Лабораторная работа с котиками №1

Этап 1

Храним соответвие <Имя кота> → <цвет, возраст, вес>

- *Имя кота:* `String`*; служит идентификатором, id*
- *Цвет:* `String`
- *Возраст:* `Int`
- *Вес:* `Int`

Чтобы хранить пару из ключа и значения, в одном массиве хранится ключ, в другом, по тому же индексу — значение.
Одновременно может храниться не более 1000 пар.
Для вхождения "Имя → цвет, возраст, вес" фиксируется формат вывода, на усмотрение исполнителя. Вывод вхождений осуществляется всегда в этом формате.
Команды для взаимодействия через CLI:

- *create*
  - *Гарантируется уникальность id, т.е. в массиве ключей не хранится ключа равного id.*
  - *Формат ввода: "*`create <имя> <цвет> <возраст> <вес>`*"*
  - *Выводит созданное вхождение*
- *read*
  - *Гарантируется корректность id, т.е. в массиве ключей точно хранится ключ равный id.*
  - *Формат ввода: "*`read <имя>`*"*
  - *Выводит прочитанное вхождение*
- *delete id*
  - *Гарантируется корректность id, т.е. в массиве ключей точно хранится ключ равный id.*
  - *Формат ввода: "*`delete <имя>`*"*
  - *Выводит OK*
- *readall*
  - *Выводит все пары ключей и значений в произвольном порядке*
  - *Формат ввода: "*`readall`*"*
  - *Каждое вхождение выводится с новой строки. По одной команде на строчке*

Пример работы

```
create рыжик рыжий 12 34
Cat: name="рыжик", color="рыжий", age=12, weight=34
create пыжик пыжий 56 78
Cat: name="пыжик", color="пыжий", age=56, weight=78
readall
Cat: name="рыжик", color="рыжий", age=12, weight=34
Cat: name="пыжик", color="пыжий", age=56, weight=78
read пыжик
Cat: name="пыжик", color="пыжий", age=56, weight=78
delete рыжик
OK
readall
Cat: name="пыжик", color="пыжий", age=56, weight=78
```

Комментарии

- В зависимости от подхода к оформлению кода, пункты про гарантию могут мешать, если выносить все операции в функцию: логично предположить что функция будет возвращать тип `Cat` (`not null`), но после цикла (на случай если ни одно вхождение не найдено) тоже нужно что-то возвращать (`null`) чтобы избежать ошибок от анализатора. В результате возвращаемый тип будет `Cat?`, что повлечёт дополнительные проверки (оператором `!!`) и места потенциальных `NullPointerException`
- Имя кота (ключ) дублируется в самой структуре. Вроде как место потенциальной ошибки, потому что ключ может не соответствовать имени, но с другой стороны удобно возвращать, принимать и хранить