프로그래밍뷰

학습목표

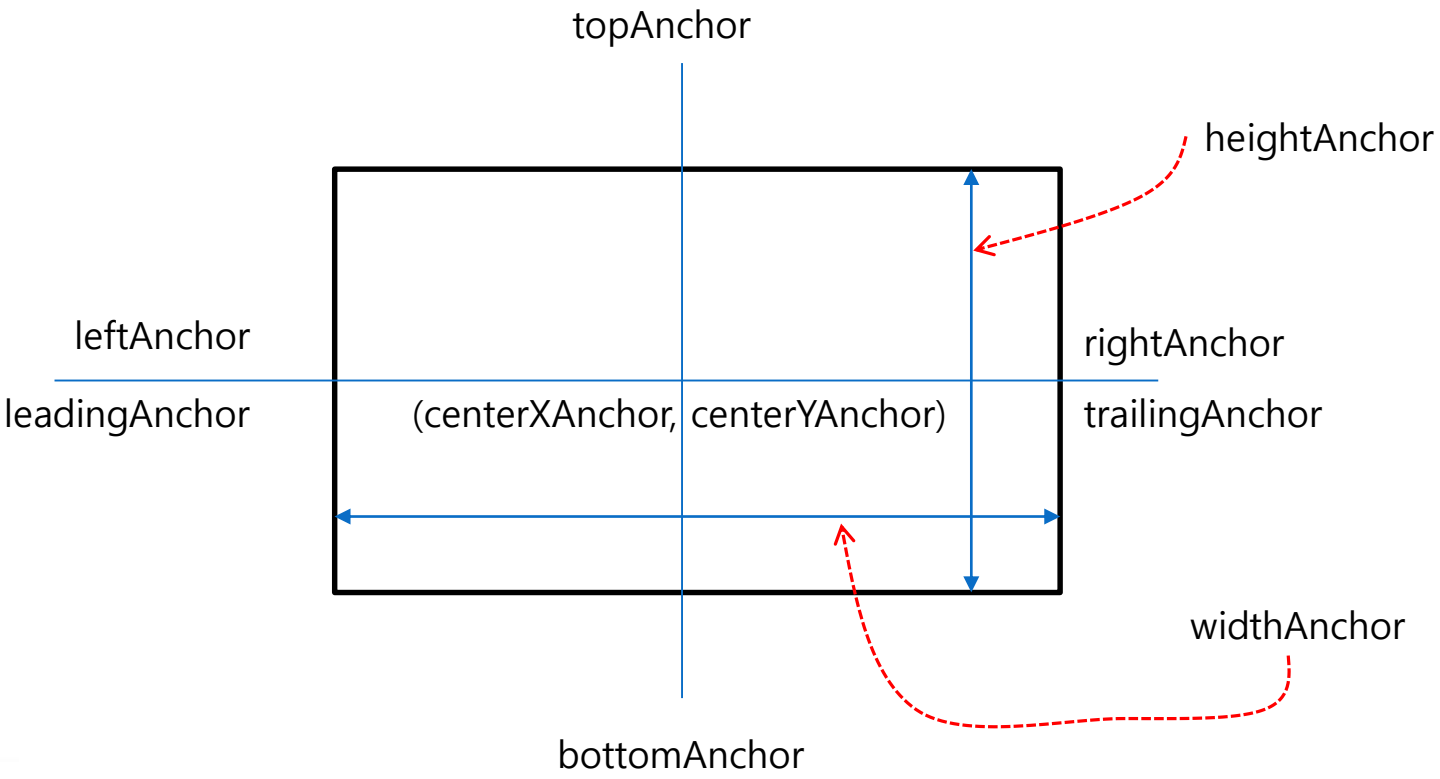
- 프로그래밍으로 View를 배치할 수 있음을 이해한다
- safeLayoutGuide의 존재를 이해한다.
- 10가지 Anchor를 이해하고 활용할 수 있다.
- LayoutGuide와 10가지 Anchor를 이용하여 Interface Builder에서의 오토레이아웃과 동일한 효과를 프로그래밍으로 할 수 있다
- 프로그래밍으로 부모/자식을 포함하는 뷰들을 배치할 수 있다

프로젝트 생성

- **새로운 프로젝트 생성**
  - 프로젝트 이름: programmingView
  - ViewController 클래스를 ConversionViewController로 변경
    - Navagator에서 ViewController.swift → ConversionViewController.swift로 변경
    - ConversionViewController.swift를 클릭하여
      - class ViewController: UIViewController → class ConversionViewController: UIViewController로 변경
    - Main.storyboard에서 ViewController을 선택
      - 4번째 Indentity Inspector에서 클래스를 ConversionViewController로 변경

LayoutGuide

- **LayoutGuide**
  - 오토레이아웃을 위한 사각형 영역
  - 10개의 앵커(Anchor)를 가지고 있다.
    - 8개의 앵커는 좌표를 의미하며
    - 2개(wdithAnchor, heightAnchor)는 크기를 의미함

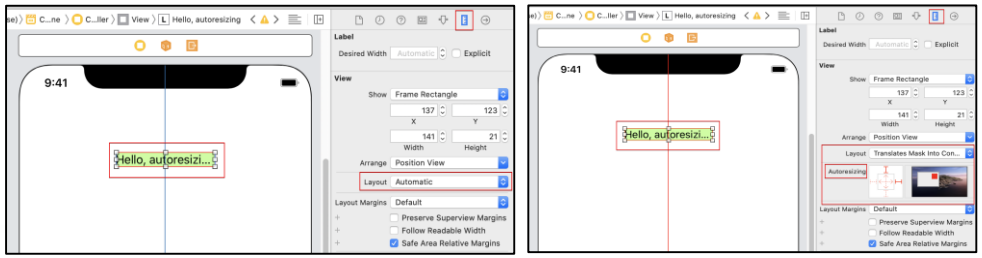


프로그래밍 순서

- **프로그래밍으로 오토레이아웃 설정**
  - 원하는 컨트롤을 생성한다.
  - 생성된 컨트롤의 translatesAutoresizingMaskIntoConstraints를 false로 설정
    - 오토리상이징(오토레이아웃이 아님)을 막아야 오토레이아웃이 적용됨
  - 생성된 컨트롤의 LayoutGuide에서 10개의 Anchor중 (x, y), (w, h)를 결정할 수 있는 앵커를 선택하여 설정
    - 예) leftAnchor, topAnchor, widthAnchor, heightAnchor

참고사항
Autoresizing: 초기 iOS에서는 다양한 Device 크기를 만족하는 App을 만들기 위해서 오토리사이징 개념을 도입하였음. 그러나 곧 다양성을 지원하기에는 한계가 있음을 알고 autolayout를 추가함. 그래서 하나의 View에 대한 constraint가 두 방식을 양립함

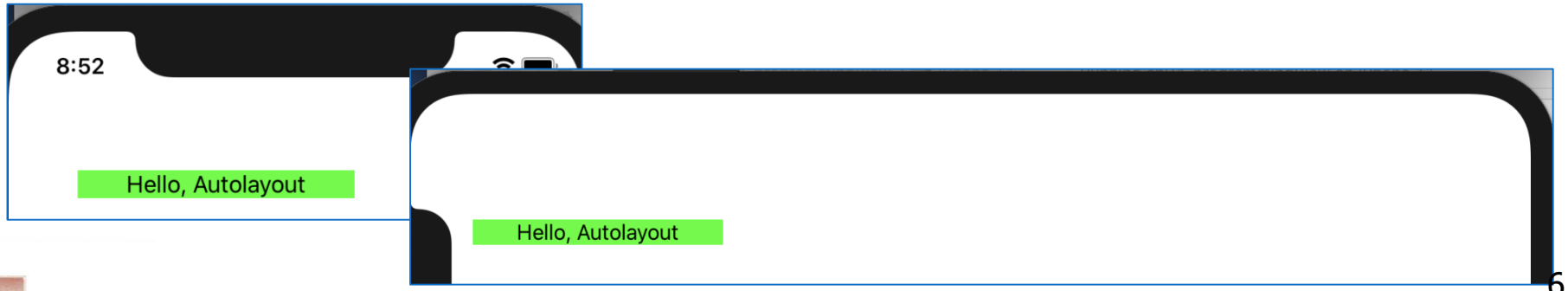
Interface Builder: 인터페이스 빌더에서 컨트롤을 놓으면, 자동으로 Autoresizing은 리셋됨, 즉 autolayout만 적용됨
Programming: autoresizing이 기본적으로 동작하도록 설정됨, 그래서 이를 해제하기 위해서 translatesAutoresizingMaskIntoConstraints를 false로 설정함



절대 레이아웃

- **기본 방법**
  - "Hello, Autolayout" Label

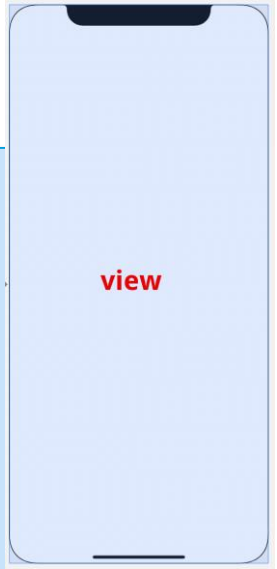
```
class ConversionViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()

        let helloLabel = UILabel(frame: CGRect(x: 100, y: 100, width: 200, height: 30))
        helloLabel.text = "Hello, Autolayout"
        helloLabel.backgroundColor = UIColor.green
        helloLabel.font = UIFont.systemFont(ofSize: 30, weight: .bold)
        helloLabel.textAlignment = .center
        view.addSubview(helloLabel)
    }
}
```

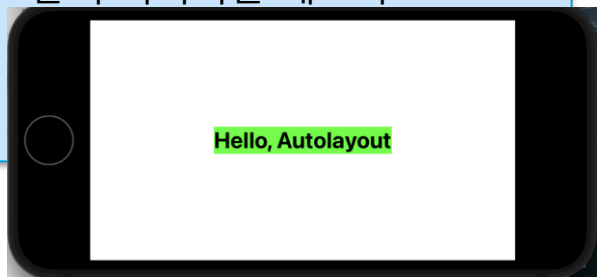


프로그래밍 오토레이아웃

- **view 객체를 기준으로**
  - 화면의 중앙으로 "Hello, autolayout" 레이아웃하기

```
class ConversionViewController : UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.

        ... // 앞의 코드를 복사하시오
        view.addSubview(helloLabel)
        // 위 코드를 주석처리하고 (1)의 위치에서 주석을 해제하고 실행해보라
        // 또 (2)의 위치로 옮겨 실행해 보라

        helloLabel.translatesAutoresizingMaskIntoConstraints = false // true로 해보라
        let centerXConstraint = helloLabel.centerXAnchor.constraint(equalTo: view.centerXAnchor, constant: 0)
        let centerYConstraint = helloLabel.centerYAnchor.constraint(equalTo: view.centerYAnchor, constant: 0)

        //(1) view.addSubview(helloLabel)
        centerXConstraint.isActive = true
        centerYConstraint.isActive = true   // 이것만 false로 해보라 또는 주석처리를 해보라
        //(2) view.addSubview(helloLabel)
    }
}
```





safe area 기준으로

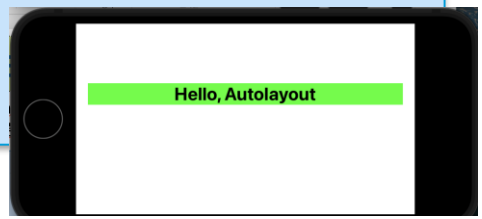
- **view.safeAreaLayoutGuide**
  - 화면의 중앙으로 "Hello, autolayout" 레이아웃하기


```
class ConversionViewController : UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        ...

        view.addSubview(helloLabel)

        helloLabel.translatesAutoresizingMaskIntoConstraints = false
        helloLabel.centerXAnchor.constraint(equalTo: view.safeAreaLayoutGuide.centerXAnchor).isActive =
true
        helloLabel.centerYAnchor.constraint(equalTo: view.safeAreaLayoutGuide.centerYAnchor).isActive =
true
    }
}
```

safe area 기준으로

- **view.safeAreaLayoutGuide**
  - "Hello, autolayout": 왼쪽여백 20, 오른쪼여백 20, 톱여백 100

```
class ConversionViewController : UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        ...
        view.addSubview(helloLabel)

        helloLabel.translatesAutoresizingMaskIntoConstraints = false

        helloLabel.leadingAnchor.constraint(equalTo: view.safeAreaLayoutGuide.leadingAnchor,
constant: 20).isActive = true

        helloLabel.trailingAnchor.constraint(equalTo: view.safeAreaLayoutGuide.trailingAnchor,
constant: -20).isActive = true

        helloLabel.topAnchor.constraint(equalTo: view.safeAreaLayoutGuide.topAnchor, constant:
100).isActive = true

    }
}
```



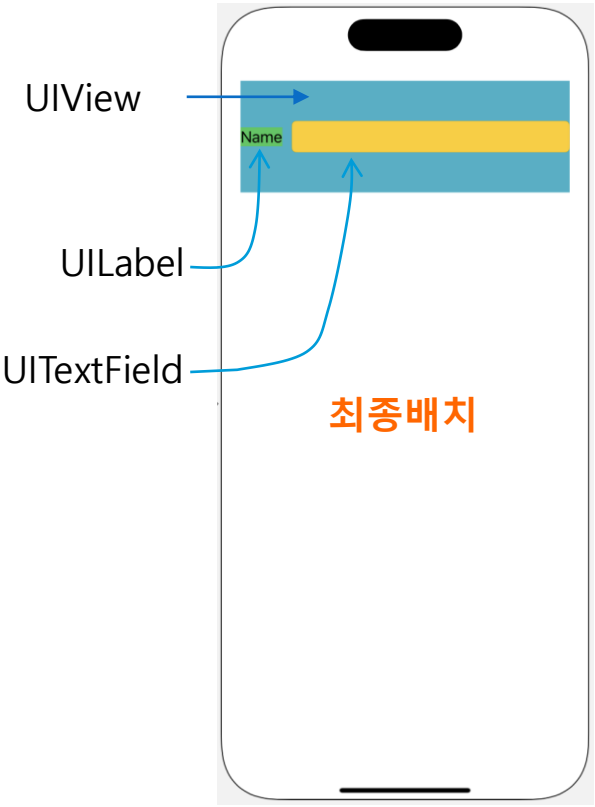
코드 단순화

- **NSLayoutConstraints.activate([, , ,])**
  - 앞과 동일한 결과

```
class ConversionViewController : UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        ...
        view.addSubview(helloLabel)
        helloLabel.translatesAutoresizingMaskIntoConstraints = false

        NSLayoutConstraint.activate([
            helloLabel.leadingAnchor.constraint(equalTo: view.safeAreaLayoutGuide.leadingAnchor,
constant: 20),
            helloLabel.trailingAnchor.constraint(equalTo: view.safeAreaLayoutGuide.trailingAnchor,
constant: -20),
            helloLabel.topAnchor.constraint(equalTo: view.safeAreaLayoutGuide.topAnchor, constant:
100)
        ]);
    }
}
```

예제: UIView 이용

- **우측과 같은 UI를 만들어라**
  - UIView, UILabel, UITextField를 맨아래 그림과 같이 놓이도록 프로그래밍하라



예제: UIView 이용

- **객체 생성**
  - UIView: 상하좌우 여백 20, 20, 20, 높이 120
  - Name: 좌여백 0, 우 여백 10, 수평중앙->UIView의 중앙에 정렬
  - UITextField: 좌여백 10, 우여백 0, 수평중앙->UIView의 중앙에 정렬
- **UIView 객체 및 부착**

```
// 뷰(outer) 객체를 생성하여 view에 부착한다
let outer = UIView()
outer.backgroundColor = UIColor(red: 00, green: 0xff, blue: 0xff, alpha: 1)
outer.translatesAutoresizingMaskIntoConstraints = false
view.addSubview(outer)

// outer의 좌우상 여백을 20, 20, 100으로하고, 높이를 120으로 설정한다
NSLayoutConstraint.activate([
   outer.leadingAnchor.constraint(equalTo: view.safeAreaLayoutGuide.leadingAnchor, constant: 20),
   outer.trailingAnchor.constraint(equalTo: view.safeAreaLayoutGuide.trailingAnchor, constant: -20),
   outer.topAnchor.constraint(equalTo: view.safeAreaLayoutGuide.topAnchor, constant: 100),
   outer.heightAnchor.constraint(equalToConstant: 120)
]);
```

예제: UIView 이용

- **nameLabel, nameTextField 객체를 생성 및 outer에 부착**

```
// UILabel 생성 및 초기화
let nameLabel = UILabel()
nameLabel.text = "Name"
nameLabel.backgroundColor = .green
nameLabel.translatesAutoresizingMaskIntoConstraints = false

// UITextField 생성 및 초기화
let nameTextField = UITextField()
nameTextField.backgroundColor = .yellow
nameTextField.translatesAutoresizingMaskIntoConstraints = false

// 부모뷰인 outer에 부착
outer.addSubview(nameLabel)
outer.addSubview(nameTextField)

// nameLabel의 허깅 우선순위를 높게 설정
nameLabel.setContentHuggingPriority(.defaultHigh, for: .horizontal)

// nameTextField의 허깅 우선순위를 낮게 설정
nameTextField.setContentHuggingPriority(.defaultLow, for: .horizontal)
```

예제: UIView 이용

- **nameLabel, nameTextField 객체의 허깅 우선순위 적용**

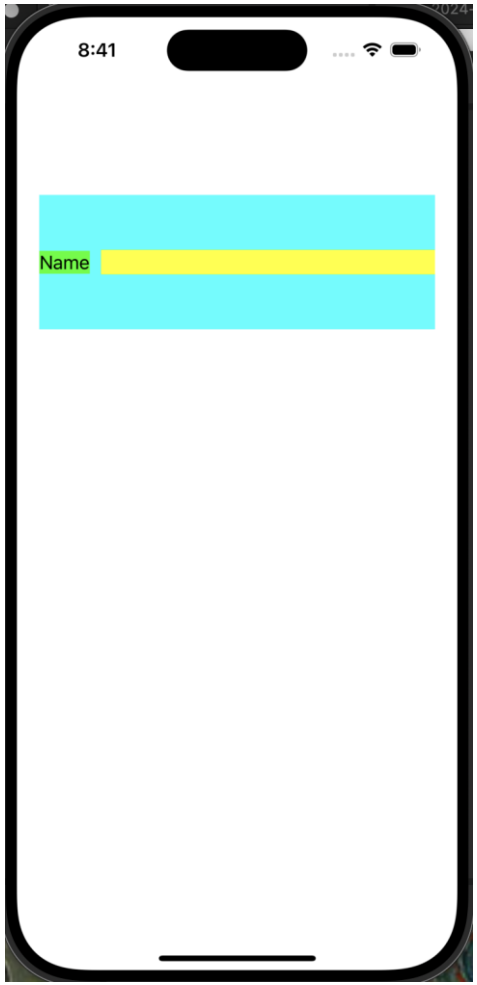
```
// nameLabel의 허깅 우선순위를 높게 설정
nameLabel.setContentHuggingPriority(.defaultHigh, for: .horizontal)
// nameTextField의 허깅 우선순위를 낮게 설정
nameTextField.setContentHuggingPriority(.defaultLow, for: .horizontal)
```

- **nameLabel, nameTextField 객체의 제한조건 설정**

```
NSLayoutConstraint.activate([
   // nameLabel을 outer의 y 중앙과 일치
   nameLabel.centerYAnchor.constraint(equalTo: outer.centerYAnchor),
   nameLabel.leadingAnchor.constraint(equalTo: outer.leadingAnchor, constant: 0),
   nameLabel.trailingAnchor.constraint(equalTo: nameTextField.leadingAnchor, constant: -10),

   nameTextField.centerYAnchor.constraint(equalTo: nameLabel.centerYAnchor, constant: 0),
   nameTextField.trailingAnchor.constraint(equalTo: outer.trailingAnchor, constant: 0)
])
```

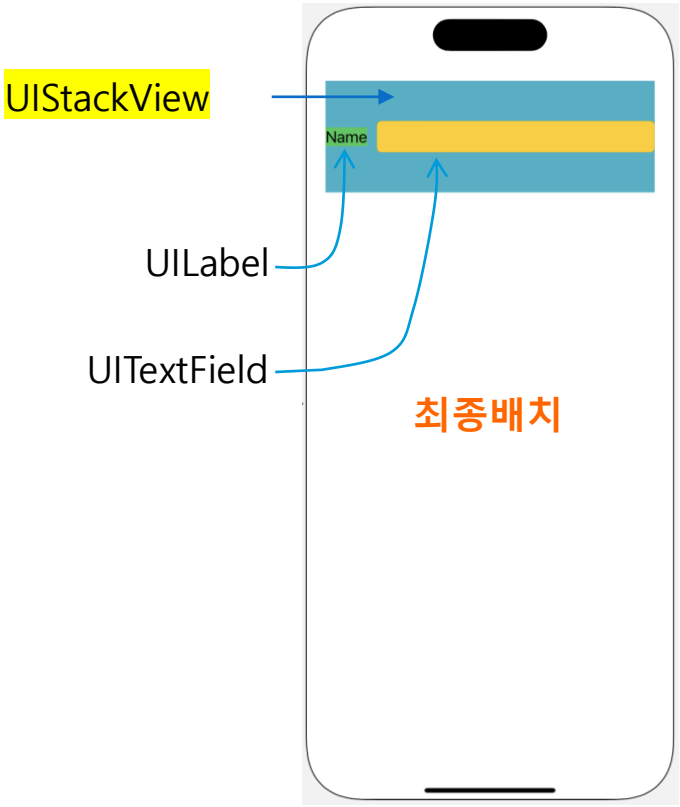
예제: UIView 이용

- 실행



예제: UIStackView 이용

- **우측과 같은 UI를 만들어라**
  - UIStackView, UILabel, UITextField를 맨 아래 그림과 같이 놓이도록 프로그래밍 하라
  - UIStackView
    - addSubView(...): 단순히 자식뷰들을 추가한다
    - addArrangedSubView(...): 자식뷰를 추가하면서 StackView의 규칙을 따른다
      - 예) axis, alignment, distribution등



예제: UIStackView 이용

- **객체 생성**
  - UIStackView: 상좌우 여백 20, 20, 20, 높이 120
    - Axis: Horizontal, Alignment: Fill, Distrinution: Fill, Spacing: 10
  - Name: 허깅 우선순위 High
  - UITextField: 허기 우선순위 Low
- **UIStackView 객체 및 부착**

```
let outer = UIStackView()
outer.backgroundColor = UIColor(red: 00, green: 0xff, blue: 0xff, alpha: 1)
outer.translatesAutoresizingMaskIntoConstraints = false

// 스택뷰 속성 설정
outer.axis = .horizontal
outer.alignment = .center
outer.distribution = .fill
outer.spacing = 10

// 스택뷰를 view에 부착
view.addSubview(outer)
```

예제: UIStackView 이용

- **UIStackView 제한조건 설정**
  - 일반 뷰와 동일하다

```
NSLayoutConstraint.activate([
   outer.leadingAnchor.constraint(equalTo: view.safeAreaLayoutGuide.leadingAnchor, constant: 20),

   // 20이 아니라 -20이다.
   outer.trailingAnchor.constraint(equalTo: view.safeAreaLayoutGuide.trailingAnchor, constant: -20),
   outer.topAnchor.constraint(equalTo: view.safeAreaLayoutGuide.topAnchor, constant: 100),
   outer.heightAnchor.constraint(equalToConstant: 120)
]);
```

예제: UIStackView 이용

- **nameLabel, nameTextField 객체를 생성 및 outer에 부착**

```
// 앞의 예제와 동일
let nameLabel = UILabel()
nameLabel.text = "Name"
nameLabel.backgroundColor = .green
nameLabel.translatesAutoresizingMaskIntoConstraints = false

// 앞의 예제와 동일
let nameTextField = UITextField()
nameTextField.backgroundColor = .yellow
nameTextField.translatesAutoresizingMaskIntoConstraints = false

// addSubView가 아니라 addArrangedSubview이다.
//      outer.addSubview(nameLabel)
//      outer.addSubview(nameTextField)
outer.addArrangedSubview(nameLabel)
outer.addArrangedSubview(nameTextField)
```

예제: UIStackView 이용

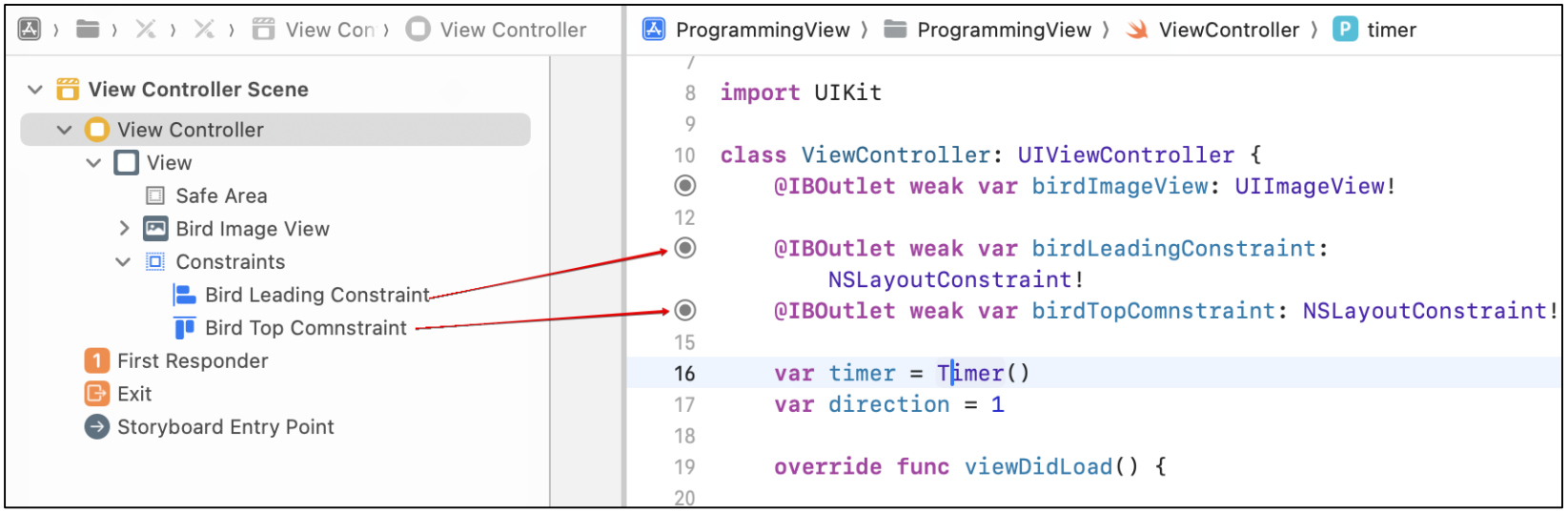
- **nameLabel, nameTextField 객체의 허깅 우선순위 적용**

```
// 앞과 동일
nameLabel.setContentHuggingPriority(.defaultHigh, for: .horizontal)
nameTextField.setContentHuggingPriority(.defaultLow, for: .horizontal)
```

- **nameLabel, nameTextField 객체의 제한조건 설정**
  - 별도의 제한 조건이 없음
    - 따라서 스택뷰를 사용하는 것이 쉽다.

IB에서의 제한조건 Outlet

- **IB에서의 제한조건 Outlet**
  - Interface Builder에서 설정한 Constraints도 UIView처럼 프로그램에서 변수로 액세스할 수 있음
  - Document Outline에서 원하는 Constraint를 Control+Drag하여 ViewController에 Drop하여 연결 생성



예제: 날아가는 새

- **예제: 날으는 새**
  - 화면의 맨위에서 새가 날아가는 애니메이션
- **UIImageView 및 설정**
  - Images 폴더에서 image.png를 프로젝트에 복사
    - Navigator 판넬에서 asset폴더를 오픈하라
    - Image.png를 그곳에 drag & drop하라
  - Object Library에서 UIImageView를 선택하여 ViewController에 드롭하라
  - UIImageView 설정
    - 속성 Inspector의 Image에 bird를 선택하라.
    - 왼쪽여백 0, 윗여백 0, 크기(40, 20)으로 설정하라.
    - 이를 ViewController의 birdImageView로 Outlet하라
  - Constraints와 ViewController연결
    - Main.storyboard와 ViewController.swift를 동시에 오픈하라
    - Document Outline에서 왼쪽여백의 Constraint를 Control+Drag하여 ViewController.swift에서 drop하라
      - 이름을 birdLeadingConstraint로 하라
    - Document Outline에서 윗여백의 Constraint를 Control+Drag하여 ViewController.swift에서 drop하라
      - 이름을 birdTopConstraint로 하라

예제: 날아가는 새

- **새의 움직임**
  - 새는 0.1초마다 왼쪽 또는 오른쪽으로 10픽셀만큼 이동
  - 양쪽 끝에 도달하면 방향을 바꾸어 이동
  - 이를 위하여 Timer() 객체가 필요
- **Timer 객체**
  - 타이머 생성

```
Timer.scheduledTimer(
                timeInterval: 0.1,                          // 호출 주기
                target: self,                               // 호출 주체
                selector: #selector(timerAction), // 실행 함수, 제스처와 유사
                userInfo: nil,                                          // 사용자 정보
                repeats: true                                           // 반복 호출 여부
)
```

  - 타이머 죽임

```
Timer.invalidate()
```

예제: 날아가는 새

- 코딩

```
class ViewController: UIViewController {

    @IBOutlet weak var birdImageView: UIImageView!
    @IBOutlet weak var birdLeadingConstraint: NSLayoutConstraint!
    @IBOutlet weak var birdTopComnstraint: NSLayoutConstraint!
    var timer: Timer!                   // 타이머 레퍼런스
    var direction = 1                   // 새가 날아가는 방향, 1: 오른쪽, -1: 외쪽
    override func viewDidLoad() {
        super.viewDidLoad()

        // 타이머 생성 및 설정
        timer = Timer.scheduledTimer(timeInterval: 0.1, target: self, selector: #selector(timerAction), userInfo: nil, repeats: true)
    }

   // 뒷 슬라이드에 계속
    …
}
```

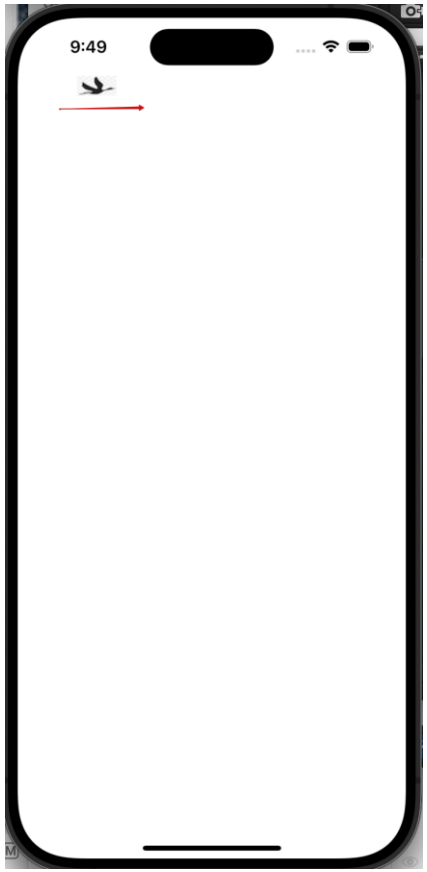
예제: 날아가는 새

- 코딩 계속

```
class ViewController: UIViewController {

    ...

    // Action함수처럼 @objc가 필요함
    @objc func timerAction(){

        // 왼쪽 여백의 값을 변경하여 새를 이동시킨다.
        birdLeadingConstraint.constant += CGFloat(direction*10)


        if birdLeadingConstraint.constant + birdImageView.frame.size.width >= view.frame.size.width{
            // 오른쪽 끝에 도달하면
            direction = -1
            // 새의 방향을 반전함
            birdImageView.transform = .init(scaleX: -1, y: 1)
        }else if birdLeadingConstraint.constant < 0{
            direction = 1
            birdImageView.transform = .init(scaleX: 1, y: 1)
        }
    }
}
```

예제: 날아가는 새

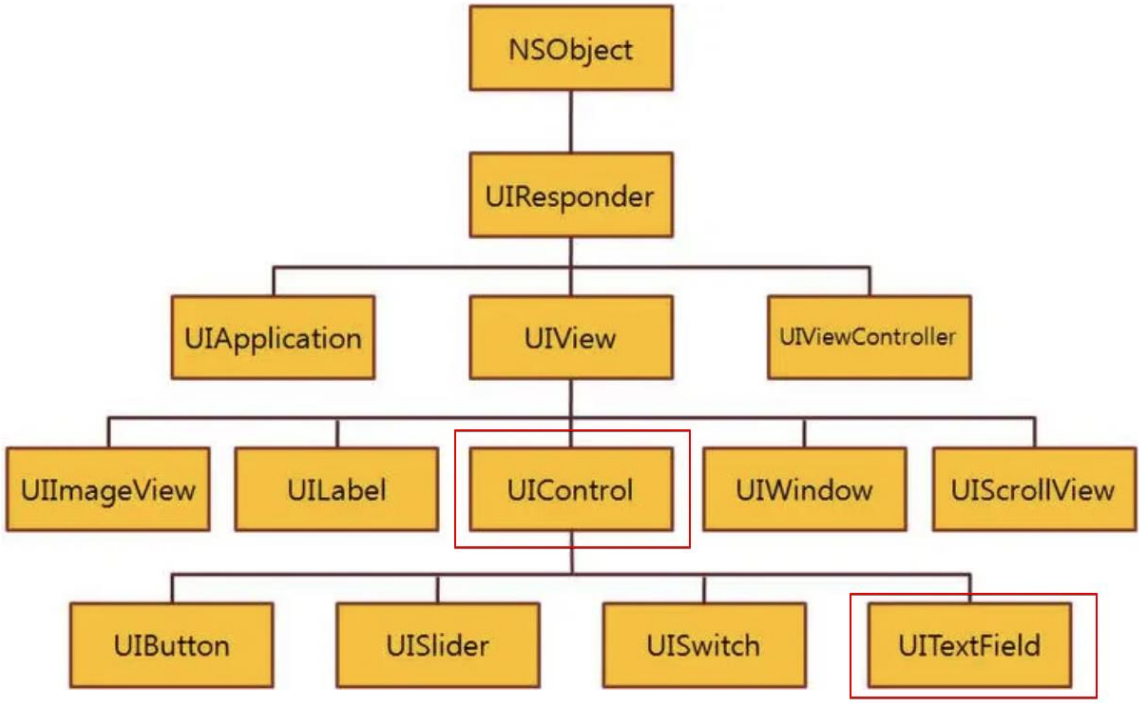
- 실행



Action달기

- **Action 달기**
  - Interface Builder에서 Action에 대한 함수를 설정할 수 있듯이 프로그래밍에서도 Action에 대한 함수를 설정할 수 있음
    - GestureRecognizer와 유사

- **Action을 가지는 클래스의 계층구조**



Action달기

- **Action 종류(UIControl.EVENT)**

touchDown 컨트롤을 터치했을 때 발생하는 이벤트
touchDownRepeat 컨트롤을 연속 터치 할 때 발생하는 이벤트
touchDragInside 컨트롤 범위 내에서 터치한 영역을 드래그 할 때 발생하는 이벤트
touchDragOutside 터치 영역이 컨트롤의 바깥쪽에서 드래그 할 때 발생하는 이벤트
touchDragEnter 터치 영역이 컨트롤의 일정 영역 바깥쪽으로 나갔다가 다시 들어왔을 때 발생하는 이벤트
touchDragExit 터치 영역이 컨트롤의 일정 영역 바깥쪽으로 나갔을 때 발생하는 이벤트
touchUpInside 컨트롤 영역 안쪽에서 터치 후 뗐을때 발생하는 이벤트
touchUpOutside 컨트롤 영역 안쪽에서 터치 후 컨트롤 밖에서 뗐을때 이벤트
touchCancel 터치를 취소하는 이벤트 (touchUp 이벤트가 발생되지 않음)
valueChanged 터치를 드래그 및 다른 방법으로 조작하여 값이 변경되었을때 발생하는 이벤트
primaryActionTriggered 버튼이 눌릴때 발생하는 이벤트 (iOS보다는 tvOS에서 사용)
editingDidBegin UITextField에서 편집이 시작될 때 호출되는 이벤트
editingChanged UITextField에서 값이 바뀔 때마다 호출되는 이벤트
editingDidEnd UITextField에서 외부객체와의 상호작용으로 인해 편집이 종료되었을 때 발생하는 이벤트
editingDidEndOnExit UITextField의 편집상태에서 키보드의 return 키를 터치했을 때 발생하는 이벤트
allTouchEvents 모든 터치 이벤트
allEditingEvents UITextField에서 편집작업의 이벤트
applicationReserved 각각의 애플리케이션에서 프로그래머가 임의로 지정할 수 있는 이벤트 값의 범위
systemReserved 프레임워크 내에서 사용하는 예약된 이벤트 값의 범위
allEvents 시스템 이벤트를 포함한 모든 이벤트

Action달기

- **UIControl.addTarget 함수**
  - UIControl은 UIView의 하위객체임

```
func addTarget(_ target: Any?,
        action: Selector,
           for controlEvents: UIControl.Event)
```

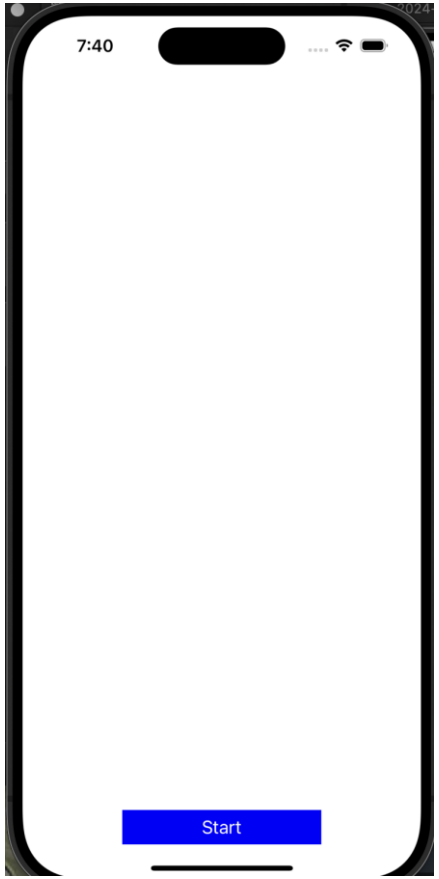
  - 특정 이벤트(UIControl.Event)가 발생하면 target에 있는 Selector 함수를 실행하라
  - Selector함 함수주소로써 #selector(함수이름)으로 주어야 함
    - 이때 함수는 @objc키워드를 주어야 함

```
@objc func 함수이름(...){

}
```

  - UIControl.Event는 앞의 슬라이드 중의 하나

UIButton에 Action달기

- **UIButton에 Action달기**
  - 버튼을 생성하여 view에 부착
    - 타이틀: Start // setTitle함수 사용
    - 배경색: 그린
  - 제약조건 설정
    - 수평으로 화면 중앙
    - Bottom이 화면의 바닥에 위치
    - 크기: 전체 화면의 ½
  - 액션 달기
    - 버튼을 클릭하면 Start/Stop가 토글링함



UIButton에 Action달기

- **UIButton 객체 생성, 설정 및 제약조건**

```
override func viewDidLoad() {
    super.viewDidLoad()

    // 버턴 객체 생성
    let button = UIButton()
    button.setTitle("Start", for: .normal)                    // 타이틀 설정
    button.backgroundColor = .green
    button.translatesAutoresizingMaskIntoConstraints = false

    view.addSubview(button)

    NSLayoutConstraint.activate([
        button.centerXAnchor.constraint(equalTo: view.safeAreaLayoutGuide.centerXAnchor),
        button.bottomAnchor.constraint(equalTo: view.safeAreaLayoutGuide.bottomAnchor),
        button.widthAnchor.constraint(equalTo: view.widthAnchor, multiplier: 0.5)
    ])
}
```

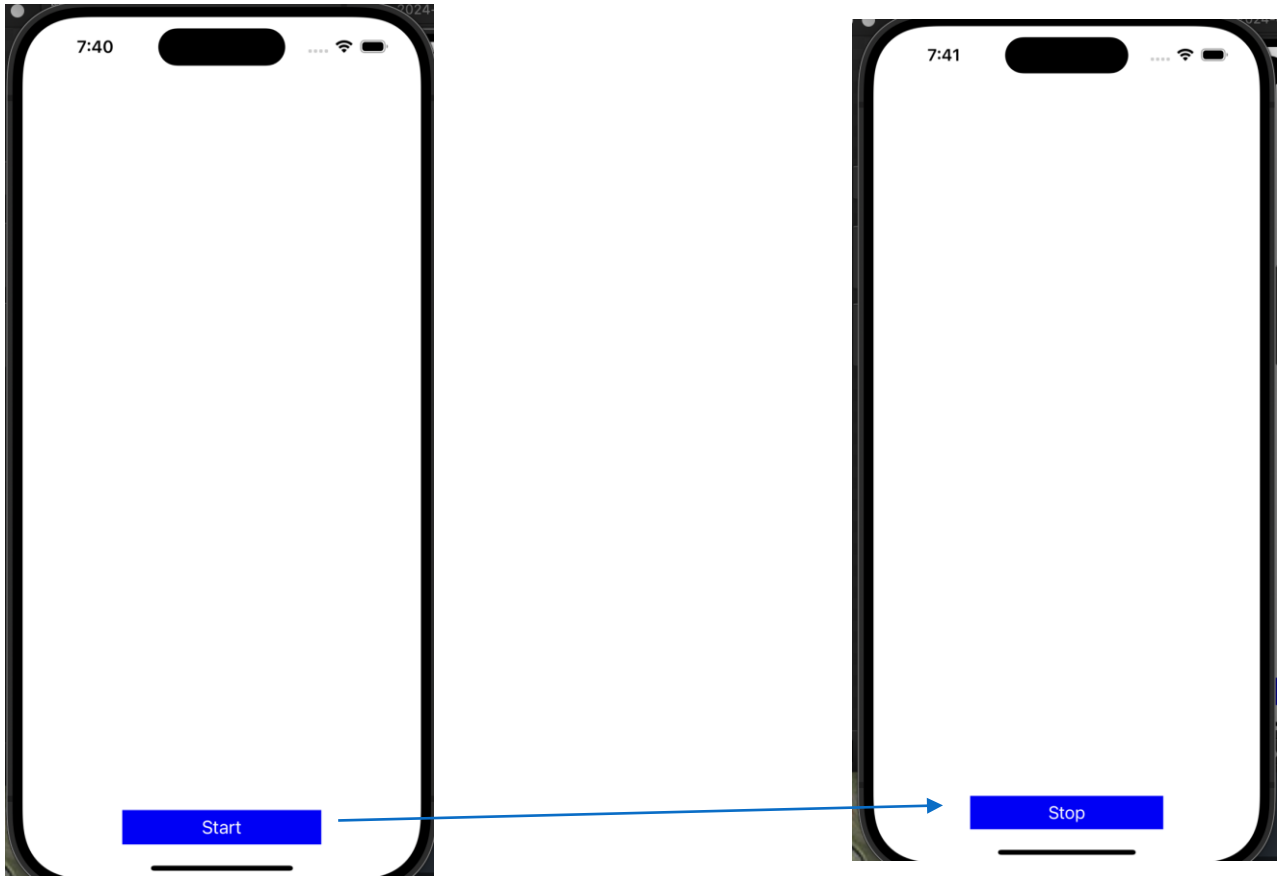
UIButton에 Action달기

- **UIButton에 touch up inside 액션 설정**

```
override func viewDidLoad() {
   super.viewDidLoad()

   // 앞부분 생략
    ...

   // GestureRecognizer과 매우 유사하다
   button.addTarget(self, action: #selector(buttonPressed), for: .touchUpInside)
}

// Objective C의 함수 호출과 동일하게 하기 위하여 @objc 키워드를 사용하여야 함
@objc func buttonPressed(_ sender: UIButton){
   if let text = sender.titleLabel?.text{
      sender.setTitle((text == "Start") ? "Stop" : "Start", for: .normal)
   }
}
```

UIButton에 Action달기

- 테스트



예제: 날아가는 새 제어하기

- **앞의 날으는 새 예제에 버턴 액션 추가**
  - viewDidLoad()에서 타이터 생성을 주석처리
  - buttonPressed함수를 다음과 같이 수정

```
@objc func buttonPressed(_ sender: UIButton){

//      if let text = sender.titleLabel?.text{
//         sender.setTitle((text == "Start") ? "Stop" : "Start", for: .normal)
//      }

   if let text = sender.titleLabel?.text{
      if text == "Start"{
         // 타이머 생성, 새가 날기 시작
         timer = Timer.scheduledTimer(timeInterval: 0.1, target: self,
                                      selector: #selector(timerAction),
                                      userInfo: nil, repeats: true)
      }else{
         // 타이머 중지
         timer.invalidate()
      }
      sender.setTitle((text == "Start") ? "Stop" : "Start", for: .normal)
   }
}
```

예제: 날아가는 새 제어하기

- **실행**
  - 버턴을 클릭할때 마다 새가 이동하다 멈춘다