```
//2 주차

import UIKit

var str = "Hello, playground"

//상수와 변수----------------------------------------------------------------------------------------------------
var constant = "상수" //let(상수)로 하면 실행 안됨 var(변수)
var variable = "변수"

constant = "변수"
variable = "변수 2"

//문자열 보간법----------------------------------------------------------------------------------------------------
//System.out.println("나의 나이는 " + 30 + "살 입니다." (자바에서 사용하는 형식)
let message = "나의 나이는 ₩(false)살 입니다."
let age = 30 //숫자. age:Int 로 안쳐도 타입을 정수형으로 인식함

print(message) //출력은 dump 로도 가능
dump(age)

//컬렉션----------------------------------------------------------------------------------------------------
let bool = false
let intValue:UInt = 0 //UInt 은 0 부터 시작하는 양수
let msg:String = "" //문자열
let decimal:Float = 0.0 //실수

let arr:Array<Int> = [1, 2, 3, 4, 5] //int 형 배열

//name - 이름
var dic:Dictionary<String, String> = Dictionary<String, String>()

//값을 추가
dic["name"] = "chulsoo"

print(dic)

//값을 제거
dic.removeValue(forKey: "name")

print(dic)

//Set----------------------------------------------------------------------------------------------------
var intSet:Set<Int> = [1, 2, 3, 4, 5] //값이 무작위 출력
```

```
//값 추가
intSet.insert(66)

print(intSet)

//중복된 값을 삽입
intSet.insert(1) //똑같은 값을 삽입해도 한번만 출력됨

print(intSet)
intSet.remove(66)
print(intSet)

//반복문 조건문----------------------------------------------------------------------------------------------------
//반복문
let values:Array<Int> = [1, 2, 3, 4, 5]

for value in values {
    print(value)
}

//자바스크립트 문법 스위프트에서도 사용
values.forEach { (value:Int) in
    print(value)
}

//조건문----------------------------------------------------------------------------------------------------
//let values:Array<Int> = [1, 2, 3, 4, 5] 위에 let values 있어서 주석처리함

for value in values {
    if value % 2 == 0 {
        print(value)
    }
}

for value in values {
    switch value {
    case 1, 3:
        print("1 과 3")
    case 2, 4:
        print("2 와 4")
    default:
        print("5")
    }
```

```
}

//함수-----------------------------------------------------------------------------------------------------
func showAction(place:String, action:String = "공부") -> Void {print("나는 ₩(place)에서 ₩(action) 중입니다.")}

showAction(place: "수영장")
showAction(place: "수영장", action: "아크로바틱")

//옵셔널---------------------------------------------------------------------------------------------------
var intVal:Int? = 12

if let value:Int = intVal {
    print(value)
}



//2-2
//구조체(swift 대부분은 구조체 타입, 내부에 프로퍼티, 함수를 선언하고 인스턴스를 만들어서 사용)----------
------------------
import Foundation

struct Example {
    //가변 프로퍼티 var
    var mutableProperty: String = ""

    //불변 프로퍼티 let
    let immutableProperty: String = ""

    //인스턴스 메소드
    func showPrint() -> Void {
        print(mutableProperty) //위에 보면 값을 안넣었기 때문에 빈칸 출력
    }
}

//불변 인스턴스
let mutableInstance:Example = Example() //mutableInstance.mutableProperty = "변경" (에러발생)
mutableInstance.showPrint()

//가변 인스턴스
var immutableInstance:Example = Example()
immutableInstance.mutableProperty = "변경"
immutableInstance.showPrint()

//구조체 실습
```

```
struct Home {
    var type:String = "주택"

    var address:String = "부산시 사상구 주례동"

    func showHome() -> Void {
        print("이 곳은 ₩(type), 주소는 ₩(address) 입니다.")
    }
}

//가변
var home:Home = Home()
home.address = "부산시 사상구 근처 주택"
home.showHome()

//불변. 인스턴스가 불변이라 변경할 수 없음
/*let home1:Home = Home()
 home.address = "부산시 사상구 근처 주택"
 home.showHome()*/

//클래스 (구조체랑 같이 코드 적으면 오류나서 주석처리) (내부에 프로퍼티 또는 함수를 선언하고 인스턴스를 만들어서 사용. 다중상속 불가능)
/*import Foundation

class Example {
    //가변 프로퍼티
    var mutableProperty: String = ""
    //불변 프로퍼티
    let immutableProperty: String = ""
    //인스턴스 메소드
    func showPrint() -> Void {
        print("출력")
    }
}

//불변 인스턴스
 let mutableInstance.Example = Example()
 mutableInstance.mutableProperty = "변경"
 mutableInstance.showPrint()
 //가변 인스턴스
 var immutableInstance:Example = Example()
 immutableInsatance.mutableProperty = "변경"
 immutableInstance.showPrint()
 */
```

```
//열거형(유사한 종류의 여러 값을 한 곳에 모아 정의한 것. enum 자체가 하나의 데이터 타입. 각각의 case 는 그 자체가 고유의 값----------
import Foundation

enum Day {
    case sun, mon, tue, wed, thu, fri, sat
    /*enum Day:String {
        case sun = "일", mon = "월", tue = "화", wed = "수", thu = "목", fri = "금", sat = "토"*/ //이건 string 으로 글자 기본값 부여한 것
    func showToday() -> Void {
        switch self {
        case .sun:
            print("일요일")
        case .mon:
            print("월요일")
        case .tue:
            print("화요일")
        case .wed:
            print("수요일")
        case .thu:
            print("목요일")
        case .fri:
            print("금요일")
        case .sat:
            print("토요일")
        }
    }
}

let today:Day = .mon

switch  today {
case .sun, .sat:
    print("주말")
default:
    print("평일")
}
today.showToday()
//print(today.rawValue) 이건 string 으로 기본값 부여하고 enum 의 원시값 부여할때


//클로저(한번만 사용하는 일회용 함수, 실행가능한 코드블럭을 의미, 함수와 달리 클로저는 이름정의 필요 없음)----------
```

```
import   Foundation

let checkAudlt = { (age:Int) -> String in   // 변수선언-필요한 매개변수 기재-반환타입 기재
    return age < 20 ? "청소년" : "성인" // 필요한 반환값 기재
}

func checkAudlt(age:Int) -> String{   // 클로저를 함수로 변경한것.
    return age < 20 ? "청소년" : "성인"
}

print("이 고객은 ₩(checkAudlt(10))입니다")


//가드(if 문 대체, 조건 참일시 통과하고 거짓일시 else 실행 후 종료, else 내부에는 return 이나 break 무조건 있어야함 ----------------------
func showAudltWithGuard(age: Int?) -> Void { //age 는 매개변수

    guard let age:Int = age, age < 130, age >= 20 else {
        print("나이값이 잘못되었거나 성인이 아닙니다.")
        return
    }

 /*    if let age:Int = age {.              //guard 문 쓰는게 간결함
        if age < 130, age >= 20 {
            print("당신은 성인입니다.")
        } else {
            print("나이값이 잘못되었거나 성인이 아닙니다.")
        }
    }*/
    print("당신은 성인입니다.")
}

showAudltWithGuard(age: 100)


//상속(다중상속 지원 X, 클래스, 프로토콜에서 가능하며 열거형이나 구조체는 상속불가)-----------------------------
---------------------------
class Car {
    var type:String = "승용차"

    func showMyCar() -> Void {}
}

class Brand: Car {
```

```
    var brand:String = "메르세데스 벤츠"

    override func showMyCar() {
        print("나의 차는 ₩(type), 브랜드는 ₩(brand) 입니다.")
    }
}

let brand:Brand = Brand()
brand.showMyCar()

//프로토콜(특정 작업 또는 기능에 적합한 메소드, 속성 및 기타 요구사항의 청사진, 자바의 interface 와 유사, 객체 혹은 인터페이스 대신 프로토콜 중심으로 설계)
import Foundation

@objc protocol SchoolProtocol {
    @objc optional var name:String { get }

    //필수 정의 함수
    func showSchoolName() -> Void

    //선택 정의 함수
    @objc optional func setSchoolName(name: String) -> Void
}

//프로토콜 상속
class School: SchoolProtocol {
    func showSchoolName() {
        print("동서대학교")
    }

    // optional 로 정의할 경우 함수를 호출하지 않아도 됨
}

let school:School = School()
school.showSchoolName()



// 3 주차

// UIKit
// -> iOS 또는 tvOS 앱을 위한 그래픽 기반의 UI 를 구성하고 관리하는 프레임워크
// -> 제스쳐, 애니메이션, 그림 그리기, 이미지 처리 등 다양한 사용자 이벤트들을 처리
// -> 화면을 구성하기 위해 필수적으로 상속해야 함
```

// View
// -> iOS 어플리케이션 화면에서 보는 내용은 윈도우와 뷰를 사용하여 나타냄
// -> UIView 클래스나 UIView 의 하위 클래스의 인스턴스로 윈도우의 한 영역에서 콘텐츠 표시
// -> 제스쳐 혹은 터치 이벤트 처리

// 뷰의 계층구조와 서브 뷰
// -> 다른 뷰를 위한 컨테이너로써의 역할도 병행
// -> 자식 뷰는 서브 뷰, 부모 뷰는 슈퍼 뷰로 불림
// -> 슈퍼 뷰와 서브 뷰 관계에 있을 경우 슈퍼 뷰가 서브 뷰에 가려짐
// -> 하나의 슈퍼 뷰에 두 개 이상의 서브 뷰가 겹치게 된다면 나중에 추가된 서브 뷰가 맨 위에 보여짐

// 오토레이아웃
// -> 안드로이드의 Constraint layout 과 유사
// -> 다양한 디바이스 크기에 대응하기 위해 사용
// -> 뷰에 주어진 조건에 따라 동적으로 계산하여 크기 조절하여 내, 외부 변화에 동적으로 반응

// 오토레이아웃의 사용 목적
// -> 디바이스의 크기가 다양한 경우
// -> 디바이스를 회전할 경우
// -> 지역화(다국어)를 지원하는 경우
// -> 콘텐츠가 동적으로 보여지는 경우
// -> 상태표시줄에 녹음 시 나타나는 오디오바와 전화 중에 나타나는 액티프 콜이 보여지거나 사라질 경우

// Constant 와 Multiplier
// Constant : 뷰와 레이아웃의 간격(값)
// Multiplier : 뷰와 레이아웃간의 비율(%)

// 안전영역(Safe Area)
// -> 새로운 디바이스(노치)가 등장하면서 새로 생긴 개념
// -> 콘텐츠가 상태바, 네비게이션바, 툴바, 탭바를 가리는 것을 방지하는 영역

// 기존의 레이아웃 vs 안전영역
// -> SafeArea 없이 사용할 경우 SuperView 를 가리키므로 탭바, 네비게이션 바 포함되므로 노치가 적용된 기기에 대응하기 힘듦

// 프로그래밍으로 제어한다면?
// -> 스토리보드에서 해당 Constraint 객체를 코드와 연결
// -> Constant 혹은 multiplier 값을 변경

// 오토레이아웃 - 인터페이스 빌더
// 1. 중앙 이미지 클릭 후 스토리보드 하단 Add new Constraints 버튼 클릭
// 2. 가로(width), 세로(height) 150 후 Add -> 빨간색 줄 나오는데 잘 된 것
// 3. 하단 차트모양 버튼 클릭 후 x 축 y 축 정중앙 체크 후 Add -> 가운데로 위치하게 됨
// 4. 스택 뷰 클릭

// 5. 윗쪽 선 해제 후 왼쪽 20, 오른쪽 20, 하단 20 설정 -> 상단 거리 제외한 하단 중앙에 위치하게 됨

```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var imageView: UIImageView!
    @IBOutlet weak var stackView: UIStackView!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }
}
```

//4 주차
/* UIKit 과 View

<UIKit>

IOS 또는 tvOS 앱을 위한 그래픽 기반의 UI 를 구성하고 관리하는 프레임워크

제스처, 애니메이션, 그림그리기, 이미지 처리 등 사용자 이벤트들을 처리

화면을 구성하기 위해 필수적으로 상속해야함

<View>

IOS 어플 화면에서 보는 내용은 윈도우와 뷰를 사용하여 나타냄

UIView 클래스나 UIView 의 하위클래스의 인스턴스로 윈도우의 한 영역에서 콘텐츠 표시

제스터 혹은 터치 이벤트 처리
——

뷰의 계층구조와 서브뷰
- 다른 뷰를 위한 컨테이너로써의 역할도 병행
- 자식 뷰는 서브 뷰, 부모 뷰는 슈퍼 뷰로 불림
- 슈퍼뷰와 서브뷰 관계에 있을 경우 슈퍼뷰가 서브뷰에 가려짐
- 하나의 슈퍼뷰에 2 개 이상의 서브뷰가 겹치게 된다면 나중에 추가된 서브뷰가 맨 위에 보여짐

뷰 컨트롤러
- UIKit 을 사용하는 앱의 인터페이스를 관리하기 위한 도구

- 뷰컨트롤러는 하나의 루트 뷰 만을 관리하고 해당루트 뷰가 여러개의 서브 뷰를 가지는 방식으로 구성

```
 종류 - 네비게이션 , 탭 바, 테이블 뷰, 그 외 여러 컨트롤러 존재
 ————————————————————————————————————————————————————
 오토레이아웃

 - 안드로이드의 Constraint layout 과 유사

 - 다양한 디바이스 크기에 대응하기 위해 사용

 - 뷰에 주어진 조건에 따라 동적으로 계산하여 크기 조절하여 내, 외부 변화에 동적으로 반응

 - 오토레이아웃 사용목적

 - 디바이스의 크기가 다양한 경우

 - 디바이스를 회전할 경우

 - 지역화(다국어)를 지원하는 경우

 - 콘텐츠가 동적으로 보여지는 경우

 - 상태표시줄에 녹음 시 나타나는 오디오바와 전화 중에 나타나는 액티브 콜이 보여지거나 사라질 경우
————————————————————————————————————————————————————
 Constant 와 Multiplier

 Constant = 뷰와 레이아웃의 간격(값)

 Multiplier = 뷰와 레이아웃간의 비율(%)
 ————————————————————————————————————————————————————
*/
// <오토레이아웃 코드>

import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var imageView: UIImageView!
    @IBOutlet weak var stackView: UIStackView!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.

        imageView.translatesAutoresizingMaskIntoConstraints = false
```

```
        stackView.translatesAutoresizingMaskIntoConstraints = false

        NSLayoutConstraint.activate([
            imageView.centerXAnchor.constraint(equalTo: self.view.centerXAnchor),
            imageView.centerYAnchor.constraint(equalTo: self.view.centerYAnchor),
            imageView.widthAnchor.constraint(equalToConstant: 150),
            imageView.heightAnchor.constraint(equalToConstant: 150),

            stackView.bottomAnchor.constraint(equalTo: self.view.safeAreaLayoutGuide.bottomAnchor, constant: -
20),
            stackView.leadingAnchor.constraint(equalTo: self.view.safeAreaLayoutGuide.leadingAnchor, constant:
20),
            stackView.trailingAnchor.constraint(equalTo: self.view.safeAreaLayoutGuide.trailingAnchor, constant: -
20)
        ])
    }


}
//로그인 화면 - 이미지뷰, 에셋

/*

 <UIImageView>

 - 안드로이드의 ImageView 와 같은 기능

 - 이미지를 넣기 위한 뷰

 <이미지 표시 방법>

 - Scale To Fill = 이미지 비율을 무시하고 그림을 넣음
 - Aspect Fit = 이미지 비율을 유지한 채로 이미지 삽입, 비율이 맞지 않을 경우 내부에 여백이 발생
 - Aspect Fill = 이미지 비율을 유지한 채로 이미지 삽입하지만 그림이 잘리게 됨,
                 Clips to Bounds 를 Yes 로 하면 그림이 잘려서 나오지 않게 됨
                 인물, 동물 사진에는 적합하지 않음
 - Redraw  = 설명없노~~~~

————————————————————————————————————————————————————————
———

 <Assets (에셋)>
 - 안드로이드의 Drawable/Raw 과 유사
 - 이미지 파일, mp3, json 파일 등 다양한 파일을 삽입을 할 수 있음
```

```
 */




// 4 주차

// 1. UIAlertController 에 대한 이해

// UIAlertController
//-안드로이드의 Alert Dialog 와 유사
//-경고, 확인 창을 표시 할 때 사용
//-알림 스타일은 actionSheet, 경고 2 가지가 있음



// UIAlertController 함수
// title = 경고창 제목
// message = 경고창에 표시 할 내용
// action = 경고 혹은 액션 시트에 추가 할 액션
// preferredStyle = 경고창 스타일 (alert, actionSheet 중 택 1)



// ActionSheet
//-두 개 이상의 선택 사항이 제공되는 경고 스타일
//-하단에서 위로 탑짐

// 경고
//-일반적인 경고 스타일
//-중요한 작업을하기 전에 표시

// UIAlertAction
//-사용자가 경고 또는 액션 시트에서 사용할 버튼과 버튼 구성을 클릭했을 때 수행 할 작업

// UIAlertAction 활용
// title = 액션 제목
// 스타일 = 액션 스타일
// isEnabled = 액션 사용 가능 여부

// UIAlertAction 스타일
// 기본 = 기본 스타일
// cancel = 작업을 취소하거나 변경하지 않은 경우
// 파괴적 = 데이터가 변경되거나 삭제되어 돌이킬 수없는 경우
```

```
// ---------------------------- 실습 ------------------ -----------

UIKit 가져 오기

class ViewController : UIViewController {

    override func viewDidLoad () {
        super.viewDidLoad ()
        // 뷰를로드 한 후 추가 설정을 수행합니다.
    }

    @IBAction func touchUpAlertButton (_ 발신자 : 모두) {
        let alertController = UIAlertController (title : "삭제", message : "정말 삭제 하시겠습니까?", preferredStyle :
.alert)
        let deleteAction = UIAlertAction (title : "삭제", 스타일 : .destructive) {
            (action : UIAlertAction)
            print ( "삭제")
        }
        let cancelAction = UIAlertAction (title : "취소", style : .cancel) {(action : UIAlertAction) in
            print ( "취소")
        }
        alertController.addAction (deleteAction)
        alertController.addAction (cancelAction)

        self.present (alertController, animated : true, 완료 : nil)
    }

    @IBAction func touchUpActionSheetButton (_ 발신자 : 모두) {
        let alertController = UIAlertController (title : "삭제", message : "정말 삭제 하시겠습니까?", preferredStyle :
.actionSheet)
        let deleteAction = UIAlertAction (title : "삭제", 스타일 : .destructive) {
            (action : UIAlertAction)
            print ( "삭제")
        }
        let cancelAction = UIAlertAction (title : "취소", style : .cancel) {(action : UIAlertAction) in
            print ( "취소")
        }
        alertController.addAction (deleteAction)
        alertController.addAction (cancelAction)

        self.present (alertController, animated : true, 완료 : nil)
    }
}
```

```
// 5 주차-위치 권한 실습 ---------------------------------------- ----------------------
UIKit 가져 오기
CoreLocation 가져 오기

class ViewController : UIViewController {
    // 버튼을 눌렀을 때 권한을 요청하고 값에 대한 결과 표시를위한 코드 (label 연결 설정 (알 수 없음 라벨))
    @IBOutlet weak var resultStatusLabel : UILabel!

    private let locationManager = CLLocationManager ()

    override func viewDidLoad () {
        super.viewDidLoad ()
        // 뷰를로드 한 후 추가 설정을 수행합니다.
    }
    // 버튼 연결 설정 (위치 권한 요청 버튼)
    @IBAction func touchUpReqPermissionButton (_ sender : Any) {
        // IOS 14 이상
        let status = locationManager.authorizationStatus
        // IOS 14 미만
        // let status = CLLocationManager.authorizationStatus ()

        스위치 상태 {
        case .authorizedWhenInUse, .authorizedAlways :
            print ( "허용")
            resultStatusLabel.text = "허용"// 권한 선택 상태 표시
        case .denied :
            print ( "권한 거절")
            resultStatusLabel.text = "권한 거절"
        case .notDetermined :
            print ( "알 수 없음 / 권한 미선택") // 위치 권한 선택 사용자에게 요청
            locationManager.requestWhenInUseAuthorization ()
            resultStatusLabel.text = "알 수 없음 / 권한 미선택"
        기본:
            print ( "권한 차단")
            resultStatusLabel.text = "권한 차단"
        }
    }

}
/ * 위 코드 작성 후 "NSLocationWhenInUseUsageDescription"관련 에러 발생시
command + shift + o 후 info.plist 검색
아무 파일에서 + 버튼 다운 후 Privacy-위치 사용시 용도 설명 등록, 값 값은 "위치 권환이 필요합니다"
 * /
```

// 5 주차-CocoPods 을 이용한 위치 표시 앱 실습 ---------- ----------
/ * 시작 전 google 에서 "google map ios sdk"검색, 시작하기 클릭, SDK 설치 진행
터미널에서 프로젝트 파일 경로에서
sudo gem install cocoapods (에러 발생시 아래 코드 입력)
sudo gem install cocoapods --source http://rubygems.org

이후 프로젝트 파일 클릭, File-new-File, other 란에 Empty 생성, Podfile 이라고 명명
Podfile 에
소스 'https://github.com/CocoaPods/Specs.git'
target 'YOUR_APPLICATION_TARGET_NAME_HERE'do <---- 프로젝트 이름으로 변경 배합 함
포드 'GoogleMaps', '4.2.0'
'GooglePlaces', '4.2.0'포드
종료
작성,
이후 터미널에서 포드 설치
API 키 소유,
이후

AppDelegate.swift 에서
GooglePlaces 작성 가져 오기,
func application () 함수 [UIApplication.LaunchOptionsKey : Any] 아래에
GMSPlacesClient.provideAPIKey ("얻어낸 API 키") 작성
이후에 ViewController.swift 로 다시 넘어서
viewDidLoad () 함수 내 super.viewDidLoad () 아래에
---------- ----------
let camera = GMSCameraPosition.camera (withLatitude : -33.86, 경도 : 151.20, 확대 / 축소 : 6.0)
let mapView = GMSMapView.map (withFrame : self.view.frame, camera : camera)
self.view.addSubview (mapView)

//지도 중앙에 마커를 만듭니다.
let marker = GMSMarker ()
marker.position = CLLocationCoordinate2D (위도 : -33.86, 경도 : 151.20)
marker.title = "시드니"
marker.snippet = "호주"
marker.map = mapView
---------- ----------
작성,
작성 후 에러 발생시 상단에 import GoogleMaps 작성

이후 프로젝트 폴더에서 info.plist 오른쪽 클릭, Open As-> Source Code 클릭
소스 파일 맨 밑 </ dict> 위에
<key> LSApplicationQueriesSchemes </ key>
<배열>

<string> googlechromes </ string>
<string> comgooglemaps </ string>
</ 배열>
삽입

하고 나서 GMSServices provideAPIKey 에러가 뜨면 실행
AppDelegate.swift 로 이동하여
GoogleMaps 작성 가져 오기,
GMSPlacesClient.provideAPIKey 부분 아래에
GMSServices.provideAPIkey ("얻어낸 API 키") 작성

위치 표시가 제대로 안되면
viewController.swift 파일에서
viewDidLoad () 함수 안에
let lat = 35.144792901569005
let lng = 129.01075261903821 작성,

iOS 용 Map SDK 라이브러리 사용하기 설정 (CocoaPods 를 이용한 위치 표시 앱 실습 18:25 참고)

완성 코드 ---------- * 표시는 실습으로 추가 된 코드 ---------- ----------
---------- ViewController.swift ----------
UIKit 가져 오기
CoreLocation 가져 오기
GooglePlaces 가져 오기 ***
GoogleMaps 가져 오기 ***

```
class ViewController : UIViewController {

    @IBOutlet weak var resultStatusLabel : UILabel!

    private let locationManager = CLLocationManager ()

    override func viewDidLoad () {
        super.viewDidLoad ()
*****
        let lat = 35.144792901569005
        lng = 129.01075261903821 하자

        let camera = GMSCameraPosition.camera (withLatitude : lat, longitude : lng, zoom : 6.0)
        let mapView = GMSMapView.map (withFrame : self.view.frame, camera : camera)
        self.view.addSubview (mapView)

        //지도 중앙에 마커를 만듭니다.
        let marker = GMSMarker ()
```

```
        marker.position = CLLocationCoordinate2D (위도 : 위도, 경도 : lng)
        marker.title = "동서 대학교"
        marker.snippet = "iOS 소프트웨어 수업"
        marker.map = mapView
*****
    }

    @IBAction func touchUpReqPermissionButton (_ sender : Any) {
        // IOS 14 이상
        let status = locationManager.authorizationStatus
        // IOS 14 미만
        // let status = CLLocationManager.authorizationStatus ()

        스위치 상태 {
        case .authorizedWhenInUse, .authorizedAlways :
            print ( "허용")
            resultStatusLabel.text = "허용"// 권한 선택 상태 표시
        case .denied :
            print ( "권한 거절")
            resultStatusLabel.text = "권한 거절"
        case .notDetermined :
            print ( "알 수 없음 / 권한 미선택") // 위치 권한 선택 사용자에게 요청
            locationManager.requestWhenInUseAuthorization ()
            resultStatusLabel.text = "알 수 없음 / 권한 미선택"
        기본:
            print ( "권한 차단")
            resultStatusLabel.text = "권한 차단"
        }
    }

}

---------- AppDelegate.swift ----------
UIKit 가져 오기
CoreData 가져 오기
GooglePlaces 가져 오기
GoogleMaps 가져 오기

@본관
class AppDelegate : UIResponder, UIApplicationDelegate {
```

```
        func application (_ application : UIApplication, didFinishLaunchingWithOptions launchOptions :
[UIApplication.LaunchOptionsKey : Any]?)-> Bool {
            // 응용 프로그램 시작 후 사용자 지정 지점을 재정의합니다.
            GMSPlacesClient.provideAPIkey ( "얻어낸 API 키")
            GMSServices.provideAPIKey ( "얻어낸 API 키")

            참을 반환
        }
 이후 코드 쭈르르 륵
 * /
```