Solidity Çalışma Notları Part 3:

Fonksiyon Modifier’lar:

Fonksiyon olarak gözüküp fonksiyonun nasıl davranacağını belirler.

Örnek Kod kullanımı:

modifier onlyOwner() {

require(isOwner());

\_;

}

Yukarıdaki kodu fonksiyonumuzun sonuna eklersek fonksiyonumuzu sadece hesap sahibi çalıştırabilir.

Böylelikle fonksiyonumuzda güvenliği arttırarak dışardan gelen saldırılardan korunmuş oluruz.

Kontratların çalışma ücreti – Gas:

Gas aslında bir fonksiyonun çalışması için gösterdiği efora karşılık ödediğimiz bedeldir diyebiliriz. Bir fonksiyon ne kadar komplike yazılmışsa genelde aynı oranda gas tüketir. Bu sebepten dolayı Solidity kodlarını ne kadar optimize yazarsak kontratlarımız o kadar iyi olur.

Peki Gas neden gerekli ?

Ethereum makinesi yavaş ve ağır bir makinedir. Güvenlği sağlamak için bu fonksiyonun outputunun doğruluğunu bilmek için bütün nodelarda execute etmek zorundadır. Bu şekilde Ethereum merkeziyetsiz olduğunu gösterir.

Gas kurtarmak için Struct yapısını kullanmak:

Normalde struct dışı yaptığımız farklı uint atamalrının gas fiyatına etkisi olmaz

Fakat bunu Struct yapısında yaparsak gas ücretini düşürebiliriz. Aşağıdaki kodda

MiniMe NormalStructtan daha az gas ücreti çıkartır.

struct NormalStruct {

uint a;

uint b;

uint c;

}

struct MiniMe { ## MiniMe struct packing yaptığı için

uint32 a;

uint32 b;

uint c;

}

Time unitler:

Now, değeri son bloğun unix timestamp’ini geri döndürür. Fakat şöyle bir durum var. Unix time genel olarak 32-bit uintlerde tutulur fakat 2038de artık bu olay overflow olur ve hata verir. Bunun yerine 64-bit kullanmak ise gas kullanımı arttırır. Burada neyi kullanacağınız size kalmıştır. Solidity aynı zamanda seconds,hours,days,weeks,years olarak değerleri atanabilir.

Örnek time unitlerin kullanımı:

function \_triggerCooldown(Zombie storage \_zombie) internal {

\_zombie.readyTime = uint32(now + cooldownTime);

}

function \_isReady(Zombie storage \_zombie) internal view returns (bool) {

return (\_zombie.readyTime <= now);

}

View fonksiyonlar herhangi bir gas ücreti tüketmezler.

Bir bilgiyi storagede saklamak oldukça fazla gas tüketimine sebep olur. Bunun yerine memory içinde array atamak daha iyi bir yoldur. Böylelikle fonksiyon çağrıldıktan sonra bilgiler yok olur. Memory arrayler arrayin uzunluğu ile birlikte declare edilmelidir. Çünkü memory arrayler dinamik arraylerden değildir.