BasicAlgorith Help Documentation

The project covers the data structure of the seven important sorting algorithm, select, insert, bubbl, merge, Shell, quick and heap sort, Can implement any type of List and array types sort, except String type，for the use of developers and learners to use these seven classic algorithms with ease.

Note：If you use this sort items, use non-basic type, you must implement YouCompare <T> interface to define comparable data.

public enum Sortord

Two parameters ASCE (in ascending order), DESC (descending order);

class SortordStaticWay<T>

Methods:

protected void swap (T [] array, int targer1, int targer2)

The array exchang for targer1 and targer2 data

protected boolean checkType (Class clazz)

Check whether the basic data types, returns true

Known Subclasses:

BubblSort，HeapSort，InsertSort，QuickSort，SelectSort，ShellSort，TwoWayMergeSort

public interface PrintTime

void println(Object[] t,int time);

Method for callback parameter, t is sorted array, time to sort lie number, starting at 1

interface YouCompare<T>

Methods:

boolean compare (T compare2);

Compare the size of the two values, the sort of non-basic data types must implement this method

interface Sort<T>

Methods

```
List<T> sort(List<T> list, Sortord asce,PrintTime    printTime);
    list: set to be sorted
    asce: sorted in ascending or descending order law
    printTime: callback
```

Note: The return type may destroy the original structure, the default is ArrayList, you need update improves

```
T[] sort(T[] array, Sortord asce,PrintTime printTime);
```

```
        array: the array to be sorted
        asce: sorted in ascending or descending order law
        printTime: callback

    ArrayList<T> sort(List<T> arrayList, Sortord asce);
        arrayList: set to be sorted
        asce: sorted in ascending or descending order law

    T[] sort(T[] array, Sortord asce);
        array: the array to be sorted
        asce: sorted in ascending or descending order law
```

Known Implementing Classes：
BubblSort，HeapSort，InsertSort，QuickSort，SelectSort，ShellSort，TwoWayMergeSort

class BubblSort<T>(冒泡排序)

Methods

```
    List<T> sort(List<T> list, Sortord asce,PrintTime    printTime);
        list: set to be sorted
        asce: sorted in ascending or descending order law
        printTime: callback
```

Note: The return type may destroy the original structure, the default is ArrayList, you need update improves

```
    T[] sort(T[] array, Sortord asce,PrintTime printTime);
        array: the array to be sorted
        asce: sorted in ascending or descending order law
        printTime: callback

    ArrayList<T> sort(List<T> arrayList, Sortord asce);
        arrayList: set to be sorted
        asce: sorted in ascending or descending order law

    T[] sort(T[] array, Sortord asce);
        array: the array to be sorted
        asce: sorted in ascending or descending order law
```

class HeapSort<T> (堆排序)

注：参数 printTime 在堆排序中未起到作用

Methods

```
    List<T> sort(List<T> list, Sortord asce,PrintTime    printTime);
        list: set to be sorted
```

```
        asce: sorted in ascending or descending order law
        printTime: callback
```
Note: The return type may destroy the original structure, the default is ArrayList, you need update improves

```
    T[] sort(T[] array, Sortord asce,PrintTime printTime);
        array: the array to be sorted
        asce: sorted in ascending or descending order law
        printTime: callback

    ArrayList<T> sort(List<T> arrayList, Sortord asce);
        arrayList: set to be sorted
        asce: sorted in ascending or descending order law

    T[] sort(T[] array, Sortord asce);
        array: the array to be sorted
        asce: sorted in ascending or descending order law
```

class InsertSort<T> (插入排序)

Methods

```
    List<T> sort(List<T> list, Sortord asce,PrintTime    printTime);
        list: set to be sorted
        asce: sorted in ascending or descending order law
        printTime: callback
```
Note: The return type may destroy the original structure, the default is ArrayList, you need update improves

```
    T[] sort(T[] array, Sortord asce,PrintTime printTime);
        array: the array to be sorted
        asce: sorted in ascending or descending order law
        printTime: callback

    ArrayList<T> sort(List<T> arrayList, Sortord asce);
        arrayList: set to be sorted
        asce: sorted in ascending or descending order law

    T[] sort(T[] array, Sortord asce);
        array: the array to be sorted
        asce: sorted in ascending or descending order law
```

class QuickSort <T> (快速排序)

Methods

```
List<T> sort(List<T> list, Sortord asce,PrintTime    printTime);
    list: set to be sorted
    asce: sorted in ascending or descending order law
    printTime: callback
```

Note: The return type may destroy the original structure, the default is ArrayList, you need update improves

```
T[] sort(T[] array, Sortord asce,PrintTime printTime);
    array: the array to be sorted
    asce: sorted in ascending or descending order law
    printTime: callback

ArrayList<T> sort(List<T> arrayList, Sortord asce);
    arrayList: set to be sorted
    asce: sorted in ascending or descending order law

T[] sort(T[] array, Sortord asce);
    array: the array to be sorted
    asce: sorted in ascending or descending order law
```

class SelectSort<T> (选择排序)

Methods

```
List<T> sort(List<T> list, Sortord asce,PrintTime    printTime);
    list: set to be sorted
    asce: sorted in ascending or descending order law
    printTime: callback
```

Note: The return type may destroy the original structure, the default is ArrayList, you need update improves

```
T[] sort(T[] array, Sortord asce,PrintTime printTime);
    array: the array to be sorted
    asce: sorted in ascending or descending order law
    printTime: callback

ArrayList<T> sort(List<T> arrayList, Sortord asce);
    arrayList: set to be sorted
    asce: sorted in ascending or descending order law

T[] sort(T[] array, Sortord asce);
    array: the array to be sorted
    asce: sorted in ascending or descending order law
```

class ShellSort<T> (希尔排序)

Methods

```
List<T> sort(List<T> list, Sortord asce,PrintTime    printTime);
    list: set to be sorted
    asce: sorted in ascending or descending order law
    printTime: callback
```

Note: The return type may destroy the original structure, the default is ArrayList, you need update improves

```
T[] sort(T[] array, Sortord asce,PrintTime printTime);
    array: the array to be sorted
    asce: sorted in ascending or descending order law
    printTime: callback

ArrayList<T> sort(List<T> arrayList, Sortord asce);
    arrayList: set to be sorted
    asce: sorted in ascending or descending order law

T[] sort(T[] array, Sortord asce);
    array: the array to be sorted
    asce: sorted in ascending or descending order law
```

class TwoWayMergeSort<T> (二路归并排序)

Methods

```
List<T> sort(List<T> list, Sortord asce,PrintTime    printTime);
    list: set to be sorted
    asce: sorted in ascending or descending order law
    printTime: callback
```

Note: The return type may destroy the original structure, the default is ArrayList, you need update improves

```
T[] sort(T[] array, Sortord asce,PrintTime printTime);
    array: the array to be sorted
    asce: sorted in ascending or descending order law
    printTime: callback

ArrayList<T> sort(List<T> arrayList, Sortord asce);
    arrayList: set to be sorted
    asce: sorted in ascending or descending order law

T[] sort(T[] array, Sortord asce);
```

```
        array: the array to be sorted
        asce: sorted in ascending or descending order law
```

代码事例：

```
public class Car implements YouCompare<Car> {

    private int height;
    private int weight;
//省略get和set方法
    public Car(int height, int weight) {
        super();
        this.height = height;
        this.weight = weight;
    }

    public boolean compare( Car compare2) {
        if(this.getHeight()>compare2.getHeight()){
            return true;
        }else if(this.getHeight()<compare2.getHeight()){
            return false;
        }else {
            if(this.getWeight()>=compare2.getWeight()){
                return true;
            }else {
                return false;
            }
        }
    }
}


public static void main(String[] args) {
        ArrayList<Car> arrayList = new ArrayList<Car>();
        arrayList.add(new Car(8, 6));
        arrayList.add(new Car(4, 6));
        arrayList.add(new Car(7, 6));
        arrayList.add(new Car(48, 6));
        arrayList.add(new Car(10, 6));
        arrayList.add(new Car(39, 6));
        arrayList.add(new Car(9, 6));
        arrayList.add(new Car(12, 6));
        arrayList.add(new Car(11, 6));
        arrayList.add(new Car(35, 6));
```

```
        arrayList.add(new Car(40, 6));
        System.out.println("冒泡排序");
        List<Car> newBubblSort = new BubblSort<Car>().sort(arrayList,
                Sortord.ASCE, new PrintTime() {
                    public void println(Object[] t, int time) {
                        // TODO Auto-generated method stub
                        if (t == null) {
                            System.out.println("null");
                            return;
                        }
                        System.out.print("遍历第" + time + "躺: ");
                        for (int i = 0; i < t.length; i++) {
                            System.out.print((Car) t[i] + " ");
                        }
                        System.out.println();
                    }
                });
}
```

作者　游神
电话 15712375939
QQ 727204747
希望互相交流
正在找工作 ing 随便写点东西*.*