УНИВЕРСИТЕТ СИНЕРГИЯ

ДОМАШНЕЕ ЗАДАНИЕ

Тема 2. Базовые возможности TypeScript

Цель работы: Отработать на практике навыки написания типизированного кода, использующего типы, дженерики, классы, методы (функции), служебные типы

План работы:

1. Написать абстрактный класс Resource<T> по работе с массивом объектов, включающий операции (методы):
   - получение полного массива объектов из контекста (get),
   - получение одного объекта по ключу/значению (getOne)
   - запись нового объекта в конец массива (add)
   - обновление найденного объекта (update)
   - удаление найденного объекта (delete)
2. Класс должен работать с любым массивом объектов, переданным ему дочерним классом при создании инстанса дочернего класса. См. дочерние классы UserModel и OrderModel.

3. Абстрактный класс должен сохранить массив данных в свой контекст (this.data) в конструкторе и проводить все операции изменения или чтения из своего контекста, не затрагивая исходные массивы с данными.

   Метод get() должен возвращать текущий массив данных из контекста, с учетом всех изменений, произведенных в других методах.

   Метод add(newObj) должен возвращать получившийся массив после добавления (push) элемента, соответственно.

   Метод getOne(key, val) должен возвращать либо один объект, удовлетворяющий условиям поиска, либо undefined. Поиск элемента может вестись по любому полю в объекте (см. примеры в console.log users.getOne и orders.getOne ниже)

Метод update(key, val, partialData) должен обновлять и возвращать либо один обновленный объект, удовлетворяющий условиям поиска, либо ничего не обновлять и возвращать undefined. Поиск элемента может вестись по любому полю в объекте (см. примеры в console.log users.upate и orders.update ниже). Данный метод не может добавлять новые поля, не соответствующие исходному интерфейсу объекта, он должен принимать третьим агрументом новые данные, которые будут переписаны в найденном объекте. Тип данных должен соответствовать Partial<T>.

Метод delete(key, val) должен удалять и возвращать либо один удаленный объект, удовлетворяющий условиям поиска, либо ничего не удалять и возвращать undefined. Поиск элемента может вестись по любому полю в объекте (см. примеры в console.log users.delete и orders.delete ниже).

Метод update(key, val, partialData) должен обновлять и возвращать либо один обновленный объект, удовлетворяющий условиям поиска, либо ничего не обновлять и возвращать undefined. Поиск элемента может вестись по любому полю в объекте (см. примеры в console.log users.upate и orders.update ниже). Данный метод не может добавлять новые поля, не соответствующие исходному интерфейсу объекта, он должен принимать третьим агрументом новые данные, которые будут переписаны в найденном объекте. Тип данных должен соответствовать Partial<T>.

Метод delete(key, val) должен удалять и возвращать либо один удаленный объект, удовлетворяющий условиям поиска, либо ничего не удалять и возвращать undefined. Поиск элемента может вестись по любому полю в объекте (см. примеры в console.log users.delete и orders.delete ниже).

Критерии оценки:
К1 — Создан базовый абстрактный класс и методы get, add, getOne, update и delete, создано поле класса, сохраняющее исходный массив и constructor (2 балла).
К2 — Написана логика для методов и конструктора согласно ТЗ (1 балл).
К3 — Типизированы аргументы и выходные данные всех методов (1 балл).
К4 — Все console.log выводят логи, описанные под ними, отсутствуют ошибки TS, не используются any или implicit any (noImplicitAny: true) (1 балл).

Итого: 5 баллов
Минимальное количество баллов, чтобы преподаватель смог зачесть вашу работу — 4

Рекомендации для выполнения:

1. Можно вынести дублирующийся в нескольких методах код поиска индекса элемента по полю и значению поля в отдельный приватный метод, например в #findIndex и переиспользовать его. Свойства и методы, начинающиеся с символа # обычно являются приватными по современной договоренности об наименовании переменных.
2. Ключи (key) для поиска должны быть ограничены названиями полей исходного объекта и собираться динамически абстрактным классом с помощью keyof
3. Тип аргумента, содержащего значение поля для поиска (val) можно задать с помощью union типов-примитивов
4. Для помощи с написанием метода обновления см. урок 6, Partial<Type>

Примеры:

```
// Абстрактный класс будет здесь
/**
* Набор исходных данных и примеров 1
*/

type DataType = {
    id: number;
    name: string;
    phone: string;
    email: string;
    address: string;
}
const mockUserData: DataType[] = [
    {
        "id": 1,
        "name": "Lane Mcdonald",
        "phone": "1-980-603-4363",
        "email": "dui@aol.com",
        "address": "P.O. Box 895, 4432 Placerat, Avenue",
    },
    {
        "id": 2,
        "name": "Emma Ford",
        "phone": "(472) 855-7514",
        "email": "aliquam.ornare@protonmail.net",
        "address": "P.O. Box 311, 427 Egestas Av.",
```

```
      "id": 3,
       "name": "Louis Juarez",
       "phone": "1-895-966-2657",
       "email": "venenatis.lacus@outlook.net",
       "address": "Ap #125-478 Sit Av.",
   },
   {
       "id": 4,
       "name": "Zorita Mason",
       "phone": "1-262-419-4287",
       "email": "arcu.vel@protonmail.net",
       "address": "P.O. Box 631, 1093 Metus Street",
   },
   {
       "id": 5,
       "name": "Harriet Acevedo",
       "phone": "1-788-618-2639",
       "email": "consequat.dolor@protonmail.net",
       "address": "P.O. Box 769, 5046 Pellentesque, Rd.",
   }
]

class UserModel extends Resouce<DataType> {
   constructor(data: DataType[]) {
       super(data)
   }
}

const users = new UserModel([...mockUserData]);

console.log('users.get()', users.get()); // Получение массива всех объектов из контекста
абстрактного класса
/**
* [LOG]: "users.get()",  [{
 "id": 1,
 "name": "Lane Mcdonald",
 "phone": "1-980-603-4363",
 "email": "dui@aol.com",
 "address": "P.O. Box 895, 4432 Placerat, Avenue"
}, {
 "id": 2,
 "name": "Emma Ford",
 "phone": "(472) 855-7514",
 "email": "aliquam.ornare@protonmail.net",
 "address": "P.O. Box 311, 427 Egestas Av."
}, {
 "id": 3,
 "name": "Louis Juarez",
 "phone": "1-895-966-2657",
 "email": "venenatis.lacus@outlook.net",
 "address": "Ap #125-478 Sit Av."
}, {
```

```
 "id": 4,
 "name": "Zorita Mason",
 "phone": "1-262-419-4287",
 "email": "arcu.vel@protonmail.net",
 "address": "P.O. Box 631, 1093 Metus Street"
}, {
 "id": 5,
 "name": "Harriet Acevedo",
 "phone": "1-788-618-2639",
 "email": "consequat.dolor@protonmail.net",
 "address": "P.O. Box 769, 5046 Pellentesque, Rd."
}]

*/
console.log('users add',users.add({
   "id": 6,
   "name": "Ivan Petrov",
   "phone": "1-788-618-2639",
   "email": "consequat.dolor@protonmail.net",
   "address": "P.O. Box 769, 5046 Pellentesque, Rd.",
})); // Push объекта в массив. Объект может быть только типа Datatype
/**
* [LOG]: "users add",  [{
 "id": 1,
 "name": "Lane Mcdonald",
 "phone": "1-980-603-4363",
 "email": "dui@aol.com",
 "address": "P.O. Box 895, 4432 Placerat, Avenue"
}, {
 "id": 2,
 "name": "Emma Ford",
 "phone": "(472) 855-7514",
 "email": "aliquam.ornare@protonmail.net",
 "address": "P.O. Box 311, 427 Egestas Av."
}, {
 "id": 3,
 "name": "Louis Juarez",
 "phone": "1-895-966-2657",
 "email": "venenatis.lacus@outlook.net",
 "address": "Ap #125-478 Sit Av."
}, {
 "id": 4,
 "name": "Zorita Mason",
 "phone": "1-262-419-4287",
 "email": "arcu.vel@protonmail.net",
 "address": "P.O. Box 631, 1093 Metus Street"
}, {
 "id": 5,
 "name": "Harriet Acevedo",
 "phone": "1-788-618-2639",
 "email": "consequat.dolor@protonmail.net",
 "address": "P.O. Box 769, 5046 Pellentesque, Rd."
}
```

```
, {
 "id": 6,
 "name": "Ivan Petrov",
 "phone": "1-788-618-2639",
 "email": "consequat.dolor@protonmail.net",
 "address": "P.O. Box 769, 5046 Pellentesque, Rd."
}]

*/

console.log('users.getOne()', users.getOne('id', 4));  // Получение одного объекта с id: 4 (возможно получение по любому ключу и его значению)
console.log('users.getOne()', users.getOne('name', 'Lane Mcdonald'));  // Получение одного объекта по совпадению name
console.log('users.getOne()', users.getOne('phone', '(472) 855-7514'));  // Получение одного объекта по совпадению phone
console.log('users.getOne()', users.getOne('id', 12));  // Не существующий id, вернется undefined
/**
* [LOG]: "users.getOne()",  {
 "id": 4,
 "name": "Zorita Mason",
 "phone": "1-262-419-4287",
 "email": "arcu.vel@protonmail.net",
 "address": "P.O. Box 631, 1093 Metus Street"
}
[LOG]: "users.getOne()",  {
 "id": 1,
 "name": "Lane Mcdonald",
 "phone": "1-980-603-4363",
 "email": "dui@aol.com",
 "address": "P.O. Box 895, 4432 Placerat, Avenue"
}
[LOG]: "users.getOne()",  {
 "id": 2,
 "name": "Emma Ford",
 "phone": "(472) 855-7514",
 "email": "aliquam.ornare@protonmail.net",
 "address": "P.O. Box 311, 427 Egestas Av."
}
[LOG]: "users.getOne()",  undefined

*/


console.log('users.update()', users.update('id', 4, { name: 'Sergey' })); // Изменение поля 'name' в объекте с 'id' равным 4, возвращает измененный объект (поиск по любому ключу, изменяемые данные должны быть совместимы с DataType (Partial))
console.log('users.update()', users.update('email', 'dui@aol.com', { address: 'Moscow, Russia', phone: '12345678910' })); // Изменение поля 'name' в объекте с 'id' равным 4, возвращает измененный объект (поиск по любому ключу, изменяемые данные должны быть совместимы с DataType (Partial))
console.log('users.update()', users.update('name', 'Leo Tolstoy', { address: 'Yasnaya polyana, Russia' })); // изменение несуществующей записи вернет undefined и ничего не изменится
/**
```

```
 "id": 6,
 "name": "Ivan Petrov",
 "phone": "1-788-618-2639",
 "email": "consequat.dolor@protonmail.net",
 "address": "P.O. Box 769, 5046 Pellentesque, Rd."
}]
*/


console.log('users.delete()', users.delete("name", "Ivan Petrov")); // Удаление объекта с полем name
"Ivan Petrov", возвращает удаленный объект
console.log('users.delete()', users.delete("id", 222));  // Удаление несуществующего объекта
/**
* [LOG]: "users.delete()",  {
 "id": 6,
 "name": "Ivan Petrov",
 "phone": "1-788-618-2639",
 "email": "consequat.dolor@protonmail.net",
 "address": "P.O. Box 769, 5046 Pellentesque, Rd."
}
[LOG]: "users.delete()",  undefined
*/

console.log('users.get()4', users.get());
/**
* [LOG]: "users.get()4",  [{
 "id": 1,
 "name": "Lane Mcdonald",
 "phone": "12345678910",
 "email": "dui@aol.com",
 "address": "Moscow, Russia"
}, {
 "id": 2,
 "name": "Emma Ford",
 "phone": "(472) 855-7514",
 "email": "aliquam.ornare@protonmail.net",
 "address": "P.O. Box 311, 427 Egestas Av."
}, {
 "id": 3,
 "name": "Louis Juarez",
 "phone": "1-895-966-2657",
 "email": "venenatis.lacus@outlook.net",
 "address": "Ap #125-478 Sit Av."
}, {
 "id": 4,
 "name": "Sergey",
 "phone": "1-262-419-4287",
 "email": "arcu.vel@protonmail.net",
 "address": "P.O. Box 631, 1093 Metus Street"
}, {
```

```
 "id": 5,
 "name": "Harriet Acevedo",
 "phone": "1-788-618-2639",
 "email": "consequat.dolor@protonmail.net",
 "address": "P.O. Box 769, 5046 Pellentesque, Rd."
}]
*/

/**
* Набор исходных данных и примеров 2
*/

interface OrderDataType {
   id: number,
   price: number
}
class OrderModel extends Resouce<OrderDataType> {
   constructor(data: OrderDataType[]) {
       super(data)
   }
}

const orders = new OrderModel([{
   id: 1,
   price: 100
}, {
   id: 2,
   price: 200,
}, {
   id: 3,
   price: 300
}]);


console.log('orders.get()', orders.get());
/**
* [LOG]: "orders.get()",  [{
 "id": 1,
 "price": 100
}, {
 "id": 2,
 "price": 200
}, {
```

```
 "id": 3,
 "price": 300
}]
*/

console.log('orders.add()', orders.add({
   id: 4,
   price: 400
}));
/**
* [LOG]: "orders.add()",  [{
 "id": 1,
 "price": 100
}, {
 "id": 2,
 "price": 200
}, {
 "id": 3,
 "price": 300
}, {
 "id": 4,
 "price": 400
}]
*/

console.log('orders.getOne', orders.getOne('id', 1));
console.log('orders.getOne', orders.getOne('price', 200));
/**
* [LOG]: "orders.getOne",  {
 "id": 1,
 "price": 100
}
[LOG]: "orders.getOne",  {
 "id": 2,
 "price": 200
}
*/

console.log('orders.update', orders.update('id', 3, {price: 500}));
/**
* [LOG]: "orders.update",  {
 "id": 3,
 "price": 500
}
*/
```

```
console.log('orders.delete', orders.delete('id', 2));
/**
* [LOG]: "orders.delete",  {
 "id": 2,
 "price": 200
}
*/

console.log('orders.get', orders.get());
/**
* [LOG]: "orders.get",  [{
 "id": 1,
 "price": 100
}, {
 "id": 3,
 "price": 500
}, {
 "id": 4,
 "price": 400
}]
*/

// Абстрактный класс будет здесь




/**
* Набор исходных данных и примеров 1
*/

type DataType = {
   id: number;
   name: string;
   phone: string;
   email: string;
   address: string;
}
const mockUserData: DataType[] = [
   {
       "id": 1,
       "name": "Lane Mcdonald",
       "phone": "1-980-603-4363",
       "email": "dui@aol.com",
       "address": "P.O. Box 895, 4432 Placerat, Avenue",
   },
```

```
    {
        "id": 2,
        "name": "Emma Ford",
        "phone": "(472) 855-7514",
        "email": "aliquam.ornare@protonmail.net",
        "address": "P.O. Box 311, 427 Egestas Av.",
    },
    {
        "id": 3,
        "name": "Louis Juarez",
        "phone": "1-895-966-2657",
        "email": "venenatis.lacus@outlook.net",
        "address": "Ap #125-478 Sit Av.",
    },
    {
        "id": 4,
        "name": "Zorita Mason",
        "phone": "1-262-419-4287",
        "email": "arcu.vel@protonmail.net",
        "address": "P.O. Box 631, 1093 Metus Street",
    },
    {
        "id": 5,
        "name": "Harriet Acevedo",
        "phone": "1-788-618-2639",
        "email": "consequat.dolor@protonmail.net",
        "address": "P.O. Box 769, 5046 Pellentesque, Rd.",
    }
]

class UserModel extends Resouce<DataType> {
    constructor(data: DataType[]) {
        super(data)
    }
}

const users = new UserModel([...mockUserData]);

console.log('users.get()', users.get()); // Получение массива всех объектов из контекста
абстрактного класса
/**
* [LOG]: "users.get()",  [{
 "id": 1,
 "name": "Lane Mcdonald",
 "phone": "1-980-603-4363",
 "email": "dui@aol.com",
 "address": "P.O. Box 895, 4432 Placerat, Avenue"
}, {
 "id": 2,
 "name": "Emma Ford",
 "phone": "(472) 855-7514",
 "email": "aliquam.ornare@protonmail.net",
 "address": "P.O. Box 311, 427 Egestas Av."
}, {
```

```
 "id": 3,
 "name": "Louis Juarez",
 "phone": "1-895-966-2657",
 "email": "venenatis.lacus@outlook.net",
 "address": "Ap #125-478 Sit Av."
}, {
 "id": 4,
 "name": "Zorita Mason",
 "phone": "1-262-419-4287",
 "email": "arcu.vel@protonmail.net",
 "address": "P.O. Box 631, 1093 Metus Street"
}, {
 "id": 5,
 "name": "Harriet Acevedo",
 "phone": "1-788-618-2639",
 "email": "consequat.dolor@protonmail.net",
 "address": "P.O. Box 769, 5046 Pellentesque, Rd."
}]

*/
console.log('users add',users.add({
   "id": 6,
   "name": "Ivan Petrov",
   "phone": "1-788-618-2639",
   "email": "consequat.dolor@protonmail.net",
   "address": "P.O. Box 769, 5046 Pellentesque, Rd.",
})); // Push объекта в массив. Объект может быть только типа Datatype
/**
* [LOG]: "users add",  [{
 "id": 1,
 "name": "Lane Mcdonald",
 "phone": "1-980-603-4363",
 "email": "dui@aol.com",
 "address": "P.O. Box 895, 4432 Placerat, Avenue"
}, {
 "id": 2,
 "name": "Emma Ford",
 "phone": "(472) 855-7514",
 "email": "aliquam.ornare@protonmail.net",
 "address": "P.O. Box 311, 427 Egestas Av."
}, {
 "id": 3,
 "name": "Louis Juarez",
 "phone": "1-895-966-2657",
 "email": "venenatis.lacus@outlook.net",
 "address": "Ap #125-478 Sit Av."
}, {
 "id": 4,
 "name": "Zorita Mason",
 "phone": "1-262-419-4287",
 "email": "arcu.vel@protonmail.net",
 "address": "P.O. Box 631, 1093 Metus Street"
```

```
}, {
 "id": 5,
 "name": "Harriet Acevedo",
 "phone": "1-788-618-2639",
 "email": "consequat.dolor@protonmail.net",
 "address": "P.O. Box 769, 5046 Pellentesque, Rd."
}, {
 "id": 6,
 "name": "Ivan Petrov",
 "phone": "1-788-618-2639",
 "email": "consequat.dolor@protonmail.net",
 "address": "P.O. Box 769, 5046 Pellentesque, Rd."
}]

*/

console.log('users.getOne()', users.getOne('id', 4));  // Получение одного объекта с id: 4 (возможно получение по любому ключу и его значению)
console.log('users.getOne()', users.getOne('name', 'Lane Mcdonald'));  // Получение одного объекта по совпадению name
console.log('users.getOne()', users.getOne('phone', '(472) 855-7514'));  // Получение одного объекта по совпадению phone
console.log('users.getOne()', users.getOne('id', 12));  // Не существующий id, вернется undefined
/**
* [LOG]: "users.getOne()",  {
 "id": 4,
 "name": "Zorita Mason",
 "phone": "1-262-419-4287",
 "email": "arcu.vel@protonmail.net",
 "address": "P.O. Box 631, 1093 Metus Street"
}
[LOG]: "users.getOne()",  {
 "id": 1,
 "name": "Lane Mcdonald",
 "phone": "1-980-603-4363",
 "email": "dui@aol.com",
 "address": "P.O. Box 895, 4432 Placerat, Avenue"
}
[LOG]: "users.getOne()",  {
 "id": 2,
 "name": "Emma Ford",
 "phone": "(472) 855-7514",
 "email": "aliquam.ornare@protonmail.net",
 "address": "P.O. Box 311, 427 Egestas Av."
}
```

```
[LOG]: "users.getOne()",  undefined

*/


console.log('users.update()', users.update('id', 4, { name: 'Sergey' })); // Изменение поля 'name' в объекте с 'id' равным 4, возвращает измененный объект (поиск по любому ключу, изменяемые данные должны быть совместимы с DataType (Partial))
console.log('users.update()', users.update('email', 'dui@aol.com', { address: 'Moscow, Russia', phone: '12345678910' })); // Изменение поля 'name' в объекте с 'id' равным 4, возвращает измененный объект (поиск по любому ключу, изменяемые данные должны быть совместимы с DataType (Partial))
console.log('users.update()', users.update('name', 'Leo Tolstoy', { address: 'Yasnaya polyana, Russia' })); // изменение несуществующей записи вернет undefined и ничего не изменится
/**
* [LOG]: "users.update()",  {
 "id": 4,
 "name": "Sergey",
 "phone": "1-262-419-4287",
 "email": "arcu.vel@protonmail.net",
 "address": "P.O. Box 631, 1093 Metus Street"
}
[LOG]: "users.update()",  {
 "id": 1,
 "name": "Lane Mcdonald",
 "phone": "12345678910",
 "email": "dui@aol.com",
 "address": "Moscow, Russia"
}
[LOG]: "users.update()",  undefined
*/


console.log('users.get()3', users.get());
/**
* [LOG]: "users.get()3",  [{
 "id": 1,
 "name": "Lane Mcdonald",
 "phone": "12345678910",
 "email": "dui@aol.com",
 "address": "Moscow, Russia"
}, {
 "id": 2,
 "name": "Emma Ford",
 "phone": "(472) 855-7514",
 "email": "aliquam.ornare@protonmail.net",
 "address": "P.O. Box 311, 427 Egestas Av."
}, {
 "id": 3,
```

```
* [LOG]: "users.update()",  {
 "id": 4,
 "name": "Sergey",
 "phone": "1-262-419-4287",
 "email": "arcu.vel@protonmail.net",
 "address": "P.O. Box 631, 1093 Metus Street"
}
[LOG]: "users.update()",  {
 "id": 1,
 "name": "Lane Mcdonald",
 "phone": "12345678910",
 "email": "dui@aol.com",
 "address": "Moscow, Russia"
}
[LOG]: "users.update()",  undefined
*/



console.log('users.get()3', users.get());
/**
* [LOG]: "users.get()3",  [{
 "id": 1,
 "name": "Lane Mcdonald",
 "phone": "12345678910",
 "email": "dui@aol.com",
 "address": "Moscow, Russia"
}, {
 "id": 2,
 "name": "Emma Ford",
 "phone": "(472) 855-7514",
 "email": "aliquam.ornare@protonmail.net",
 "address": "P.O. Box 311, 427 Egestas Av."
}, {
 "id": 3,
 "name": "Louis Juarez",
 "phone": "1-895-966-2657",
 "email": "venenatis.lacus@outlook.net",
 "address": "Ap #125-478 Sit Av."
}, {
 "id": 4,
 "name": "Sergey",
 "phone": "1-262-419-4287",
 "email": "arcu.vel@protonmail.net",
 "address": "P.O. Box 631, 1093 Metus Street"
}, {
 "id": 5,
 "name": "Harriet Acevedo",
 "phone": "1-788-618-2639",
 "email": "consequat.dolor@protonmail.net",
 "address": "P.O. Box 769, 5046 Pellentesque, Rd."
}, {
```

```
 "name": "Louis Juarez",
 "phone": "1-895-966-2657",
 "email": "venenatis.lacus@outlook.net",
 "address": "Ap #125-478 Sit Av."
}, {
 "id": 4,
 "name": "Sergey",
 "phone": "1-262-419-4287",
 "email": "arcu.vel@protonmail.net",
 "address": "P.O. Box 631, 1093 Metus Street"
}, {
 "id": 5,
 "name": "Harriet Acevedo",
 "phone": "1-788-618-2639",
 "email": "consequat.dolor@protonmail.net",
 "address": "P.O. Box 769, 5046 Pellentesque, Rd."
}, {
 "id": 6,
 "name": "Ivan Petrov",
 "phone": "1-788-618-2639",
 "email": "consequat.dolor@protonmail.net",
 "address": "P.O. Box 769, 5046 Pellentesque, Rd."
}]
*/


console.log('users.delete()', users.delete("name", "Ivan Petrov")); // Удаление объекта с полем name
"Ivan Petrov", возвращает удаленный объект
console.log('users.delete()', users.delete("id", 222));  // Удаление несуществующего объекта
/**
* [LOG]: "users.delete()",  {
 "id": 6,
 "name": "Ivan Petrov",
 "phone": "1-788-618-2639",
 "email": "consequat.dolor@protonmail.net",
 "address": "P.O. Box 769, 5046 Pellentesque, Rd."
}
[LOG]: "users.delete()",  undefined
*/

console.log('users.get()4', users.get());
/**
* [LOG]: "users.get()4",  [{
 "id": 1,
 "name": "Lane Mcdonald",
 "phone": "12345678910",
 "email": "dui@aol.com",
 "address": "Moscow, Russia"
}, {
 "id": 2,
 "name": "Emma Ford",
```

```
 "phone": "(472) 855-7514",
 "email": "aliquam.ornare@protonmail.net",
 "address": "P.O. Box 311, 427 Egestas Av."
}, {
 "id": 3,
 "name": "Louis Juarez",
 "phone": "1-895-966-2657",
 "email": "venenatis.lacus@outlook.net",
 "address": "Ap #125-478 Sit Av."
}, {
 "id": 4,
 "name": "Sergey",
 "phone": "1-262-419-4287",
 "email": "arcu.vel@protonmail.net",
 "address": "P.O. Box 631, 1093 Metus Street"
}, {
 "id": 5,
 "name": "Harriet Acevedo",
 "phone": "1-788-618-2639",
 "email": "consequat.dolor@protonmail.net",
 "address": "P.O. Box 769, 5046 Pellentesque, Rd."
}]
*/
```

```
/**
* Набор исходных данных и примеров 2
*/

interface OrderDataType {
   id: number,
   price: number
}
class OrderModel extends Resouce<OrderDataType> {
   constructor(data: OrderDataType[]) {
       super(data)
   }
```

```
}

const orders = new OrderModel([{
   id: 1,
   price: 100
}, {
   id: 2,
   price: 200,
}, {
   id: 3,
   price: 300
}]);


console.log('orders.get()', orders.get());
/**
* [LOG]: "orders.get()",  [{
 "id": 1,
 "price": 100
}, {
 "id": 2,
 "price": 200
}, {
 "id": 3,
 "price": 300
}]
*/

console.log('orders.add()', orders.add({
   id: 4,
   price: 400
}));
/**
* [LOG]: "orders.add()",  [{
 "id": 1,
 "price": 100
}, {
 "id": 2,
 "price": 200
}, {
 "id": 3,
 "price": 300
}, {
 "id": 4,
```

```
 "price": 400
}]
*/

console.log('orders.getOne', orders.getOne('id', 1));
console.log('orders.getOne', orders.getOne('price', 200));
/**
* [LOG]: "orders.getOne",  {
 "id": 1,
 "price": 100
}
[LOG]: "orders.getOne",  {
 "id": 2,
 "price": 200
}
*/

console.log('orders.update', orders.update('id', 3, {price: 500}));
/**
* [LOG]: "orders.update",  {
 "id": 3,
 "price": 500
}
*/

console.log('orders.delete', orders.delete('id', 2));
/**
* [LOG]: "orders.delete",  {
 "id": 2,
 "price": 200
}
*/

console.log('orders.get', orders.get());
/**
* [LOG]: "orders.get",  [{
 "id": 1,
 "price": 100
}, {
 "id": 3,
 "price": 500
}, {
 "id": 4,
 "price": 400
}]
*/
```