作业

1.重构create，使用新的帮助函数

```
pub fn create(origin) {
  let sender = ensure_signed(origin)?;
  // 作业：重构create方法，避免重复代码
  let kitty_id = Self::kitties_count();
  //if kitty_id == T::KittyIndex::max_value() {
  //  return Err("Kitties count overflow");
  //}
  Self::next_kitty_id()?;
  // Generate a random 128bit value
  //let payload = (
  //  <randomness_collective_flip::Module<T> as Randomness<T::Hash>>
  ::random_seed(),
  //  &sender,
  //  <system::Module<T>>::extrinsic_index(),
  //  <system::Module<T>>::block_number(),
  //);
  let payload = Self::random_value(&sender);
  let dna = payload.using_encoded(blake2_128);
  // Create and store kitty
  let kitty = Kitty(dna);
  //<Kitties<T>>::insert(kitty_id, kitty);
  //<KittiesCount<T>>::put(kitty_id + 1.into());
  // Store the ownership information
  //let user_kitties_id = Self::owned_kitties_count(&sender);
  //<OwnedKitties<T>>::insert((sender.clone(), user_kitties_id), kitty_id);
  //<OwnedKittiesCount<T>>::insert(sender, user_kitties_id + 1.into());
  Self::insert_kitty(sender, kitty_id, kitty);
}
```

2.完成combine_dna（在main中测试过）

```
fn combine_dna(kitty1_dna: u8, kitty2_dna: u8,
selector: u8) -> u8 {
    let temp1:u8;
    let temp2:u8;
    temp1 =  kitty1_dna & selector;
    temp2 =  kitty2_dna & !selector;
    let out:u8;
    out = temp1 | temp2;
    return out;
}


fn main() {
  let dna1:u8 = 0b11110000;
  let dna2:u8 = 0b11001100;
  let selector:u8 = 0b10101010;
  let out = combine_dna(dna1, dna2, selector);
  println!("0b{:08b}", out);
}
```

```
   Compiling substrate-kitties v2.0.0 (/Users/Mac/project/Team4/projects/les
son4old)
    Finished dev [unoptimized + debuginfo] target(s) in 4.34s
     Running `target/debug/substrate-kitties --dev --execution=native -lrunt
ime=debug`
0b11100100
```

// 测试数据：dna1 = 0b11110000, dna2 = 0b11001100, selector = 0b10101010, 返回值 0b11100100

3.设计加密猫模块V3

需求：a.transfer kitty转移猫 b.要求复杂度必须忧于O(n)

https://substrate.dev/rustdocs/v1.0/parity_codec/alloc/collections/struct.LinkedList.html
将用户拥有的猫用`linked list`结构来代替数组
使用linked list来达到transfer复杂度必须忧于O(n)的目标
将用户拥有的猫用linked list结构来代替数组
use std::collections::LinkedList;
let KittyList: LinkedList<u32> = LinkedList::new();

```
//验证调用者
let sender = ensure_signed(origin)?;
//验证调用者是否是所有者
let owner = Self::owner_of(kitty_id).ok_or("No owner for this kitty")?;
ensure!(owner == sender, "You do not own this kitty");
let owner = Self::owner_of(kitty_id).ok_or("No owner for this kitty")?;
ensure!(owner == from, "'from' account does not own this kitty");
//修改转让人与被转让人的拥有的猫的数量
let owned_kitty_count_from = Self::owned_kitty_count(&from);
let owned_kitty_count_to = Self::owned_kitty_count(&to);
let new_owned_kitty_count_to = owned_kitty_count_to.checked_add(1).ok_or("Transfer
causes overflow of 'to' kitty balance")?;
let new_owned_kitty_count_from =
owned_kitty_count_from.checked_sub(1).ok_or("Transfer causes underflow of 'from' kitty
balance")?;
//转让操作
let kitty_index = <OwnedKittiesIndex<T>>::get(kitty_id);
if kitty_index != new_owned_kitty_count_from {
    let last_kitty_id = <OwnedKittiesArray<T>>::get((from.clone(),
new_owned_kitty_count_from));
    <OwnedKittiesArray<T>>::insert((from.clone(), kitty_index), last_kitty_id);
    <OwnedKittiesIndex<T>>::insert(last_kitty_id, kitty_index);
}
<KittyOwner<T>>::insert(&kitty_id, &to);
<OwnedKittiesIndex<T>>::insert(kitty_id, owned_kitty_count_to);
<OwnedKittiesArray<T>>::remove((from.clone(), new_owned_kitty_count_from));
<OwnedKittiesArray<T>>::insert((to.clone(), owned_kitty_count_to), kitty_id);
<OwnedKittiesCount<T>>::insert(&from, new_owned_kitty_count_from);
<OwnedKittiesCount<T>>::insert(&to, new_owned_kitty_count_to);
//申明事件
Self::deposit_event(RawEvent::Transferred(from, to, kitty_id));
Ok(())
```

额外作业

1.创建新的polkadot apps项目

https://github.com/polkadot-js/apps/tree/master/packages/app-123code

2.设计如何在substrate中实现树形结构

可以使用Parity提供的Base-16 Modified Merkle Tree ("Trie") data structure：
https://github.com/paritytech/trie