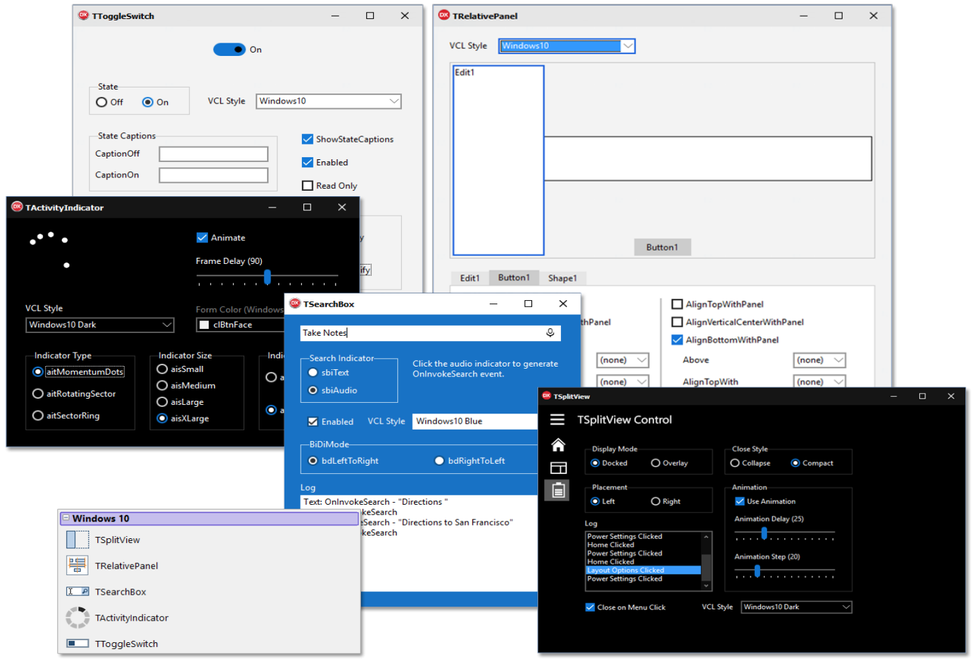
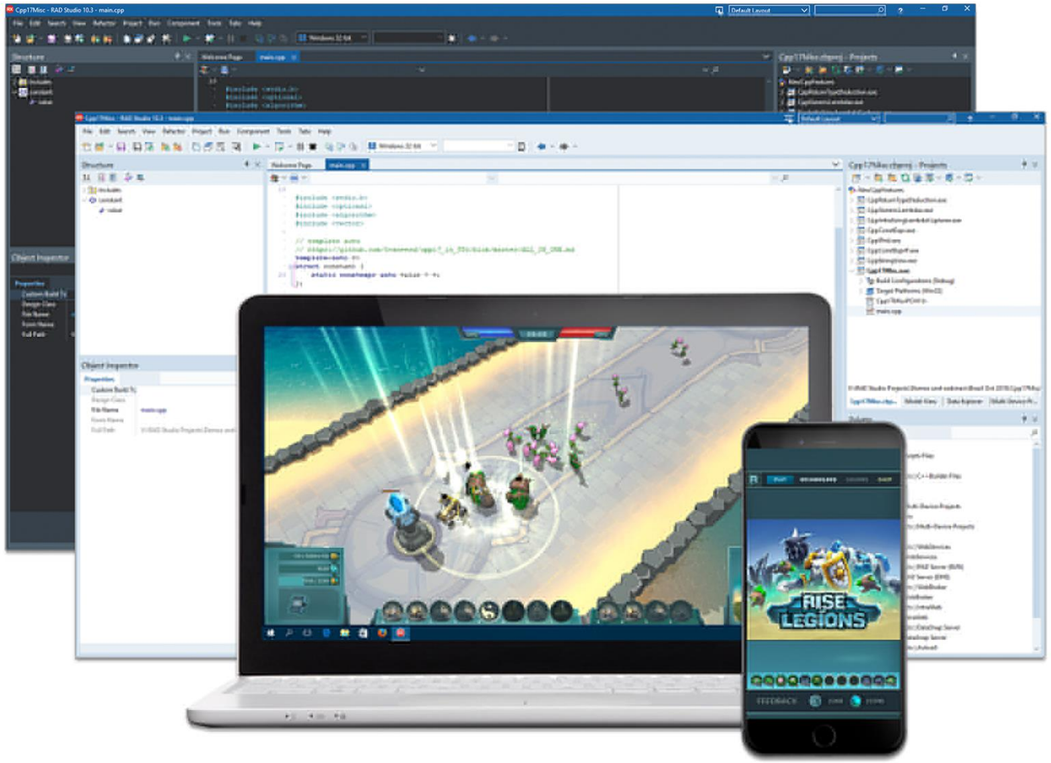
[IDE 트랙- 기초] 개발환경 사용법 및 아주 간단한 애플리케이션 만들기

- 델파이 툴 소개
- 델파이 개발 환경
- VCL / FMX 간단한 비교
- 델파이 프로젝트를 구성하는 파일들 살펴보기
- 프로젝트 소스 살펴보기
- 유니트의 구조 살펴보기
- 간단하게 컴포넌트 사용 방법
- [실습] Hello Word 작성하기

빠르고 시각적인 개발

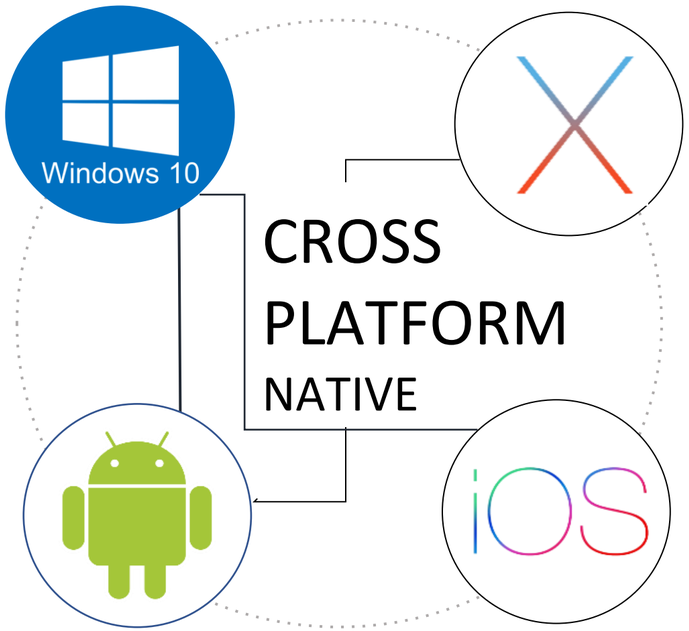
RAD 스튜디오는 윈도우10용 강력한 VCL 컨트롤과 윈도우, 맥, 리눅스, iOS 그리고 안드로이드 애플리케이션 개발 가능한 FMX 멀티-디바이스 제공

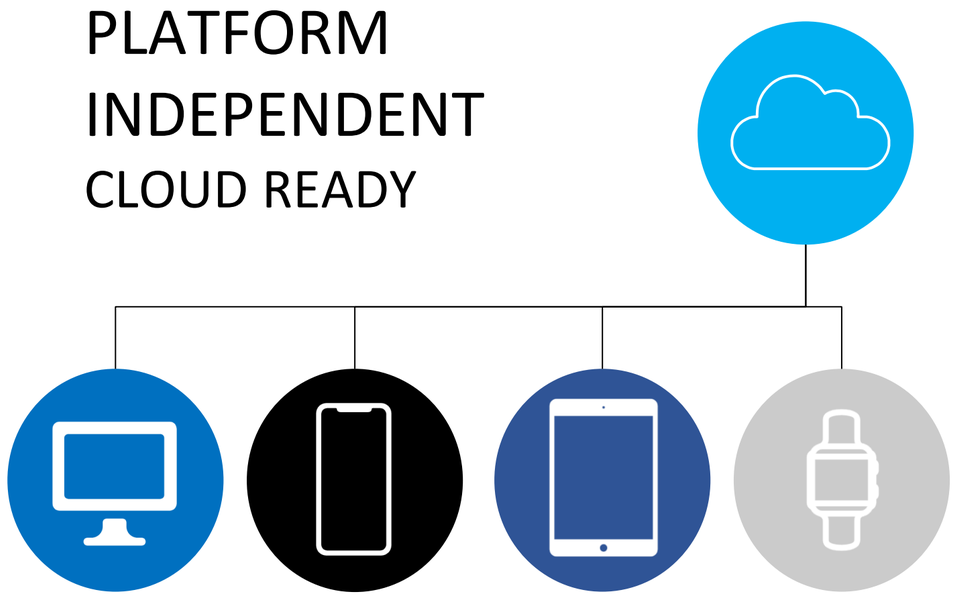




RAD 스튜디오 개요

RAD 스튜디오는 C++과 델파이 개발자 모두에게 사랑받는 기능을 갖춘 최고의 IDE.
네이티브 성능의 크로스 플랫폼 배포를 위한 디자인, 코드, 디버깅과 테스트 제공.





대부분의 비즈니스 사용은 여전히 데스크탑에서 시작

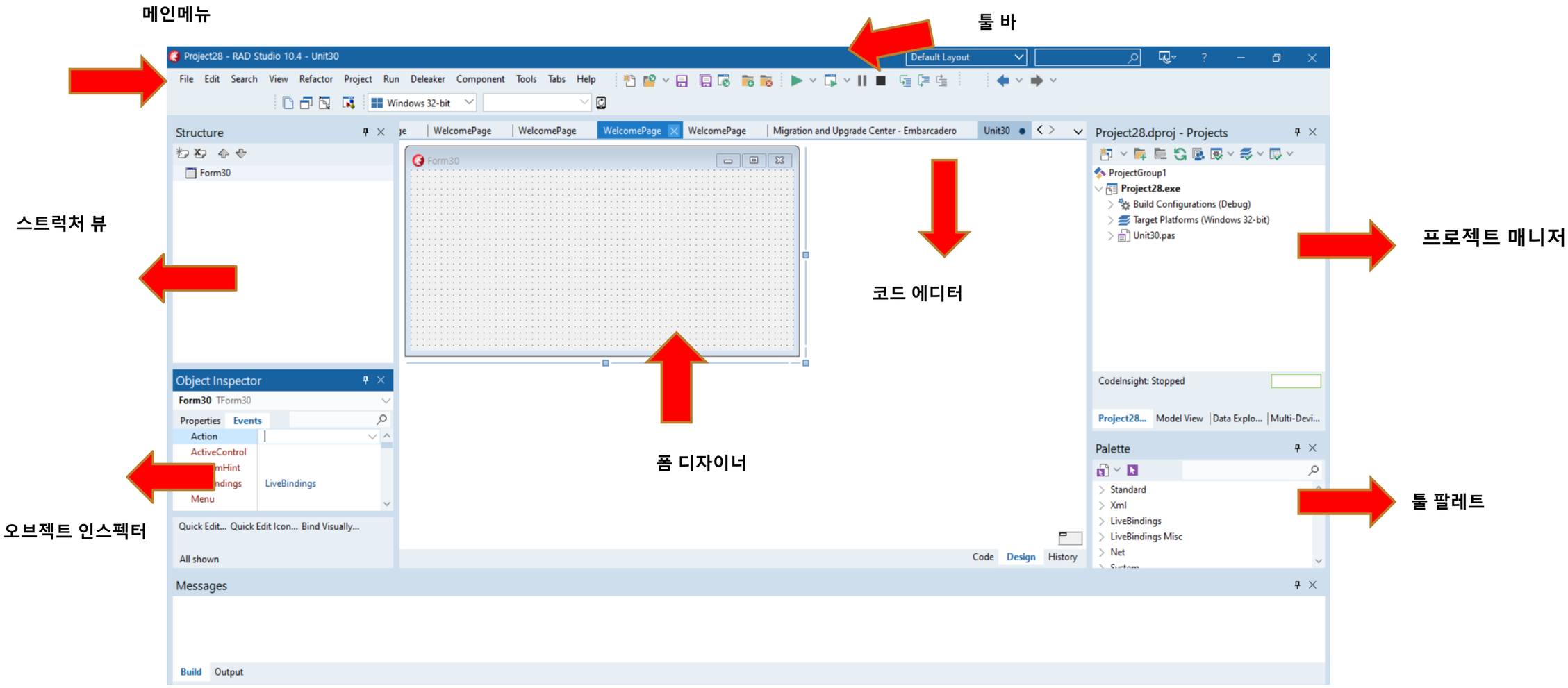
새로운 앱 vs
데스크톱 앱 복제

amazon WorkSpaces

Thinfinity® VirtualUI™

데스크탑을
“브라우저”로
확장하는 솔루션

개발환경 사용법 및 아주 간단한 애플리케이션 만들기



프로젝트의 종류

- VCL 프로젝트(VCL Form Application) - 윈도우 프로그램 개발
- FMX 프로젝트(Multi-Device Application) - 윈도우/맥/iOS/안드로이드 용 프로그램 개발
- 콘솔 프로젝트(Console Application) - 화면이 없는 커맨드 라인 기반 프로그램 개발

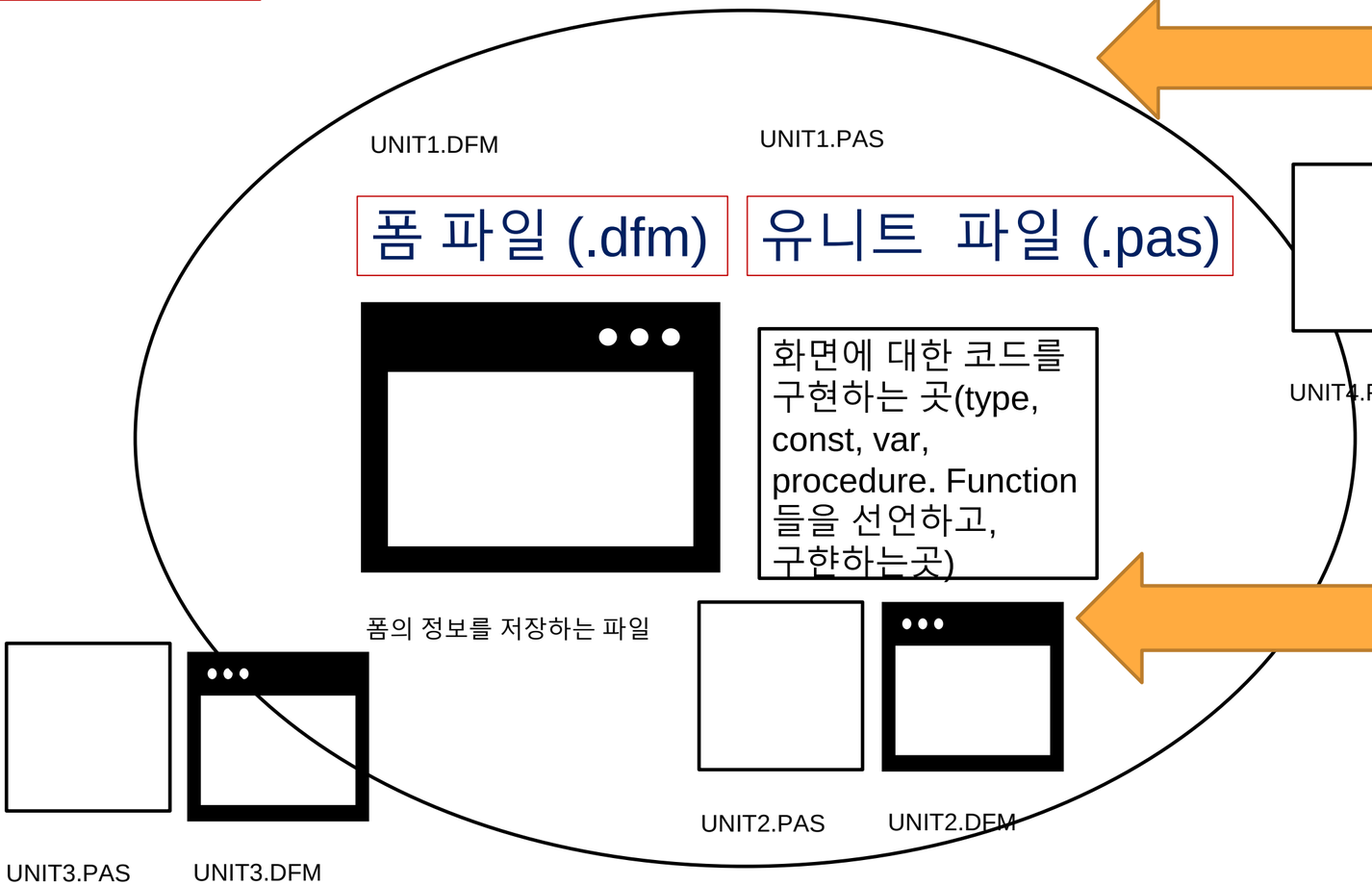
프로젝트 시작

위저드 사용 New Vcl Application

프로젝트 파일(.dpr)

프로젝트 파일은
메인 소스로 보자기 처럼 모든 파일들의 정보를 연결합니다.

New Unit



유니트 각 각을 컴파일하여 .dcu (Delphi Compiled Unit)를 만듭니다.

New Form

프로젝트 구성 파일들

파일	파일 확장자	설명
프로젝트 파일	.dpr	프로젝트 메인 소스로 프로젝트에서 사용되는 모든 파일들을 연결하는 파일로 일종의 보자기 역할 (* 여러 개의 프로젝트를 그룹화하여 프로젝트그룹파일 (.groupproj)도 만들 수 있음
프로젝트 파일	.dproj	델파이 2007부터 생성되는 프로젝트 파일로 개발 프로젝트의 저장 및 로드 형식으로 사용되며 소스 코드 파일 참조, 컴파일러 및 링커 설정, 프로젝트 디렉토리 및 기타 프로젝트 설정을 포함합니다. Xml 형식으로 프로젝트를 컴파일하는 데 사용되는 MSBuild 호환 파일입니다.
유니트 파일	.pas	코딩의 최소단위로 필요한 타입, 상수, 변수, 프로시저, 함수 등을 선언하고 구현하는 파일
폼 파일	.dfm	폼을 구성하는 정보가 저장되는 파일로 태생은 이진 바이너리 파일이지만, 델파이 5.0부터 텍스트 형태로 저장되고 편집 가능
Delphi Compiled Unit	.dcu	유니트가 컴파일 된 파일(c 언어의 .obj 파일과 비슷하나 c의 오브젝트 파일과 달리 dcu는 오브젝트파일과 선언부가 합쳐진 상태라고 말 할 수 있는데 pas 파일의 기본 구조, 즉 형 정의, 변수, 상수 정의 등을 모두 가지고 있어서 다른 유닛이 dcu의 선언부를 참조 할 수 있다.
리소스파일	.res	윈도우의 표준 리소스 파일을 사용합니다. 이 리소스 파일에는 비트맵, 커서, 아이콘, 텍스트 등 응용 프로그램에서 사용되는 자원(resource) 들이 포함되어 있습니다.
	.dsk	프로젝트와 이름이 같은 확장자로, 프로젝트의 어떤 창이 어떤 위치에 열려 있는지 등의 상태에 대한 정보를 저장

유니트 구조

Section	설명
interface	Type,const, var, procedure, function등을 선언하는 부분으로 이곳에 선언된 것들은 자기 유니트및 외부 유니트에서도 사용가능(단 uses절에 추가되어야 함)
implementation	구현부분으로 interface에 선언된 루틴들을 구현하는섹션 이 부분에도 필요한 type, const, var, procedure, function들을 선언할 수 있으나 외부 유니트에서는 사용할 수 없음
initialization	메모리를 할당, 변수에 디폴트 값을 할당하는등 초기처리 작업을 수행하는 섹션 (uses절을 만났을때 이 섹션이 수행됨)
finalization	Initialization 의 반대로 마무리 작업을 수행

클래스와 개체

- 클래스

붕어빵 만드는 방식 즉 논리적인 대상

- 개체,오브젝트

붕어빵 틀에서 구워 나온 붕어빵(물리적인 대상)

- 인스턴스

생생된 각각의 고유

*객체 지향 프로그래밍(OOP)에서 **인스턴스**(**instance**)는 해당 클래스의 구조로 컴퓨터 저장공간에서 할당된 실체를 의미한다. 여기서 클래스는 속성과 행위로 구성된 일종의 설계도이다. OOP에서 객체는 클래스와 **인스턴스**를 포함한 개념이다.

Uses

- ## Uses 의미

  프로젝트 (.dpr) 파일의 uses 절 뒤에 in과 파일 이름이 있는 유닛만 프로젝트의 일부로 간주

- ## Uses 범의(검색 경로)

  모든 유닛은 컴파일러의 검색 경로에 있어야합니다. 현재 디렉토리, library path 에 지정된 유닛들을 참조합니다.

- ## Uses 순서

  uses 절에 순서는 초기화 순서를 결정하고 컴파일러가 식별자를 찾는 방식에 영향을 줍니다. 같은 이름을 가진 변수, 상수, 유형, 프로시저 또는 함수가 있는 다른 유닛을 uses하는 경우 컴파일러는 uses 절에서 마지막에 나열된 유닛을 사용합니다. 특정 유닛에서 식별자에 액세스 하려면 UnitName.Identifier를 추가해야 합니다.

- ## Uses 가시성

  즉, 유닛의 인터페이스 섹션에서 시작하여 다른 유닛의 인터페이스 섹션을 통해 참조를 따라 해당 유닛으로 복귀 할 수 없어야 합니다. 서로 참조하는 경우 implementation에 uses 해야합니다.

•클래스

클래스는 속성과 행위를 갖는 레코드형과 마찬가지로 일종의 자료형(Type)입니다.
클래스 형으로 정의한 변수는 그 자체가 개체(오브젝트)가 되는 것이 아니라 메모리에 자리잡기 위해서는 인스턴스 하는 작업이 필요한데 이를 생성(Create)이라고 합니다.
내부적으로 데이터와 메소드를 가지고 있습니다. 클래스는 Type 절에서 선언합니다.
클래스를 선언할 때는 예약어 **class**를 사용하며 괄호 안에 계승 받을 선조 클래스를 표시합니다.
조상클래스를 생략하면 가상 상위의 클래스 **TObject**에서 계승 받는 것을 의미합니다 .

•개체

메모리에 생성된 오브젝트가 바로 개체 또는 인스턴스 입니다. 클래스가 붕어빵을 만드는 틀이라면 개체는 구워져 나온 붕어빵과 같습니다. 개체는 자신만의 데이터를 가지며 클래스의 행위를 수행합니다

생성자/파괴자 루틴

- **Create**

➢ 생성자는 예약어 Constructor로 시작하는 특별한 프로시저로 개체를 생성하고 초기화하는 동작을 수행하는 루틴입니다. 보통 생성자는 Create라는 이름으로 만들며 다음과 같은 형태를 갖습니다.

➢ 생성자는 오브젝트가 생성되기 전에 호출되기 때문에 클래스 이름으로 지정하여 호출하도록 되어 있습니다. 새로운 클래스를 작성할 때 필요에 따라 생성 자를 작성할 수 있습니다.

Constructor TMyobj.Create;

Constructor TCompoent.Create(AOwner:TCompoent);

생성자는 형 이름으로 호출되는데 오브젝트를 위한 메모리 공간을 확보하고
선언된 필드들을 초기값으로 만들어 줍니다 그 후에 프로그래머가 기술한 생성자 코드를 수행하고 끝나면 할당된 오브젝트에 대한 주소 값을 돌려주게 됩니다.

- **Destroy**

  파괴자는 예약어 Destructor로 시작하는 루틴으로 생성자와 반대로 작업을 정리하고 메모리를 해제하는 기능을 수행합니다.

  Destructor TMyobj.Destroy;

- **Free**

  할당된 개체의 인스턴스를 체크하여 내부적으로 Destroy를 부르는 루틴입니다.

  Myobj.Free;

- FreeAndNil 과 Free의 차이점

개체(컴포넌트) 사용 방법

- Uses 절 추가

  Uses utest4;

- 변수 선언

  Var H : THourly;

- 생성자 Create를 호출하여 메모리할당&초기처리

  H := THouryly.Create;

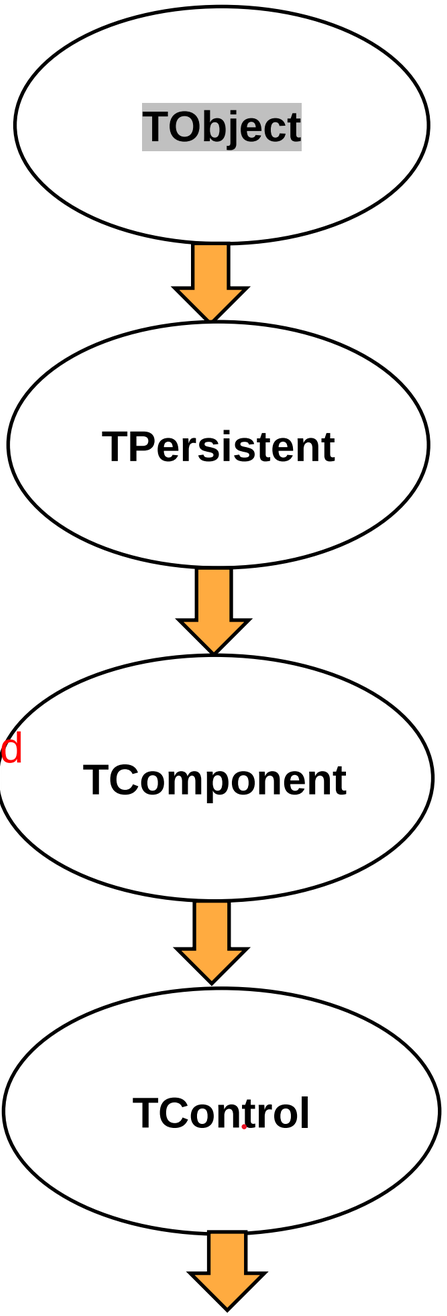
- 변수, 루틴사용

  H.GetName

- Free를 호출하여(내부적으로 Destroy) 메모리에서 해제

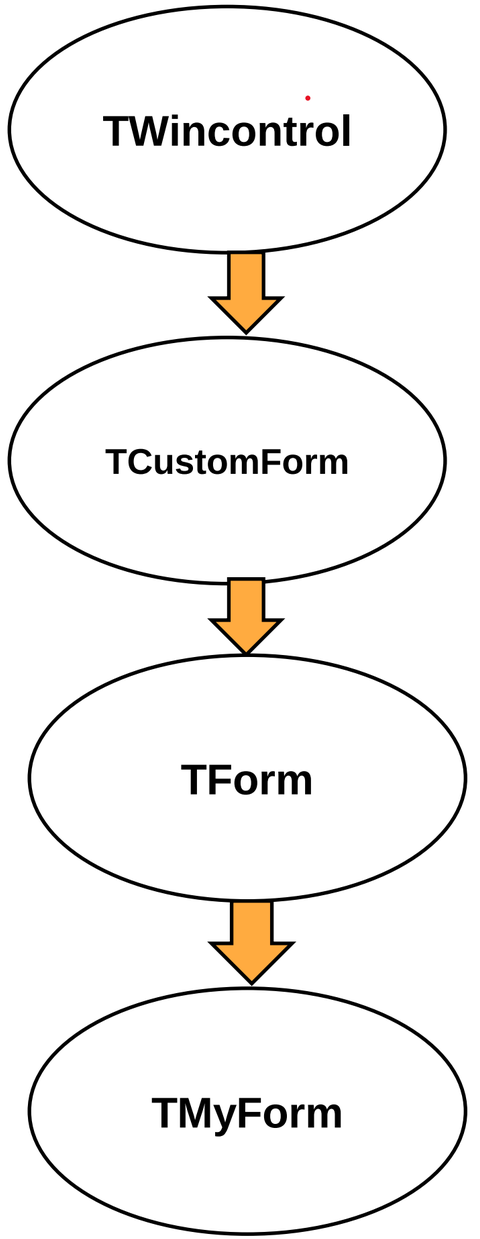
  H.Free;

클래스 계층



RTTI(Run Time
Type Infotmation)

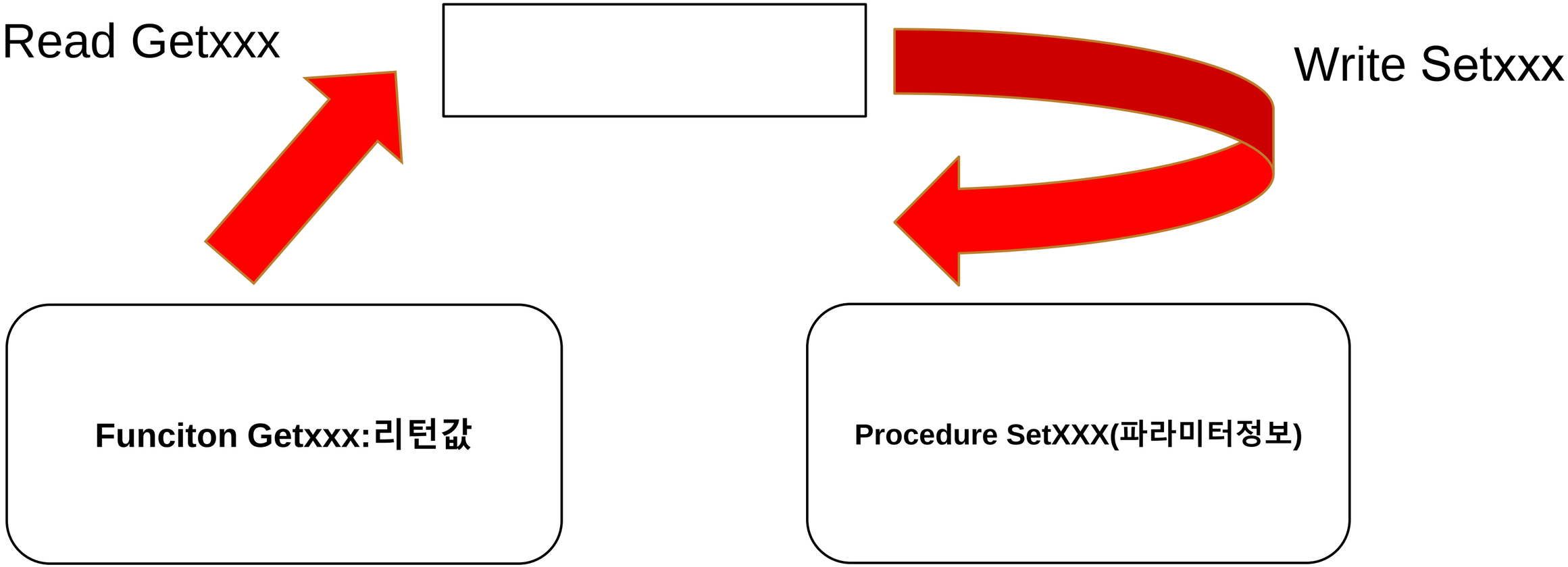
Property,Event,Method



Published 추가됨

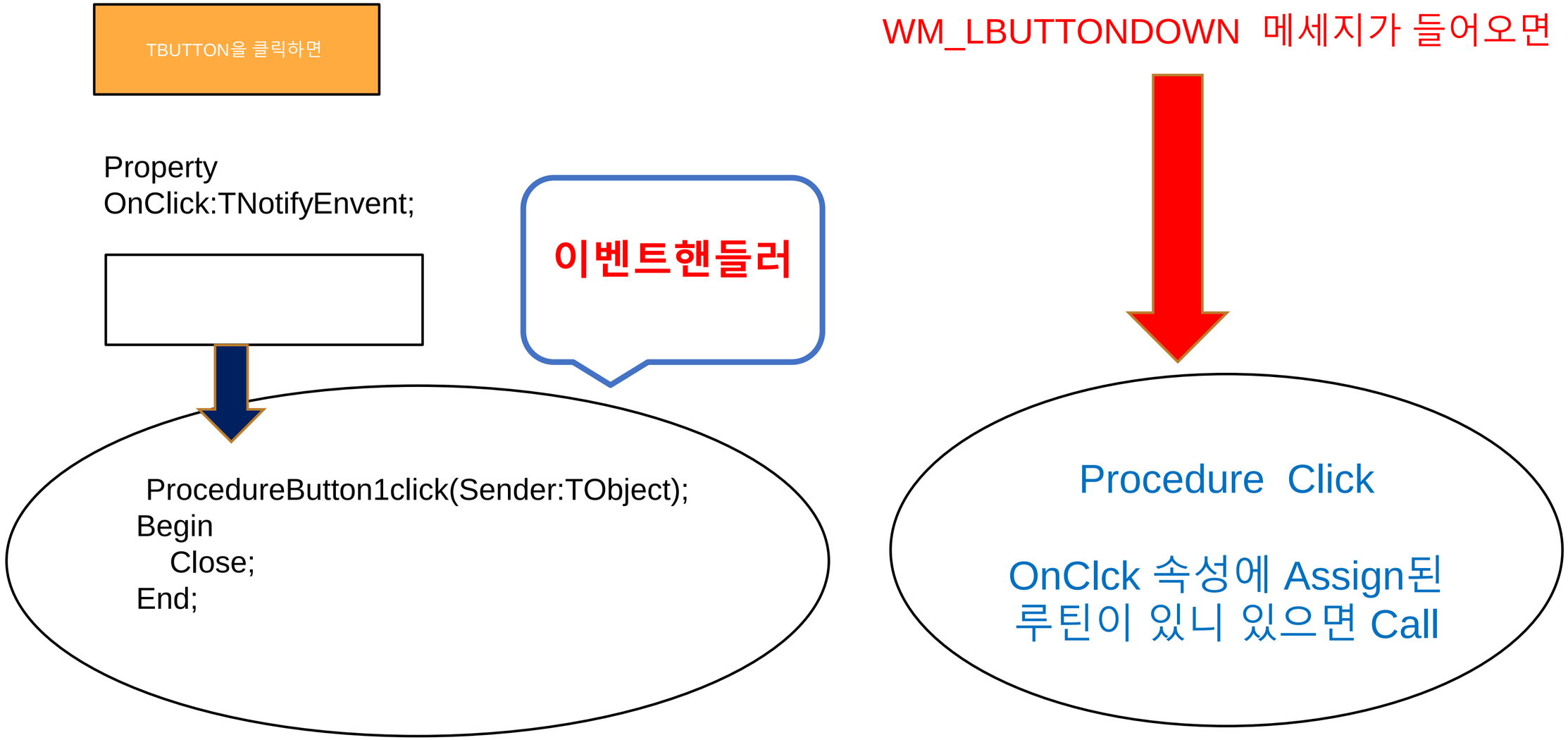
속성

- 속성(property)은 attribute variable로서 일종의 통로이다.

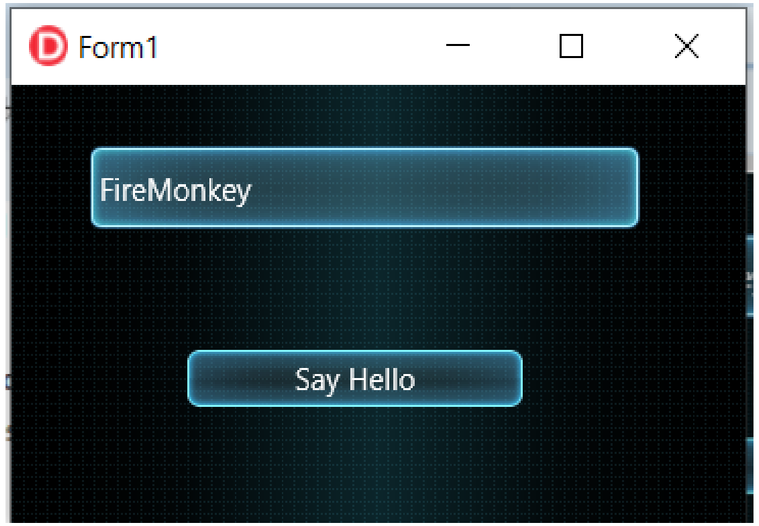


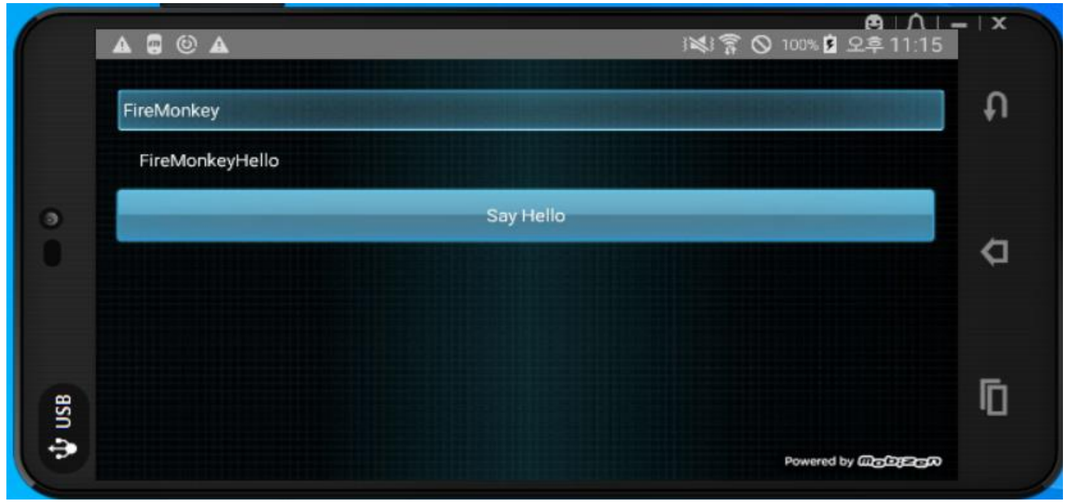
이벤트

- 이벤트(Event)는 약속된 시점이니 동작



실습: Hello Word 작성





버튼 클릭시 Edit의 Text의 다음과 같이 표시합니다.

```
Procedure TForm1.Button1Click(Sender:TObject)
Begin
  Label1.Text := Edit1.Text  + 'Hello';
End;
```