zombiefactory.sol

```
pragma solidity ^0.4.25;

import "./ownable.sol";
import "./safemath.sol";

contract ZombieFactory is Ownable {

  using SafeMath for uint256;
  using SafeMath32 for uint32;
  using SafeMath16 for uint16;

  event NewZombie(uint zombieId, string name, uint dna);

  uint dnaDigits = 16;
  uint dnaModulus = 10 ** dnaDigits;
  uint cooldownTime = 1 days;

  struct Zombie {
    string name;
    uint dna;
    uint32 level;
    uint32 readyTime;
    uint16 winCount;
    uint16 lossCount;
  }

  Zombie[] public zombies;

  mapping (uint => address) public zombieToOwner;
  mapping (address => uint) ownerZombieCount;

  function _createZombie(string _name, uint _dna) internal {
    uint id = zombies.push(Zombie(_name, _dna, 1, uint32(now + cooldownTime),
0, 0)) - 1;
    zombieToOwner[id] = msg.sender;
    ownerZombieCount[msg.sender] = ownerZombieCount[msg.sender].add(1);
    emit NewZombie(id, _name, _dna);
  }

  function _generateRandomDna(string _str) private view returns (uint) {
    uint rand = uint(keccak256(abi.encodePacked(_str)));
    return rand % dnaModulus;
  }

  function createRandomZombie(string _name) public {
    require(ownerZombieCount[msg.sender] == 0);
    uint randDna = _generateRandomDna(_name);
    randDna = randDna - randDna % 100;
    _createZombie(_name, randDna);
  }

}
```