ES6+ DESIGN PATTERN VIEW PATTERN

3

1 2 4 5 6

LOOP PATTERN

동일한 구조의 반복

Iteration

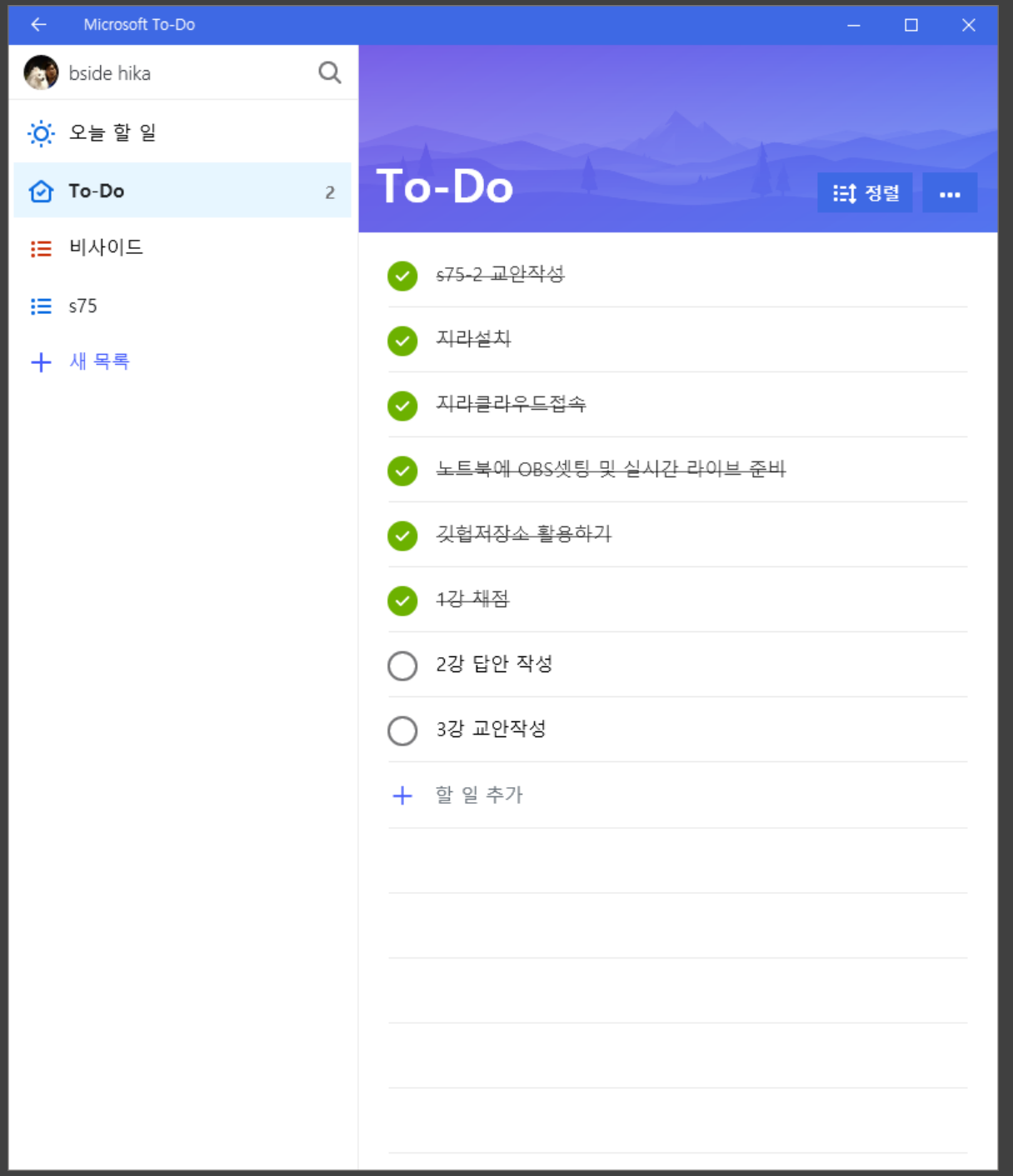
ITERATOR PATTERN

알고리즘 전개에 따른 반복

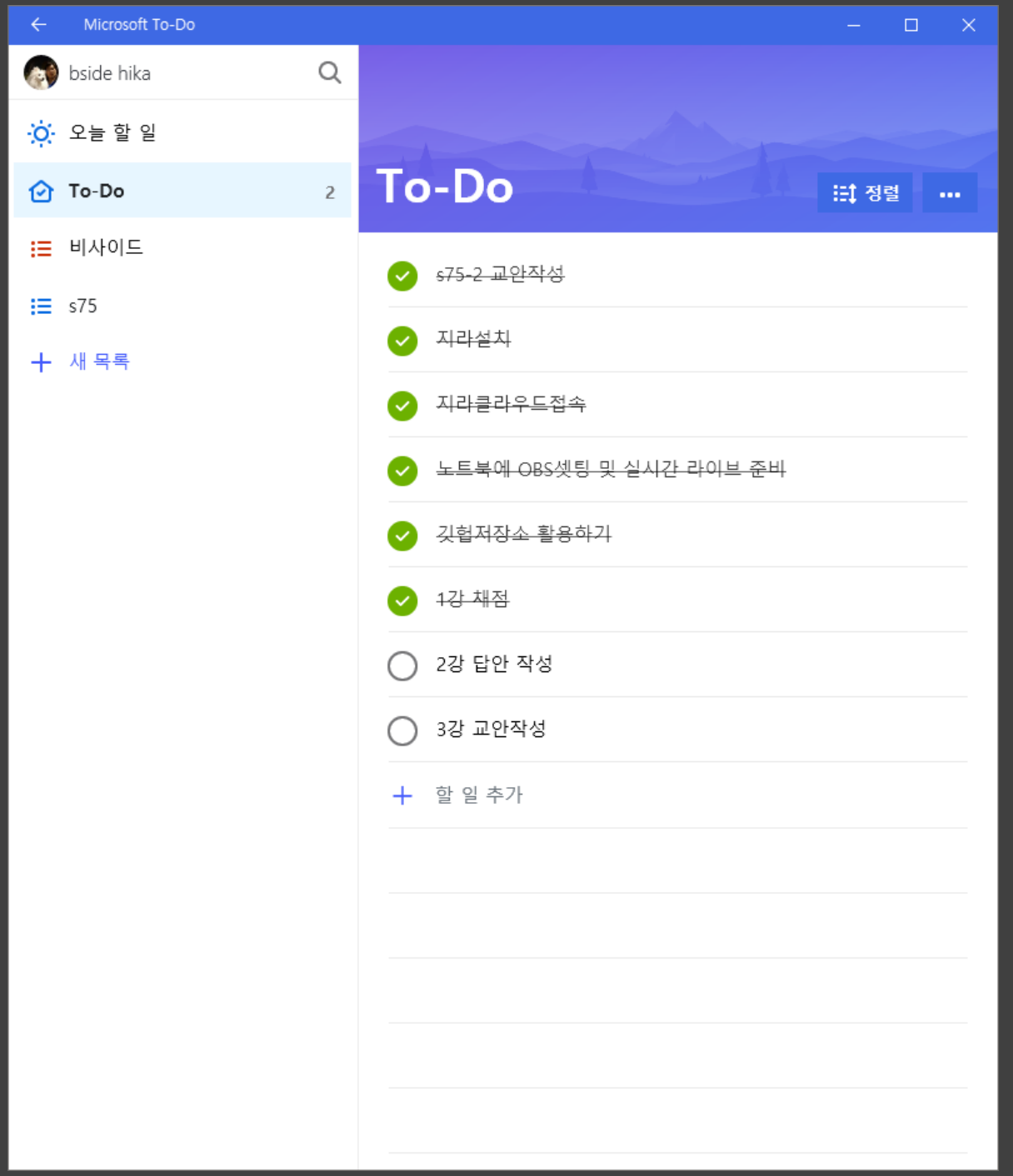
Recursion

COMPOSITE, VISITOR PATTERN

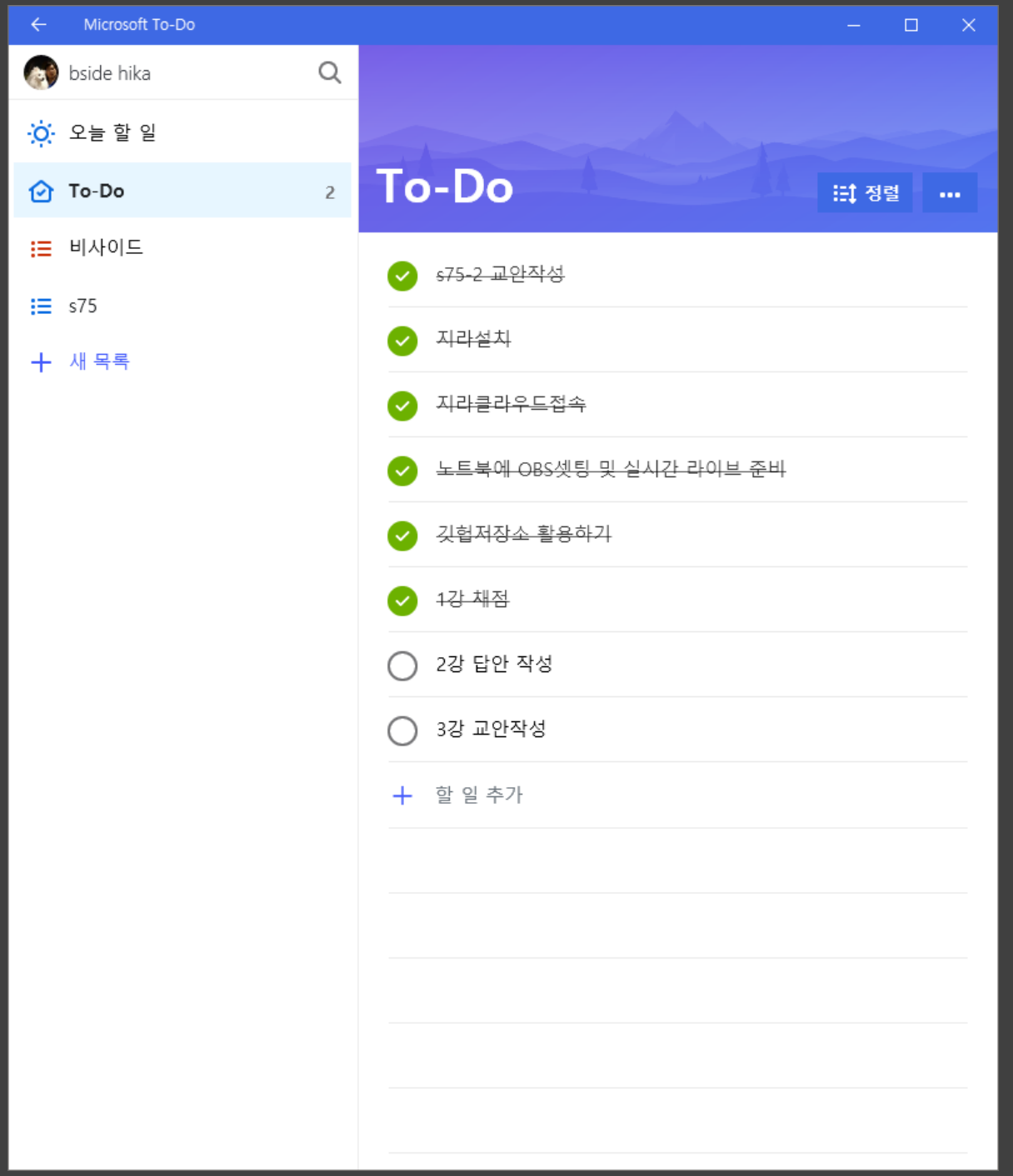
TODO DOMAIN


Microsoft To-Do
bside hika
오늘 할 일
To-Do
2
비사이드
s75
새 목록
To-Do
정렬
s75-2 교안작성
지라설치
지라클라우드접속
노트북에 OBS셋팅 및 실시간 라이브 준비
깃헙저장소 활용하기
1강 채점
2강 답안 작성
3강 교안작성
할 일 추가

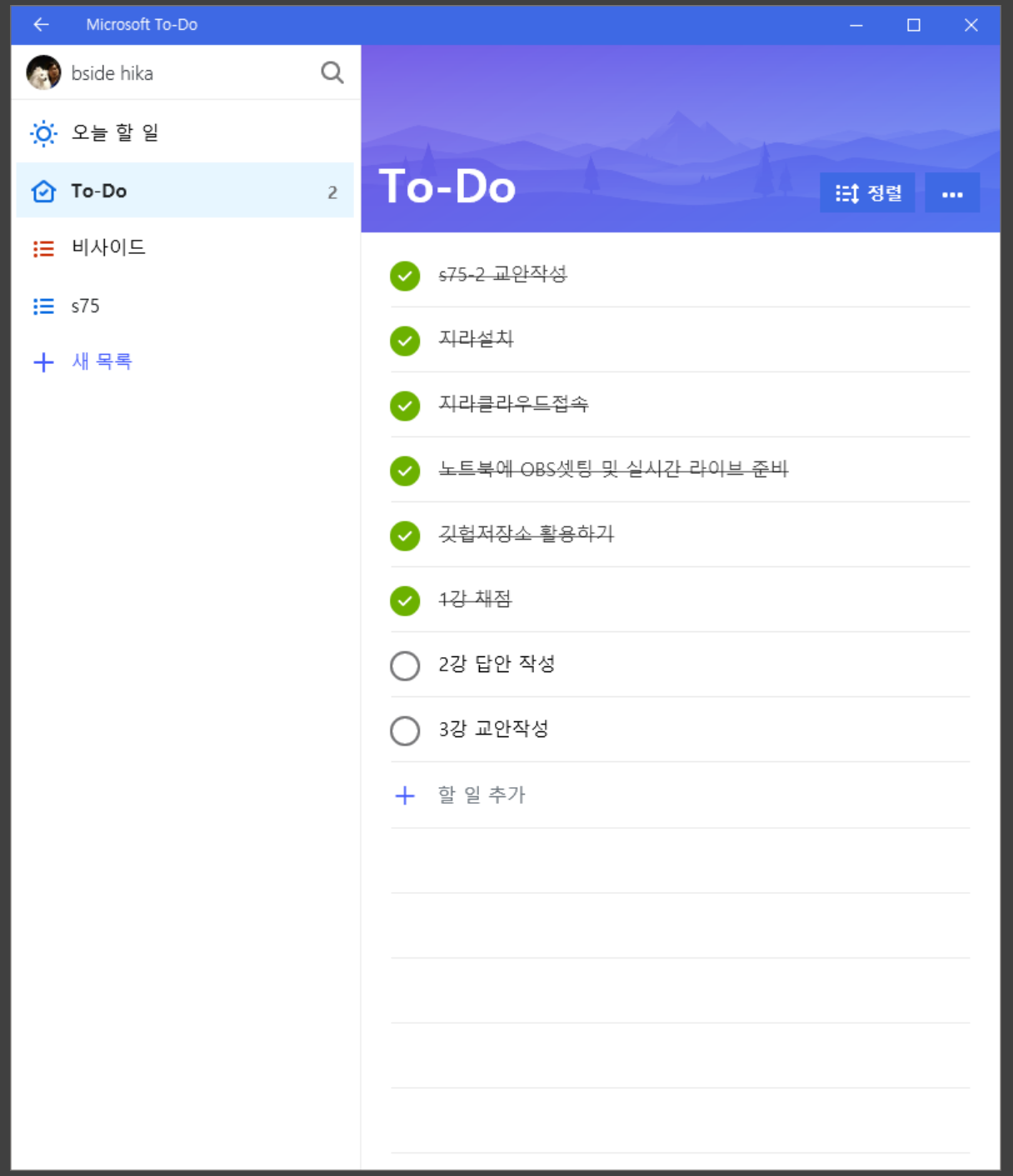
```
const list1 = new TaskList('비사이드');
list1.add("지라설치");
list1.add("지라클라우드접속");

const list2 = new TaskList('s75');
list2.add("2강 답안 작성");
list2.add("3강 교안 작성");

console.log(list1.byRegister());

console.log(list2.byDate());
```



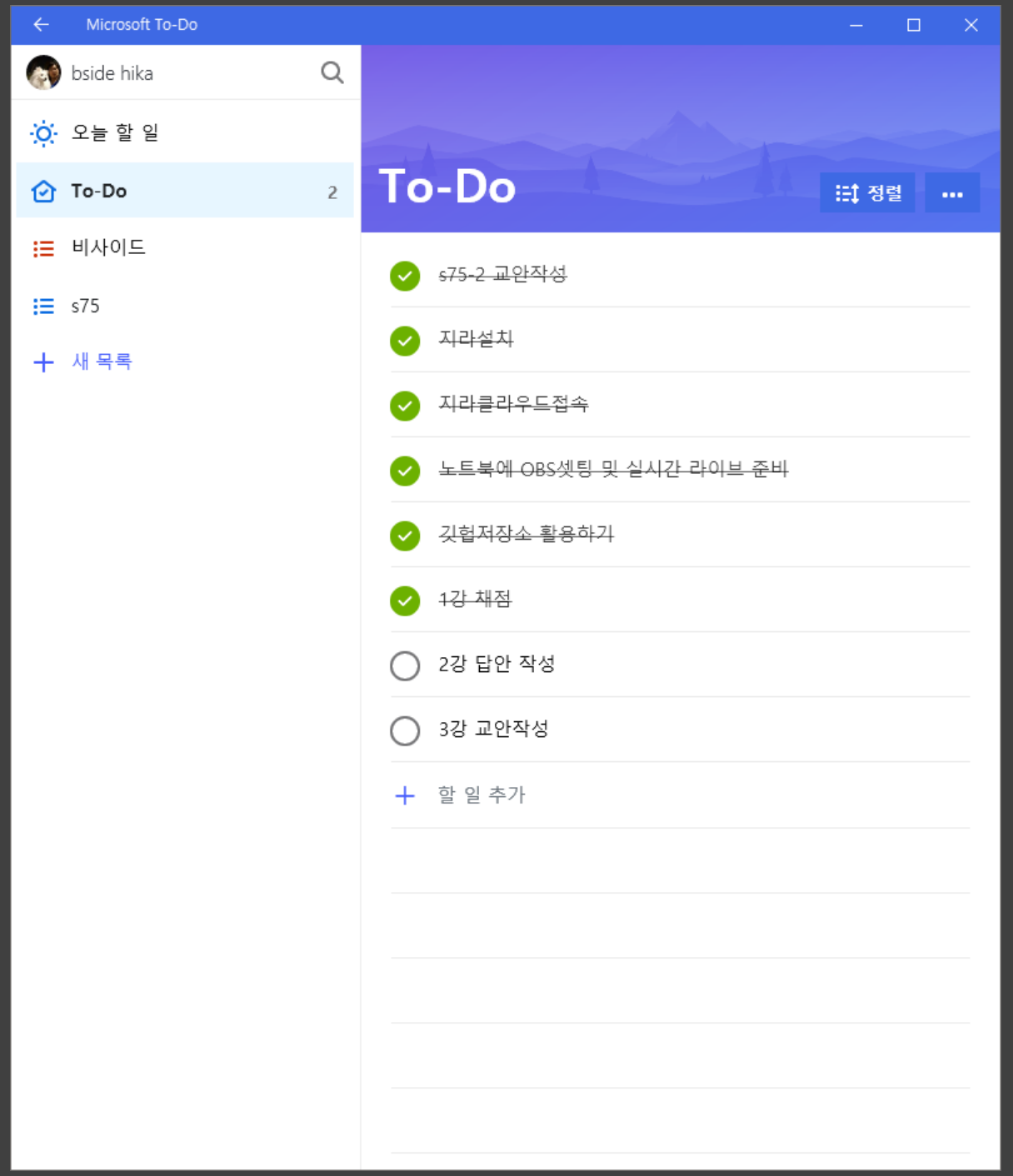
```
const list1 = new TaskList('비사이드');
list1.add("지라설치");
list1.add("지라클라우드접속");

const list2 = new TaskList('s75');
list2.add("2강 답안 작성");
list2.add("3강 교안 작성");

console.log(list1.byRegister());

console.log(list2.byDate());
```



```
const list1 = new TaskList('비사이드');
list1.add("지라설치");
list1.add("지라클라우드접속");

const list2 = new TaskList('s75');
list2.add("2강 답안 작성");
list2.add("3강 교안 작성");

console.log(list1.byRegister());

console.log(list2.byDate());
```



```
const list1 = new TaskList('비사이드');
list1.add("지라설치");
list1.add("지라클라우드접속");

const list2 = new TaskList('s75');
list2.add("2강 답안 작성");
list2.add("3강 교안 작성");

console.log(list1.byTitle());

console.log(list2.byDate());
```



```
const Task = class{
   constructor(title, date){
      if(!title) throw 'invalid title';
      this._title = title;
      this._date = date;
      this._isComplete = false;
   }
   isComplete(){return this._isComplete;}
   toggle(){this._isComplete = !this._isComplete;}
   sortTitle(task){
      return this._title > task._title;
   }
   sortDate(task){
      return this._date > task._date;
   }
};
const taskSort = {
   title:(a, b)=>a.sortTitle(b),
   date:(a, b)=>a.sortDate(b)
};
```

```
const Task = class{
   constructor(title, date){
      if(!title) throw 'invalid title';
      this._title = title;
      this._date = date;
      this._isComplete = false;
   }
   isComplete(){return this._isComplete;}
   toggle(){this._isComplete = !this._isComplete;}
   sortTitle(task){
      return this._title > task._title;
   }
   sortDate(task){
      return this._date > task._date;
   }
};
const taskSort = {
   title:(a, b)=>a.sortTitle(b),
   date:(a, b)=>a.sortDate(b)
};
```

```
const Task = class{
   constructor(title, date){
      if(!title) throw 'invalid title';
      this._title = title;
      this._date = date;
      this._isComplete = false;
   }
   isComplete(){return this._isComplete;}
   toggle(){this._isComplete = !this._isComplete;}
   sortTitle(task){
      return this._title > task._title;
   }
   sortDate(task){
      return this._date > task._date;
   }
};
const taskSort = {
   title:(a, b)=>a.sortTitle(b),
   date:(a, b)=>a.sortDate(b)
};
```

```
const Task = class{
   constructor(title, date){
      if(!title) throw 'invalid title';
      this._title = title;
      this._date = date;
      this._isComplete = false;
   }
   isComplete(){return this._isComplete;}
   toggle(){this._isComplete = !this._isComplete;}
   sortTitle(task){
      return this._title > task._title;
   }
   sortDate(task){
      return this._date > task._date;
   }
};
const taskSort = {
   title:(a, b)=>a.sortTitle(b),
   date:(a, b)=>a.sortDate(b),
};
```

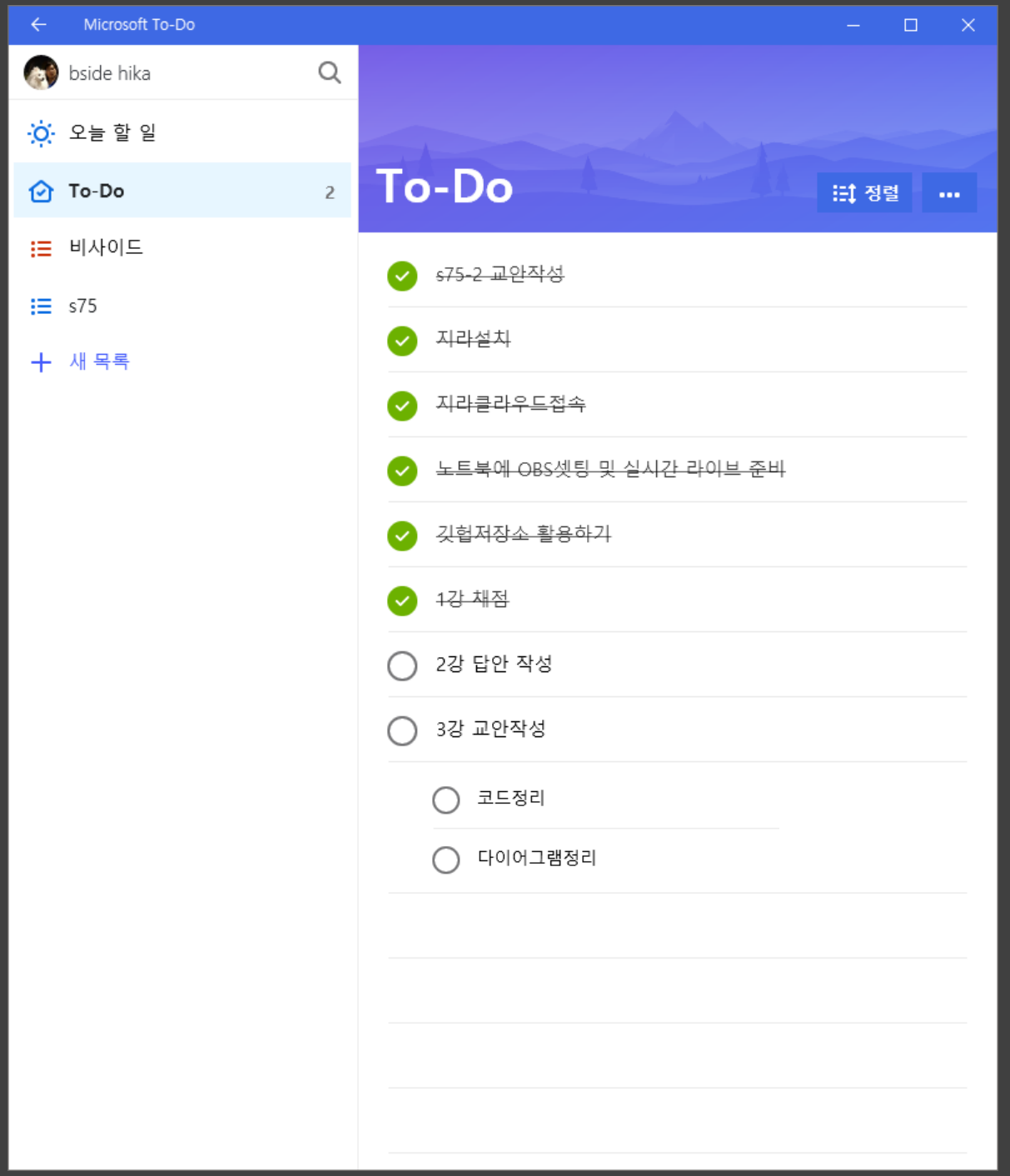
```
const TaskList = class{
   constructor(title){
      if(!title) throw 'invalid title';
      this._title = title;
      this._list = [];
   }
   add(title, date = Date.now()){this._list.push(new Task(title, date));}
   remove(task){
      const list = this._list;
      if(list.includes(task)) list.splice(list.indexOf(task), 1);
   }
   byTitle(stateGroup = true){return this._getList('title', stateGroup);}
   byDate(stateGroup = true){return this._getList('date', stateGroup);}
   _getList(sort, stateGroup){
      const list = this._list, s = taskSort[sort];
      return !stateGroup ? [...list].sort(s) : [
         ...list.filter(v=>!v.isComplete()).sort(s),
         ...list.filter(v=>v.isComplete()).sort(s)
      ];
   }
};
```

```
const TaskList = class{
   constructor(title){
      if(!title) throw 'invalid title';
      this._title = title;
      this._list = [];
   }
   add(title, date = Date.now()){this._list.push(new Task(title, date));}
   remove(task){
      const list = this._list;
      if(list.includes(task)) list.splice(list.indexOf(task), 1);
   }
   byTitle(stateGroup = true){return this._getList('title', stateGroup);}
   byDate(stateGroup = true){return this._getList('date', stateGroup);}
   _getList(sort, stateGroup){
      const list = this._list, s = taskSort[sort];
      return !stateGroup ? [...list].sort(s) : [
         ...list.filter(v=>!v.isComplete()).sort(s),
         ...list.filter(v=>v.isComplete()).sort(s)
      ];
   }
};
```

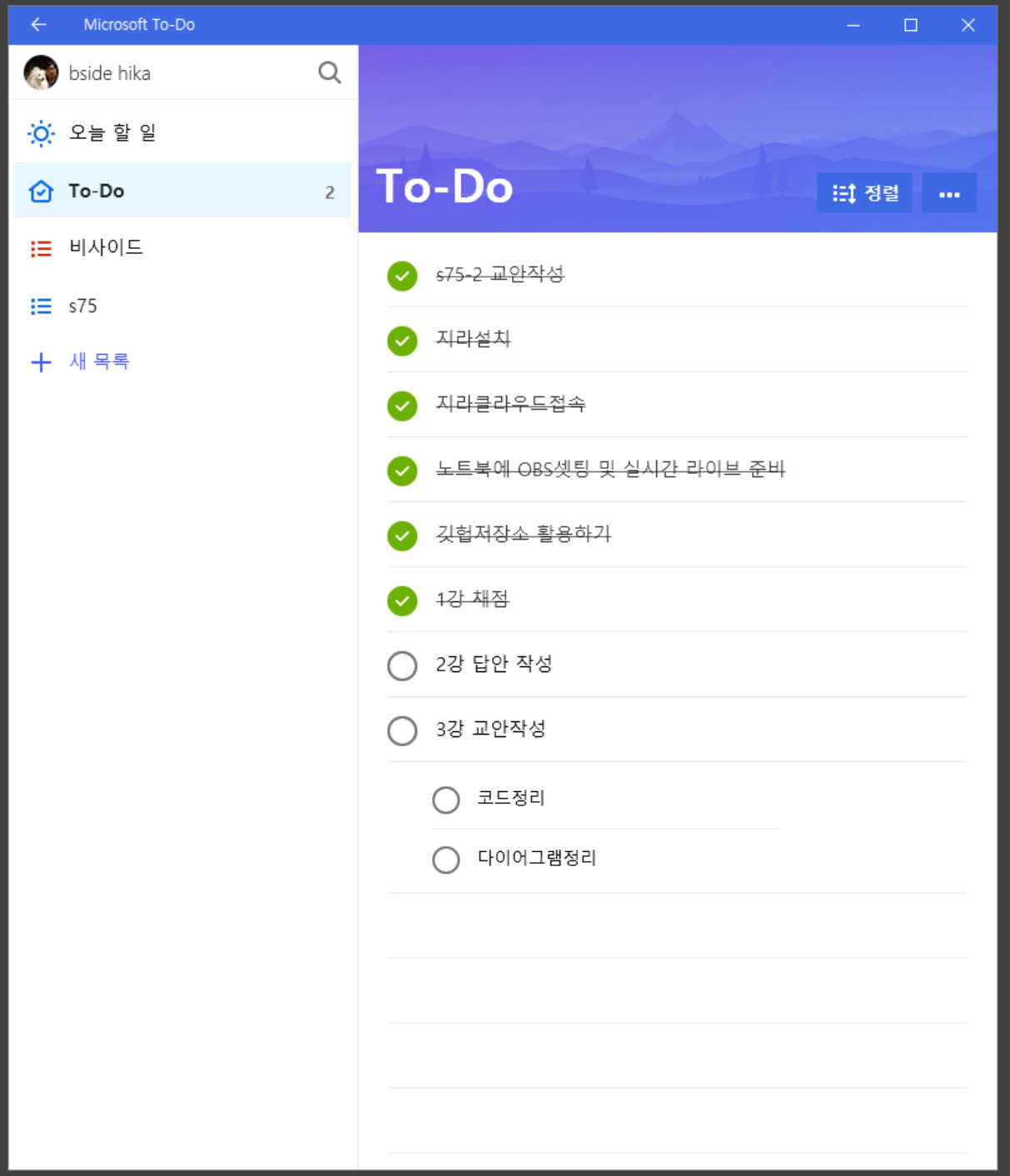
```
const TaskList = class{
   constructor(title){
      if(!title) throw 'invalid title';
      this._title = title;
      this._list = [];
   }
   add(title, date = Date.now()){this._list.push(new Task(title, date));}
   remove(task){
      const list = this._list;
      if(list.includes(task)) list.splice(list.indexOf(task), 1);
   }
   byTitle(stateGroup = true){return this._getList('title', stateGroup);}
   byDate(stateGroup = true){return this._getList('date', stateGroup);}
   _getList(sort, stateGroup){
      const list = this._list, s = taskSort[sort];
      return !stateGroup ? [...list].sort(s) : [
         ...list.filter(v=>!v.isComplete()).sort(s),
         ...list.filter(v=>v.isComplete()).sort(s)
      ];
   }
};
```

TODO SUB ITEM

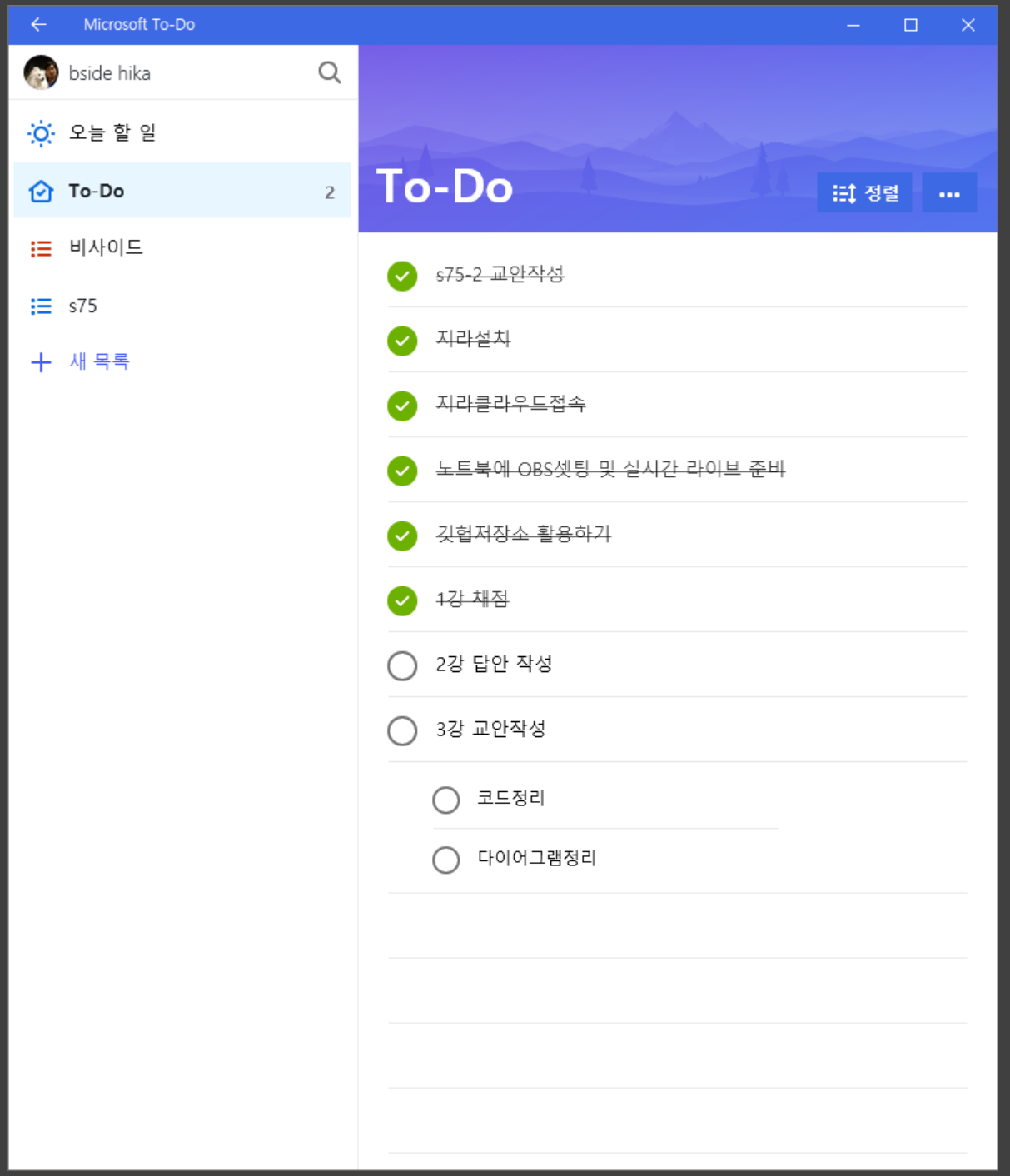
```
const list1 = new TaskList('비사이드');
list1.add("지라설치");
list1.add("지라클라우드접속");

const list2 = new TaskList('s75');
list2.add("2강 답안 작성");
list2.add("3강 교안 작성");


const list = list2.byDate();
list[1].task.add("코드정리");
list[1].task.add("다이어그램정리");

console.log(list2.byDate()[1].sub);
```



```
const list1 = new TaskList('비사이드');
list1.add("지라설치");
list1.add("지라클라우드접속");

const list2 = new TaskList('s75');
list2.add("2강 답안 작성");
list2.add("3강 교안 작성");


const list = list2.byDate();
list[1].task.add("코드정리");
list[1].task.add("다이어그램정리");

console.log(list2.byDate()[1].sub);
```



```
const list1 = new TaskList('비사이드');
list1.add("지라설치");
list1.add("지라클라우드접속");

const list2 = new TaskList('s75');
list2.add("2강 답안 작성");
list2.add("3강 교안 작성");


const list = list2.byDate();
list[1].task.add("코드정리");
list[1].task.add("다이어그램정리");

console.log(list2.byDate()[1].sub);
```



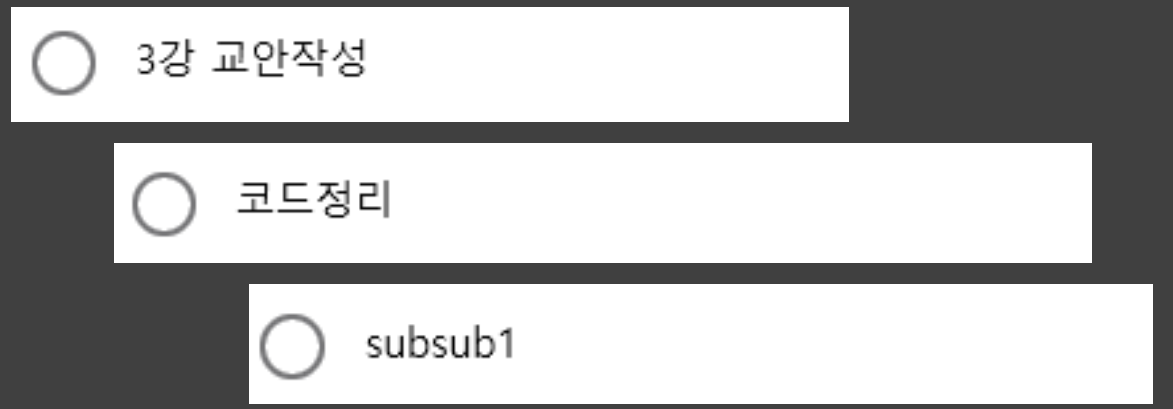
```
const Task = class{
    constructor(title, date){
        if(!title) throw 'invalid title';
        this._title = title;
        this._date = date;
        this._isComplete = false;
        this._list = [];
    }
    add(title, date = Date.now()){this._list.push(new Task(title, date));}
    remove(task){
        const list = this._list;
        if(list.includes(task)) list.splice(list.indexOf(task), 1);
    }
    _getList(sort, stateGroup){
        const list = this._list, s = taskSort[sort];
        return {
            task:this
            sub:!stateGroup ? [...list].sort(s) : [
                ...list.filter(v=>!v.isComplete()).sort(s),
                ...list.filter(v=>v.isComplete()).sort(s)
            ]
        };
    }
    isComplete(){return this._isComplete;}
    toggle(){this._isComplete = !this._isComplete;}
    sortTitle(task){return this._title > task._title;}
    sortDate(task){return this._date > task._date;}
};
```

```
const TaskList = class{
   constructor(title){
      if(!title) throw 'invalid title';
      this._title = title;
      this._list = [];
   }
   add(title, date = Date.now()){this._list.push(new Task(title, date));}
   remove(task){
      const list = this._list;
      if(list.includes(task)) list.splice(list.indexOf(task), 1);
   }
   byTitle(stateGroup = true){return this._getList('title', stateGroup);}
   byDate(stateGroup = true){return this._getList('date', stateGroup);}
   _getList(sort, stateGroup){
      const list = this._list, s = taskSort[sort];
      return (!stateGroup ? [...list].sort(s) : [
         ...list.filter(v=>!v.isComplete()).sort(s),
         ...list.filter(v=>v.isComplete()).sort(s)
      ]).map(v=>v._getList());
   }
};
```

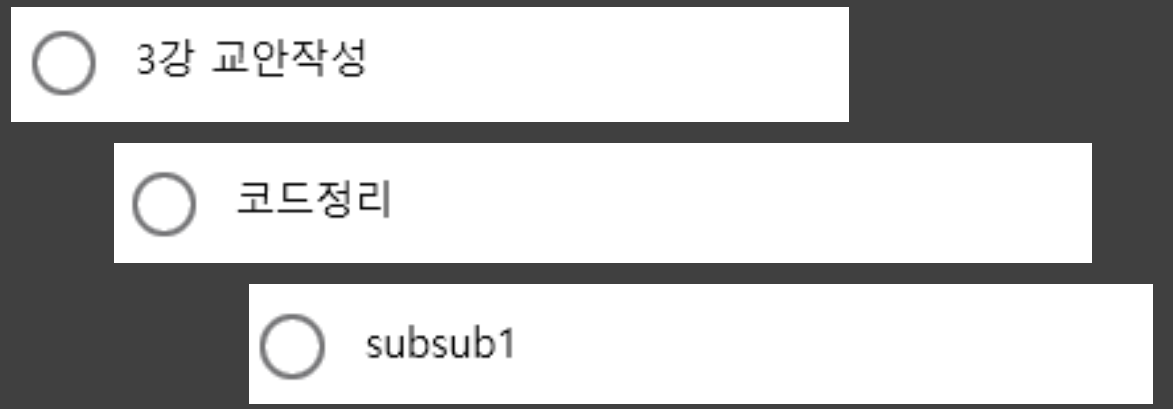
COMPOSITE

- 3강 교안작성
- 코드정리
  - subsub1

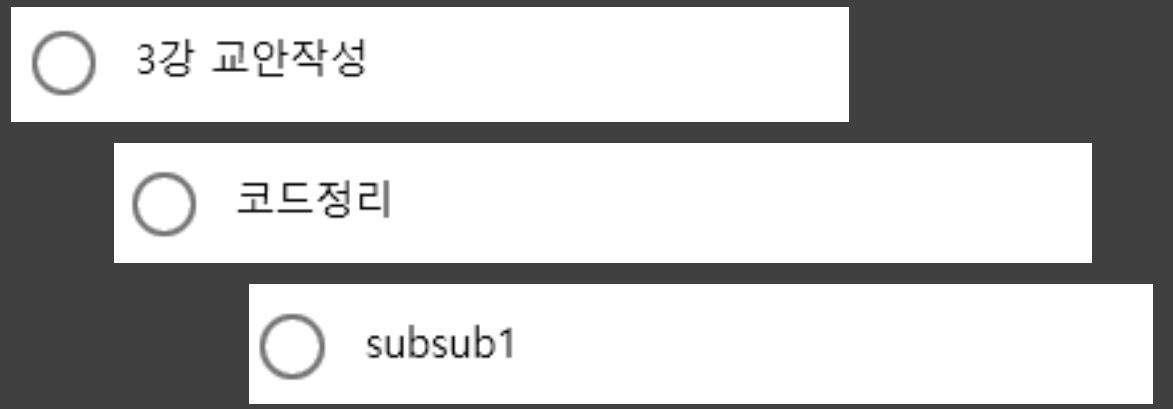
```
const list1 = new TaskList('s75');
const item1 = new TaskItem("3강교안작성");
list1.add(item1);
const sub1 = new TaskItem("코드정리");
item1.add(sub1);
const subsub1 = new TaskItem("subsub1");
sub1.add(subsub1);

list1.getResult(Task.title);

{item:'s75',
 children:[
   {item:taskItem('3강교안작성'),
    children:[
      {item:taskItem('코드정리'),
       children:[
         {item:taskItem('subsub1'),
          children:[]}
       ]}
    ]}
 ]
}
```



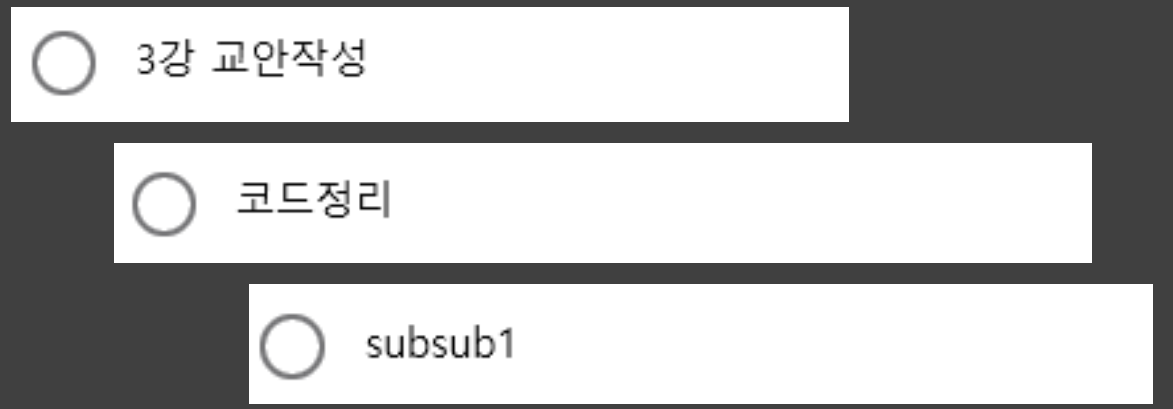
```
const list1 = new TaskList('s75');
const item1 = new TaskItem("3강교안작성");
list1.add(item1);
const sub1 = new TaskItem("코드정리");
item1.add(sub1);
const subsub1 = new TaskItem("subsub1");
sub1.add(subsub1);

list1.getResult(Task.title);

{item:'s75',
 children:[
   {item:taskItem('3강교안작성'),
    children:[
      {item:taskItem('코드정리'),
       children:[
         {item:taskItem('subsub1'),
          children:[]}
       ]}
    ]}
 ]
}
```



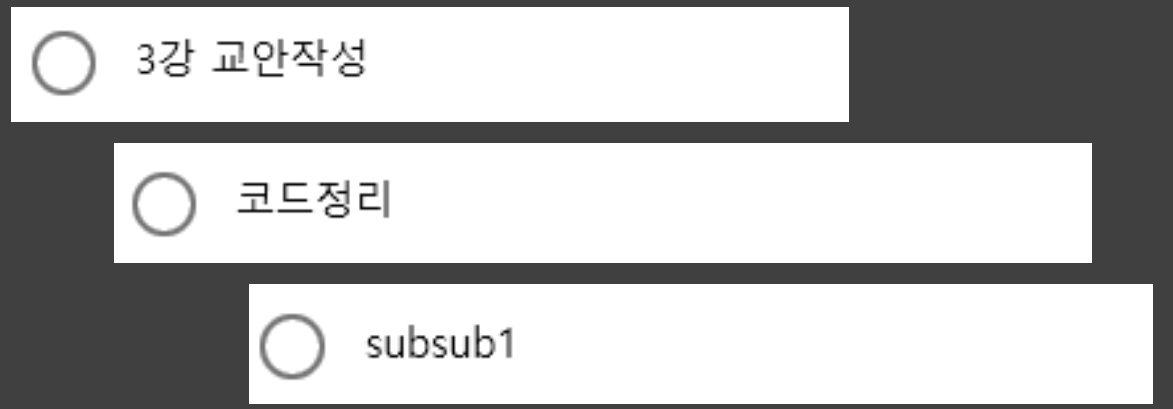
```
const list1 = new TaskList('s75');
const item1 = new TaskItem("3강교안작성");
list1.add(item1);
const sub1 = new TaskItem("코드정리");
item1.add(sub1);
const subsub1 = new TaskItem("subsub1");
sub1.add(subsub1);

list1.getResult(Task.title);

{item:'s75',
 children:[
   {item:taskItem('3강교안작성'),
    children:[
      {item:taskItem('코드정리'),
       children:[
         {item:taskItem('subsub1'),
          children:[]}
       ]}
    ]}
 ]
}
```



```
const list1 = new TaskList('s75');
const item1 = new TaskItem("3강교안작성");
list1.add(item1);
const sub1 = new TaskItem("코드정리");
item1.add(sub1);
const subsub1 = new TaskItem("subsub1");
sub1.add(subsub1);

list1.getResult(Task.title);

{item:'s75',
 children:[
   {item:taskItem('3강교안작성'),
    children:[
      {item:taskItem('코드정리'),
       children:[
         {item:taskItem('subsub1'),
          children:[]}
       ]}
    ]}
 ]
}
```

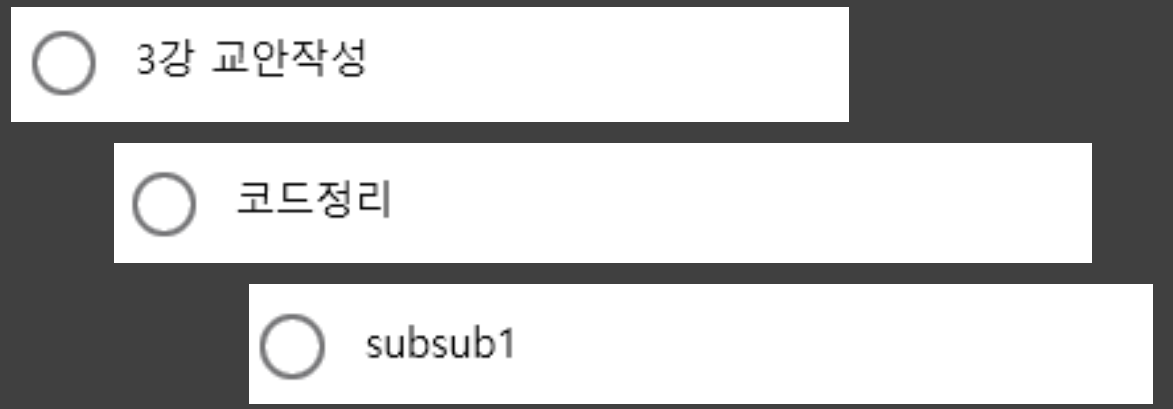


```
const list1 = new TaskList('s75');
const item1 = new TaskItem("3강교안작성");
list1.add(item1);
const sub1 = new TaskItem("코드정리");
item1.add(sub1);
const subsub1 = new TaskItem("subsub1");
sub1.add(subsub1);

list1.getResult(Task.title);
```

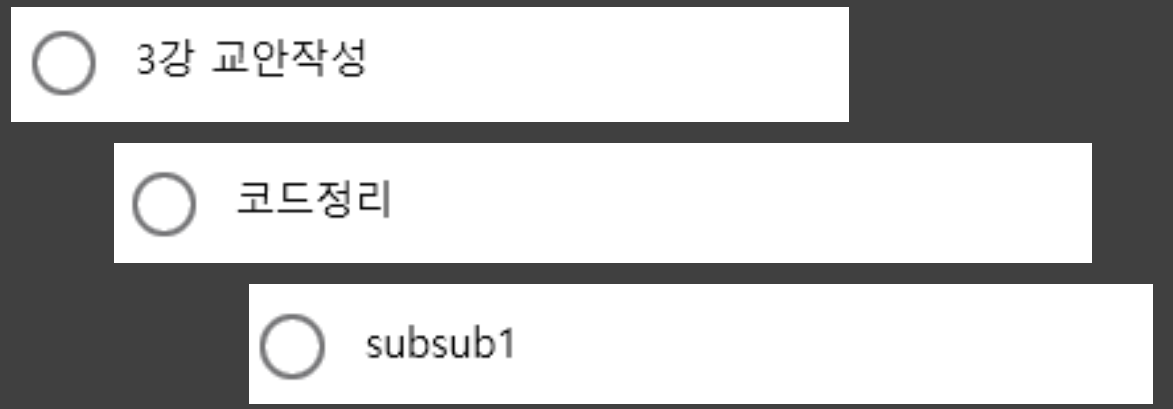



```
{item:'s75',
 children:[
   {item:taskItem('3강교안작성'),
    children:[
      {item:taskItem('코드정리'),
       children:[
         {item:taskItem('subsub1'),
          children:[]}
       ]}
    ]}
 ]
}
```

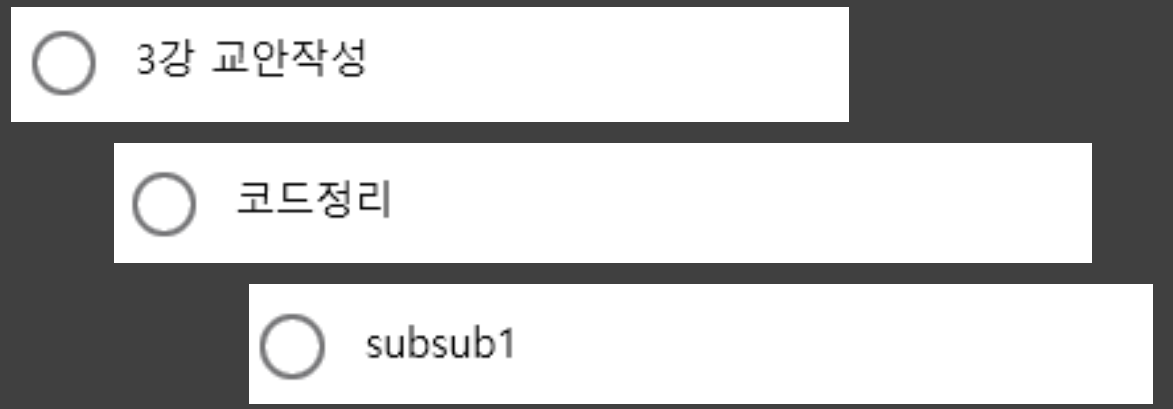
```
const list1 = new TaskList('s75');
const item1 = new TaskItem("3강교안작성");
list1.add(item1);
const sub1 = new TaskItem("코드정리");
item1.add(sub1);
const subsub1 = new TaskItem("subsub1");
sub1.add(subsub1);

list1.getResult(Task.title);

{item:'s75',
 children:[
   {item:taskItem('3강교안작성'),
    children:[
      {item:taskItem('코드정리'),
       children:[
         {item:taskItem('subsub1'),
          children:[]}
       ]}
    ]}
 ]
}
```



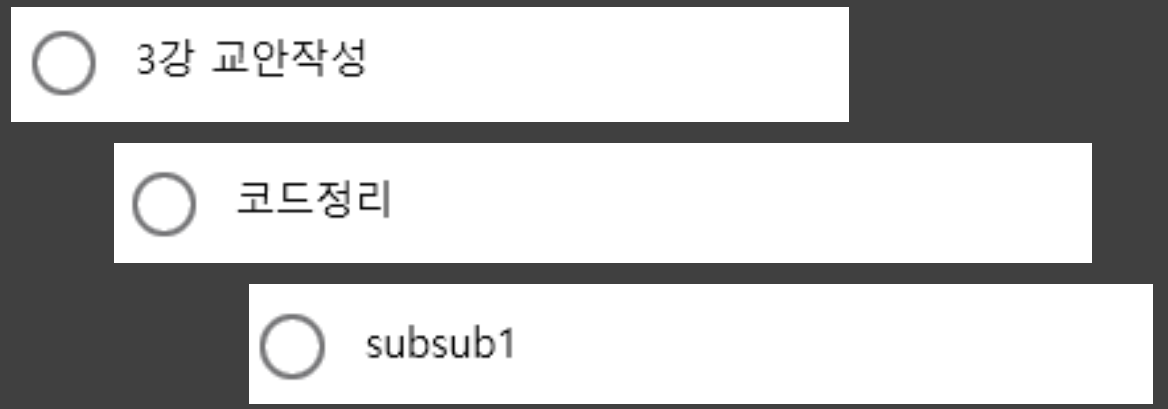
```
const list1 = new TaskList('s75');
const item1 = new TaskItem("3강교안작성");
list1.add(item1);
const sub1 = new TaskItem("코드정리");
item1.add(sub1);
const subsub1 = new TaskItem("subsub1");
sub1.add(subsub1);

list1.getResult(Task.title);

{item:'s75',
 children:[
   {item:taskItem('3강교안작성'),
    children:[
      {item:taskItem('코드정리'),
       children:[
         {item:taskItem('subsub1'),
          children:[]}
       ]}
    ]}
 ]
}
```



```
const list1 = new TaskList('s75');
const item1 = new TaskItem("3강교안작성");
list1.add(item1);
const sub1 = new TaskItem("코드정리");
item1.add(sub1);
const subsub1 = new TaskItem("subsub1");
sub1.add(subsub1);

list1.getResult(Task.title);

{item:'s75',
 children:[
   {item:taskItem('3강교안작성'),
    children:[
      {item:taskItem('코드정리'),
       children:[
         {item:taskItem('subsub1'),
          children:[]}
       ]}
    ]}
 ]
}
```



```
const list1 = new TaskList('s75');
const item1 = new TaskItem("3강교안작성");
list1.add(item1);
const sub1 = new TaskItem("코드정리");
item1.add(sub1);
const subsub1 = new TaskItem("subsub1");
sub1.add(subsub1);

list1.getResult(Task.title);

{item:'s75',
 children:[
   {item:taskItem('3강교안작성'),
    children:[
      {item:taskItem('코드정리'),
       children:[
         {item:taskItem('subsub1'),
          children:[]}
       ]}
    ]}
 ]
}
```



```
const Task = class{
    static title(a, b){return a.sortTitle(b);}
    static date(a, b){return a.sortDate(b);}

    constructor(title){
         if(!title) throw 'invalid title'; else this._title = title;
        this._list = [];
    }
    add(task){if(task instanceof Task) this._list.push(task); else throw 'invalid';}
    remove(task){
        const list = this._list;
        if(list.includes(task)) list.splice(list.indexOf(task), 1);
    }
    getResult(sort, stateGroup){
        const list = this._list;
        return {
            item:this._getResult(),
            children:(!stateGroup ? [...list].sort(sort) : [
                ...list.filter(v=>!v.isComplete()).sort(sort),
                ...list.filter(v=>v.isComplete()).sort(sort)
            ]).map(v=>v.getResult(sort, stateGroup))
        };
    }
    _getResult(){throw 'override';}
    isComplete(){throw 'override';}
    sortTitle(){throw 'override';}
    sortDate(){throw 'override';}
};
```

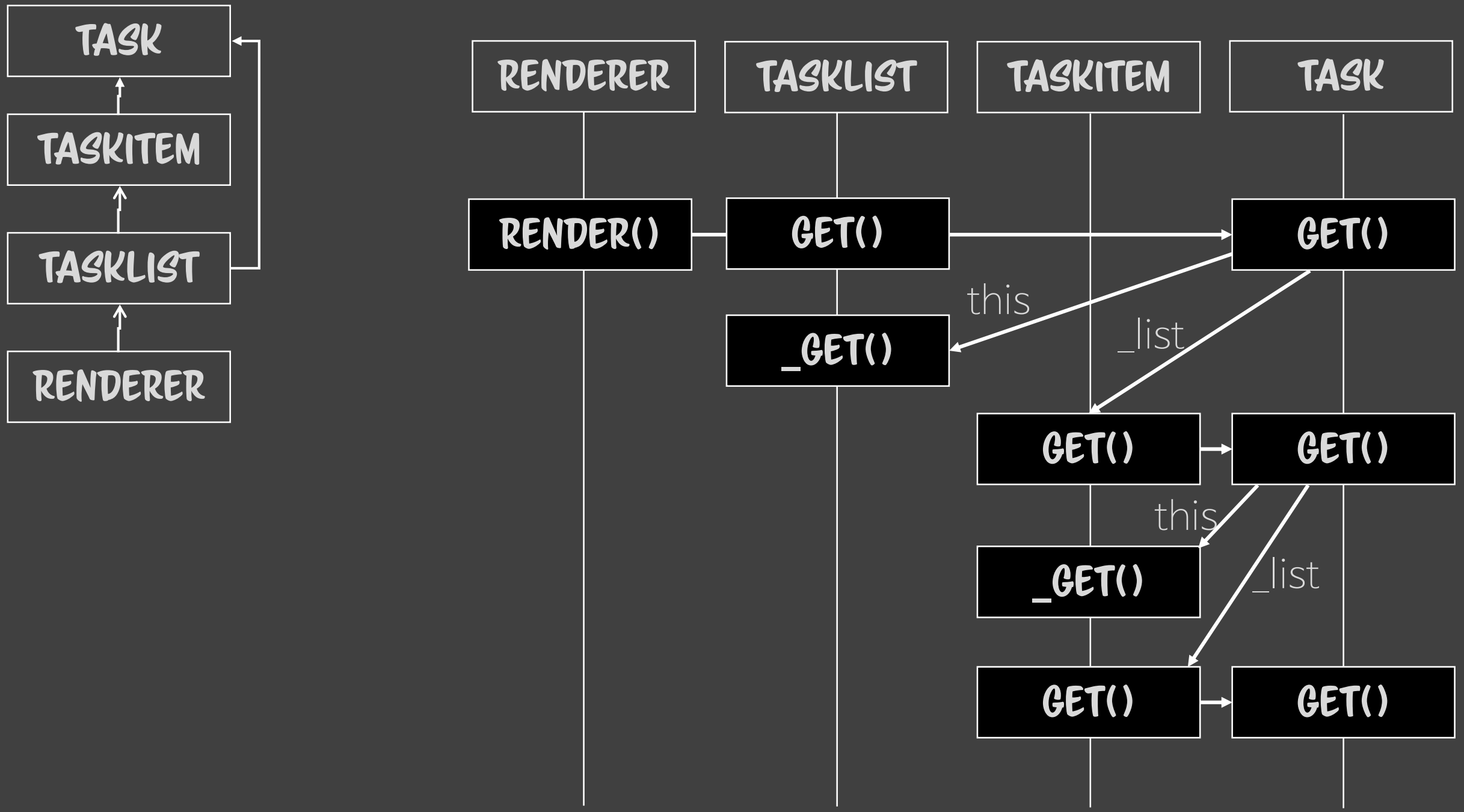
```
const TaskItem = class extends Task{
  constructor(title, date = Date.now()){
    super(title);
    this._date = date;
    this._isComplete = false;
  }
  _getResult(sort, stateGroup){return this;}
  isComplete(){return this._isComplete;}
  sortTitle(task){return this._title > task._title;}
  sortDate(task){return this._date > task._date;}

  toggle(){this._isComplete = !this._isComplete;}
};
```

```
const Task = class{
   ...
   _getResult(){throw 'override';}
   isComplete(){throw 'override';}
   sortTitle(){throw 'override';}
   sortDate(){throw 'override';}
};
```

```
const TaskList = class extends Task{
   constructor(title){super(title);}
   _getResult(){return this._title;}
   isComplete(){}
   sortTitle(){return this;}
   sortDate(){return this;}

   byTitle(stateGroup = true){return this.getResult(Task.title, stateGroup);}
   byDate(stateGroup = true){return this.getResult(Task.date, stateGroup);}
};
```

```
const Task = class{
   ...
   _getResult(){throw 'override';}
   isComplete(){throw 'override';}
   sortTitle(){throw 'override';}
   sortDate(){throw 'override';}
};
```

COMPOSITION RENDERING


TASK
TASKITEM
TASKLIST
RENDERER
RENDERER
TASKLIST
TASKITEM
TASK
RENDER()
GET()
GET()
this
_list
_GET()
GET()
GET()
this
_GET()
_list
GET()
GET()

```
const el =(tag, ...attr)=>{
   const el = document.createElement(tag);
   for(let i = 0; i < attr.length;){
      const k = attr[i++], v = attr[i++];
      if(typeof el[k] == 'function') el[k](...(Array.isArray(v) ? v : [v]));
      else if(k[0] == '@') el.style[k.substr(1)] = v;
      else el[k] = v;
   }
   return el;
};
```

```
const Task = class{
    static title(a, b){return a.sortTitle(b);}
    static date(a, b){return a.sortDate(b);}

    constructor(title){
         if(!title) throw 'invalid title'; else this._title = title;
        this._list = [];
    }
    add(task){if(task instanceof Task) this._list.push(task); else throw 'invalid';}
    remove(task){
        const list = this._list;
        if(list.includes(task)) list.splice(list.indexOf(task), 1);
    }
    getResult(sort, stateGroup){
        const list = this._list;
        return {
            item:this,
            children:(!stateGroup ? [...list].sort(sort) : [
                ...list.filter(v=>!v.isComplete()).sort(sort),
                ...list.filter(v=>v.isComplete()).sort(sort)
             ]).map(v=>v.getResult(sort, stateGroup))
      };
    }
    isComplete(){throw 'override';}
    sortTitle(){throw 'override';}
    sortDate(){throw 'override';}
};
```

```
const TaskItem = class extends Task{
    constructor(title, date = new Date()){
        super(title);
        this._date = date;
        this._isComplete = false;
    }
    isComplete(){return this._isComplete;}
    sortTitle(task){return this._title > task._title;}
    sortDate(task){return this._date > task._date;}

    toggle(){this._isComplete = !this._isComplete;}
};

const TaskList = class extends Task{
    constructor(title){super(title);}
    isComplete(){}
    sortTitle(){return this;}
    sortDate(){return this;}
};
```

```
const DomRenderer = class{
    constructor(list, parent){
        this._parent = parent;
        this._list = list;
        this._sort = 'title';
    }
    add(parent, title, date){
        parent.add(new TaskItem(title, date));
        this.render();
    }
    remove(parent, task){
        parent.remove(task);
        this.render();
    }
    toggle(task){
        if(task instanceof TaskItem){
            task.toggle();
            this.render();
        }
    }
```

```
render(){
    const parent = this._parent;
    parent.innerHTML = '';
    parent.appendChild('title,date'.split(',').reduce((nav,c)=>(
        nav.appendChild(
            el('button', 'innerHTML', c,
                '@fontWeight', this._sort == c ? 'bold' : 'normal',
                'addEventListener', ['click', e=>(this._sort = Task[c], this.render())])
            ), nav
    ), el('nav')));
    this._render(parent, this._list, this._list.getResult(this._sort), 0);
}
```

```
_render(base, parent, {item, children}, depth){
    const temp = [];
    base.style.paddingLeft = depth * 10 + 'px';
    if(item instanceof TaskList){
        temp.push(el('h2', 'innerHTML', item._title));
    }else{
        temp.push(
            el('h3', 'innerHTML', item._title,
                '@textDecoration', item.isComplete() ? 'line-through' : 'none'),
            el('time', 'innerHTML', item._date.toString(), 'datetime', item._date.toString()),
            el('button', 'innerHTML', item.isComplete() ? 'progress' : 'complete',
                'addEventListener', ['click', _=>this.toggle(item)]
            ),
            el('button', 'innerHTML', 'remove',
                'addEventListener', ['click', _=>this.remove(parent, item)]
            )
        )
    }
    const sub = el('section',
        'appendChild', el('input', 'type', 'text'),
        'appendChild', el('button', 'innerHTML', 'addTask',
        'addEventListener', ['click', e=>this.add(item, e.target.previousSibling.value)])
    );
    children.forEach(v=>{this._render(sub, item, v, depth + 1)});
    temp.push(sub);
    temp.forEach(v=>base.appendChild(v));
}
```

```
const list1 = new TaskList('s75');
const item1 = new TaskItem("3강교안작성");
list1.add(item1);
const sub1 = new TaskItem("코드정리");
item1.add(sub1);
const subsub1 = new TaskItem("subsub1");
sub1.add(subsub1);

list1.getResult(Task.title);

const todo = new DomRenderer(list1, sel('#todo'));
todo.render();
```

PRACTICE #1

DomRenderer대신 ConsoleRenderer를 작성하고 둘의 공통점을 모아 추상 Renderer도 작성한다.

PRACTICE #2

DomRenderer는 코드로 HTML을 생성하기 뷰 업데이트가 어렵다.
template태그를 사용하여 뷰를 분리하여 작성하라.