**Category**

*class* Category {

constructor({ id, name = '', keywords = '' }) {

this.id = id;

this.name = name;

this.keywords = keywords;

this.parentCategoryId = null;

this.parent = null;

this.children = [];

}

add(node) {

node.parent = this;

node.parentCategoryId = this.id;

this.children.push(node);

}

}

export {Category};

**Categories**

*class* Categories {

constructor(rootNode) {

this.root= rootNode;

}

FindByCategoryId(id) {

//why breadth first traverse

//

*let* retObj = {};

*const* nodes= [this.root];

while(nodes.length) {

*const* node = nodes.shift();

if(node.id == id) {

retObj = node;

retObj.keywords = this.\_getKeywords(node);

break;

} else {

nodes.push(...node.children)

}

}

return retObj;

}

GetCategoriesAtTheLevel(tree\_level) {

*let* node\_level = 0;

*const* retCategories = [];

*const* nodes = [this.root, 'LEVEL\_UP'];

while (nodes.length) {

*const* node = nodes.shift();

if (node === "LEVEL\_UP") {

if (node\_level == tree\_level) {

break;

} else {

node\_level++;

nodes.push(node);

}

} else {

if (node\_level == tree\_level) {

retCategories.push(node.id);

}

nodes.push(...node.children);

}

}

return retCategories;

}

// private function but still it will be exposed outside.

\_getKeywords(node) {

if (node.keywords && node.keywords.length > 0) {

return node.keywords;

} else {

return this.\_getKeywords(node.parent);

}

}

}

export {Categories};

**TESTS**

import {Category} from './category';

import { Categories } from "./categories";

describe('Categories', () *=>* {

*let* *root* = null;

*let* categories;

*let* secondLevelChildren;

beforeEach(() *=>* {

*root* = new Category({ id: -1 });

*const* cat1\_1 = new Category({ id: 100, name: "Business", keywords: "Money" });

*const* cat1\_2 = new Category({ id: 200, name: "Tutoring", keywords: "Teaching" });

*const* cat2\_1\_1 = new Category({ id: 101, name: "Accounting", keywords: "Taxes" });

*const* cat2\_1\_2 = new Category({ id: 102, name: "Taxation"});

*const* cat3\_1\_1 = new Category({ id: 103, name: "Corporate Tax" });

*const* cat3\_1\_2 = new Category({ id: 109, name: "Small Business Tax" });

*const* cat2\_2\_1 = new Category({ id: 201, name: "Computer" });

*const* cat3\_2\_1 = new Category({ id: 202, name: "Operating System" });

cat2\_1\_1.add(cat3\_1\_1);

cat2\_1\_1.add(cat3\_1\_2);

cat1\_1.add(cat2\_1\_1);

cat1\_1.add(cat2\_1\_2);

cat2\_2\_1.add(cat3\_2\_1);

cat1\_2.add(cat2\_2\_1);

*root*.add(cat1\_1);

*root*.add(cat1\_2);

categories = new Categories(*root*);

secondLevelChildren = categories.root.children;

});

test("should have a root category", () *=>* {

expect(categories.root).toEqual(*root*);

});

test("can add second level categories", () *=>* {

expect(secondLevelChildren.length).toEqual(2);

expect(secondLevelChildren[0].parent).toEqual(*root*);

});

test("can add third level categories", () *=>* {

*const* \_3\_1\_children = secondLevelChildren[0].children;

*const* \_3\_2\_children = secondLevelChildren[1].children;

expect(\_3\_1\_children.length).toEqual(2);

expect(\_3\_2\_children.length).toEqual(1);

});

test("can add fourth level categories", () *=>* {

*const* \_3\_1\_children = secondLevelChildren[0].children;

*const* \_3\_2\_children = secondLevelChildren[1].children;

*const* \_4\_1\_1\_children = \_3\_1\_children[0].children;

*const* \_4\_1\_2\_children = \_3\_1\_children[1].children;

*const* \_4\_2\_1\_children = \_3\_2\_children[0].children;

expect(\_4\_1\_1\_children.length).toEqual(2);

expect(\_4\_1\_2\_children).toEqual([]);

expect(\_4\_2\_1\_children.length).toEqual(1);

});

test('findByCategoryID should find existing category', () *=>* {

*const* category\_201 = categories.FindByCategoryId(201);

expect(category\_201.name).toEqual('Computer');

expect(category\_201.parentCategoryId).toEqual(200);

expect(category\_201.keywords).toEqual('Teaching');

*const* category\_103 = categories.FindByCategoryId(103);

expect(category\_103.name).toEqual("Corporate Tax");

expect(category\_103.parentCategoryId).toEqual(101);

expect(category\_103.keywords).toEqual("Taxes");

*const* category\_null = categories.FindByCategoryId(999);

expect(category\_null).toEqual({});

});

test("GetCategoriesAtTheLevel should return all the categories at that level", () *=>* {

*const* cats\_2 = categories.GetCategoriesAtTheLevel(2);

expect(cats\_2.length).toEqual(3);

expect(cats\_2).toEqual([101,102,201]);

*const* cats\_3 = categories.GetCategoriesAtTheLevel(3);

expect(cats\_3.length).toEqual(3);

expect(cats\_3).toEqual([103, 109, 202]);

*const* cats\_level\_not\_exist = categories.GetCategoriesAtTheLevel(6);

expect(cats\_level\_not\_exist.length).toEqual(0);

});

});