加密僵尸：僵尸作战系统

可支付

payable 方法是让 Solidity 和以太坊变得如此酷的一部分 —— 它们是一种可以接收以太的特殊函数。

先放一下。当你在调用一个普通网站服务器上的API函数的时候，你无法用你的函数传送美元——你也不能传送比特币。

但是在以太坊中， 因为钱 (_以太_), 数据 (*事务负载*)， 以及合约代码本身都存在于以太坊。你可以在同时调用函数 并付钱给另外一个合约。

```
contract OnlineStore {
  function buySomething() external payable {
    // 检查以确定0.001以太发送出去来运行函数:
    require(msg.value == 0.001 ether);
    // 如果为真，一些用来向函数调用者发送数字内容的逻辑
    transferThing(msg.sender);
  }
}
```

在这里，msg.value 是一种可以查看向合约发送了多少以太的方法，另外 ether 是一个内建单元。

这里发生的事是，一些人会从 web3.js 调用这个函数 (从DApp的前端)， 像这样：

```
// 假设 `OnlineStore` 在以太坊上指向你的合约:
OnlineStore.buySomething().send(from: web3.eth.defaultAccount, value: web3.utils.toWei(0.001))
```

注意这个 value 字段， JavaScript 调用来指定发送多少(0.001)以太。如果把事务想象成一个信封，你发送到函数的参数就是信的内容。 添加一个 v 信封里面放钱 —— 信件内容和钱同时发送给了接收者。

> 注意： 如果一个函数没标记为payable， 而你尝试利用上面的方法发送以太，函数将拒绝你的事务。

提现

```
contract GetPaid is Ownable {
  function withdraw() external onlyOwner {
    owner.transfer(this.balance);
  }
}
```

你可以通过 transfer 函数向一个地址发送以太， 然后 this.balance 将返回当前合约存储了多少以太。 所以如果100个用户每人向我们支付1以太， this.balance 将是100以太。

你可以通过 transfer 向任何以太坊地址付钱。 比如，你可以有一个函数在 msg.sender 超额付款的时候给他们退钱：

```
uint itemFee = 0.001 ether;
msg.sender.transfer(msg.value - itemFee);
```

或者在一个有卖家和卖家的合约中， 你可以把卖家的地址存储起来， 当有人买了它的东西的时候，把买家支付的钱发送给它 seller.transfer(msg.

随机数

他是不安全的，但对于本游戏已经足够，这里不做阐述用cekka256产生随机数的方法

重构通用逻辑

用 modifier 来清理代码并避免重复编码。

LEVI_104