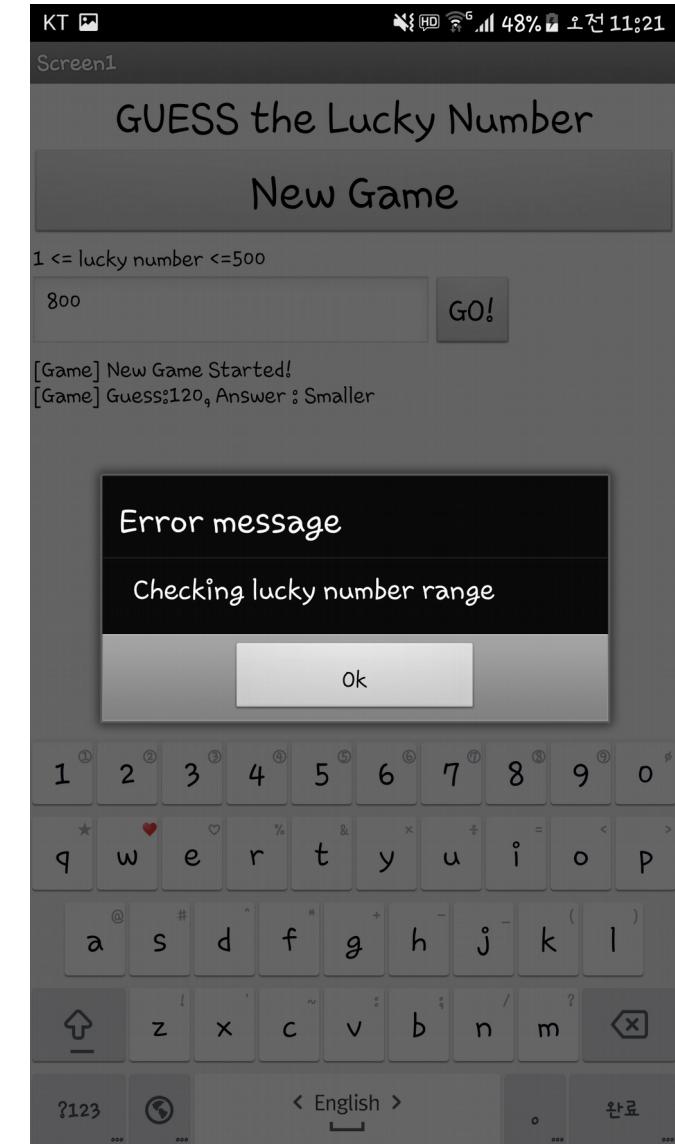
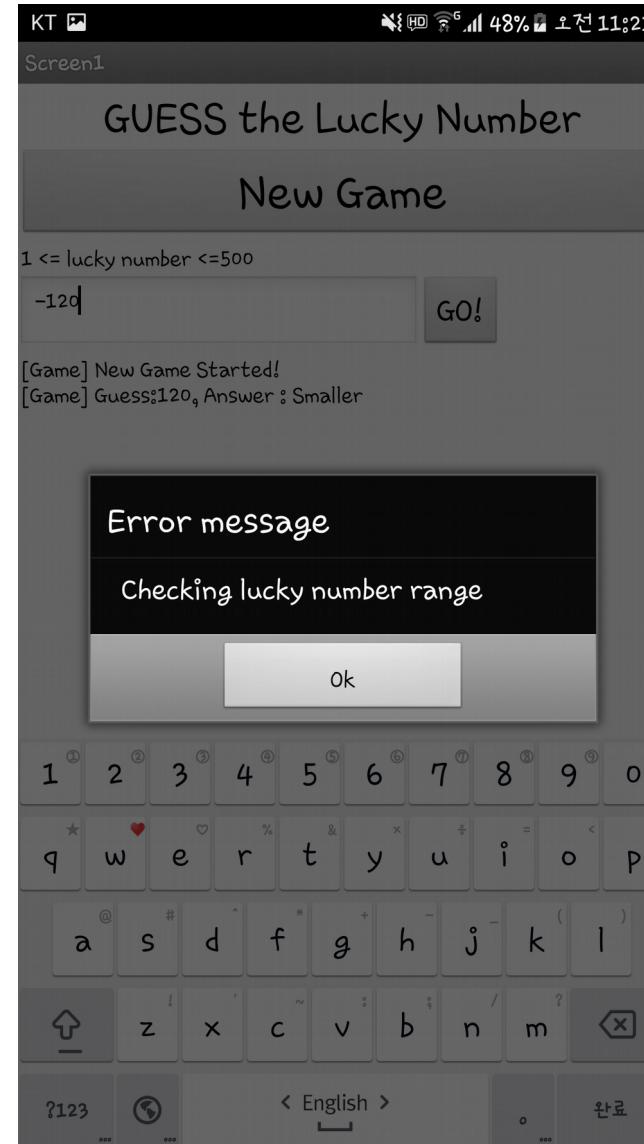
Guess number가 숫자가
아닌 문자인 경우.



테스트 과정

시스템 통합 테스트 – Error 처리

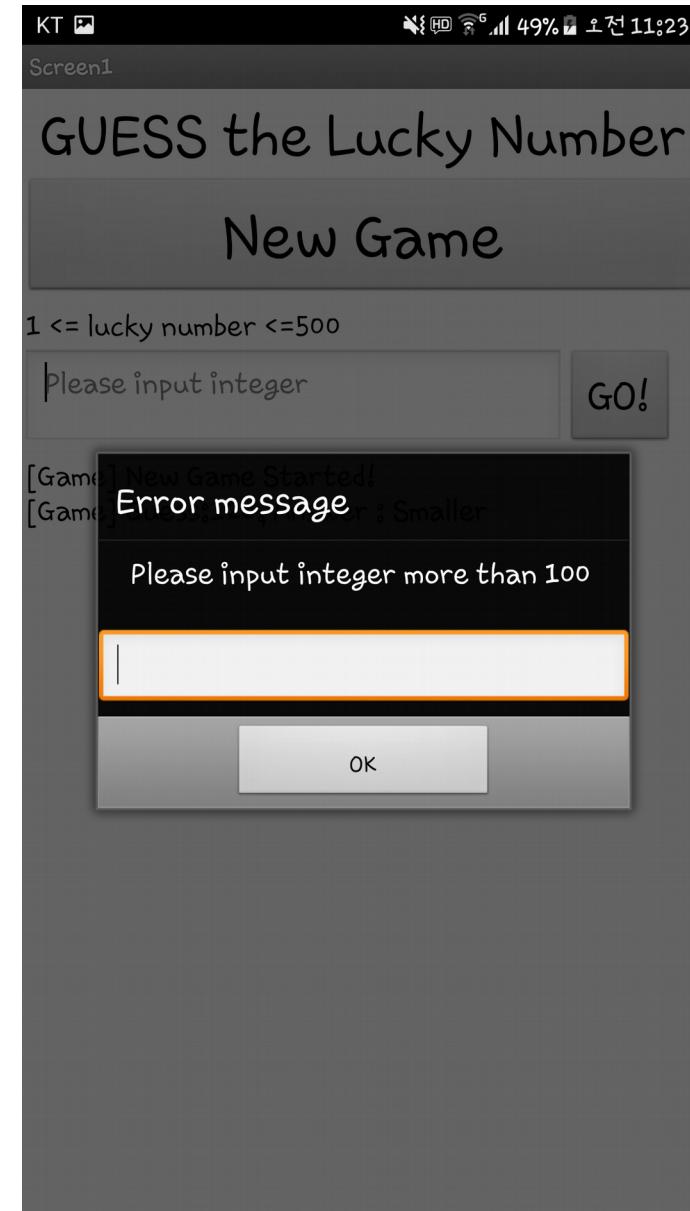
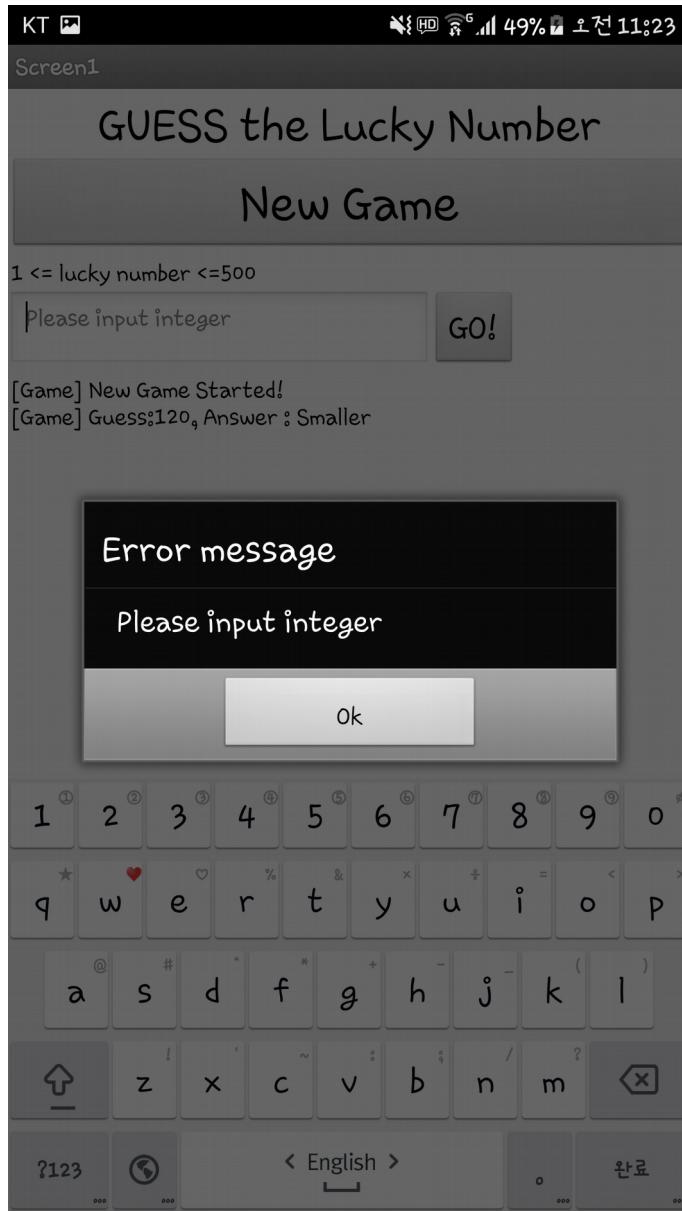
Guess number가 범위를 벗어난 경우



테스트 과정

시스템 통합 테스트 – (Error 처리)

아무것도 입력하지
않은 경우!



고찰 및 토의

- 고찰

1. 숫자의 범위를 정해주거나 최댓값을 정해줄 때, 빈칸인 경우를 어떻게 하면 좋을까 라고 생각하다가 길이를 이용하여 0인 경우 일 때를 처리해주니 에러 처리를 할 수 있었다.
2. 문자인지 아닌지를 판별할 때, 문자입니까? 라고 생각하고 블록 코딩을 만드는 것 보다 숫자가 아닙니까? 라고 생각하고 블록 코딩을 만드는 것이 더 효율적이었던 것 같다.
3. 게임의 핵심 Logic 설계 중, 추측 값이 luck number보다 큰지 작은지 판별할 때, 숫자를 인자로 넘겨주고 return 할 때는 큰지, 작은지를 알려주는 string 으로 반환하니 쉽게 문제를 해결할 수 있었다.
- 4. 범위를 벗어난 숫자의 입력을 서버 에러에서 처리하고 싶었지만 그러지 못한 점이 아쉽다.

고찰 및 토의

- 토의 : 잘 만들었는가?

1. 우리가 만드는 게임의 규칙을 잘 이해를 한 후, 게임을 만들어서 그런지 괜찮게 만든 것 같다.
2. 어떤 경우에 에러가 나야 하는지 정확히 파악 한 후 각각의 경우에 맞게 에러 처리를 해주었다.
3. 맞추기를 끝나면 결과를 보여주고 새로운 게임을 시작할 수 있도록 해준 것이 잘한 점 같다.
4. 서버에러를 많이 하지 못한 점이 아쉬운 점 인 것 같다.

감사합니다.

한 학기 동안 수고 많으셨습니다.