**School Catalogue**

Let’s put your knowledge of classes to the test by creating a digital school catalog for the New York City Department of Education. The Department of Education wants the catalog to hold quick reference material for each school in the city.

We need to create classes for primary, middle, and high schools. Because these classes share properties and methods, each will inherit from a parent School class. Our parent and three child classes have the following properties, getters, setters, and methods:

**School**

* **Properties**: name (string), level (one of three strings: 'primary', 'middle', or 'high'), and numberOfStudents (number)
* **Getters**: all properties have getters
* **Setters**: the numberOfStudents property has a setter
* **Methods**: .quickFacts() and .pickSubstituteTeacher() (this is a static method)

**Primary**

* Includes everything in the School class, plus one additional property
* **Properties**: pickupPolicy (string)

**Middle**

* Does not include any additional properties or methods

**High**

* Includes everything in the School class, plus one additional property
* **Properties**: sportsTeams (array of strings)

If you’re looking for a challenge, create the constructor() and getters for the four classes above. Then, use the setter and methods specifications in steps five, six, and seven to finish the project.

If you get stuck during this project or would like to see an experienced developer work through it, click “**Get Help**“ to see a **project walkthrough video**.

**Tasks**

**1/19Complete**

Mark the tasks as complete by checking them off

**Create ParentSchool Class**

**1.**

Let’s start by making a parent class for Primary, Middle, and High classes. Create an empty class named School.

Hint

Use the following syntax to create an empty class:

class emptyClass {

}

**2.**

Inside the School class, create an empty constructor() that accepts three parameters. The names of these parameters should match the properties listed in the narrative above.

Hint

The three parameters should include name, level, and numberOfStudents.

**3.**

Inside the constructor(), set the School properties. Prepend all of the properties with an underscore (\_).

Hint

Set the \_name property to name, set the \_level property to level, and set the \_numberOfStudents property to numberOfStudents. Use the following syntax:

class emptyClass {

constructor(propertyOne, propertyTwo, propertyThree) {

this.\_propertyOne = propertyOne;

this.\_propertyTwo = propertyTwo;

this.\_propertyThree = propertyThree;

}

}

**4.**

Create getters for the name, level, and numberOfStudents properties. Each getter should return the value saved to the property.

Hint

Use the following syntax to create getters:

class emptyClass {

constructor(propertyOne) {

...

}

get propertyOne() {

return this.\_propertyTwo;

}

}

**5.**

Create a setter for numberOfStudents. The method should first check if the input (newNumberOfStudents) is a number.

If it is a number, then set the numberOfStudents property. If not, log, 'Invalid input: numberOfStudents must be set to a Number.'

Stuck? Get a hint

**6.**

Under your getters, create a method named quickFacts that logs the following string to the console:

SCHOOL NAME educates NUMBER OF STUDENTS students at the LEVEL school level.

In place of SCHOOL NAME, substitute an instance’s name value. In place of NUMBER OF STUDENTS, substitute an instance’s numberOfStudents value. In place of LEVEL, substitute an instance’s level value.

Hint

Use the following syntax to define a method with string interpolation.

methodName() {

console.log(`You can interpolate a property like this: ${propertyOne}.`);

}

**7.**

Under .quickFacts(), create a static method named pickSubstituteTeacher. This method will receive one parameter, named substituteTeachers. The substituteTeachers parameter will take an array of strings.

Inside the method, randomly generate a whole number between 0 and one less than the length of the substituteTeachers array. Use this number to access and return the substitute teacher at that randomly generated index.

Hint

Use the static keyword to create a static method.

static methodName() {

...

}

Use the following to randomly generate a whole number between zero and one less than the length of an array:

Math.floor(myArray.length \* Math.random());

Access the value at an index of an array with the following syntax:

myArray[myIndex];

**Create PrimarySchool Class**

**8.**

Next, we’’ll build a PrimarySchool class that extends School. If you feel comfortable building the PrimarySchool class on your own, give it a shot. If not, use the steps below to help you along the way.

Whether you want to follow the steps or not, it’s important to note, the PrimarySchool class only has one additional property, pickupPolicy.

Create an empty PrimarySchool class that extends School.

Hint

Use the following syntax to extend a class:

class ChildClass extends ParentClass {

}

**9.**

Inside the PrimarySchool class, create a constructor() that accepts three arguments.

Hint

The constructor() must accept arguments for name, numberOfStudents, and pickupPolicy.

**10.**

Call super on the first line of the PrimarySchool‘s constructor(). Pass it any arguments that the parent constructor() uses.

Since this is the PrimarySchool class, pass 'primary' as the argument for the level parameter in the parent constructor.

Hint

Pass name, 'primary', and numberOfStudents to the parent constructor().

**11.**

Use the remaining argument to set the pickupPolicy property.

Hint

Use this.\_propertyName to set property values.

**12.**

Since our PrimarySchool class inherits Schools‘s properties and getters, we only need to create one new getter in the PrimarySchool class. Add this getter to the PrimarySchool class. Each getter should return the value saved to that property.

Hint

Use the following syntax to add a getter for the pickupPolicy property.

get myProperty() {

return this.\_myProperty;

}

**Create HighSchool Class**

**13.**

In this task, you will create a HighSchool class that extends the School class. In addition to the properties, getters, and methods in School, the HighSchool includes the following:

* **Properties**: sportsTeams (array of strings)
* **Getters**: A getter for the sportsTeams array. The getter should log all of the sports to the console.

Take a look at the Hint for step-by-step instructions.

Hint

Follow these steps to complete the task:

* Create an empty HighSchool class that extends School.
* Create a constructor() that accepts arguments for name, numberOfStudents, and sportsTeams.
* On the first line of the constructor(), call super and pass it name, 'high', and numberOfStudents as arugments.
* On the next line, set the sportsTeams property using this.\_
* Under the constructor(), create a getter for sportsTeams. Use console.log() to print the sportsTeams value to the console.

**Instances of PrimarySchool and HighSchool**

**14.**

Create a PrimarySchool instance with the following properties:

* Name: 'Lorraine Hansbury'
* Number of Students: 514
* Pickup Policy: 'Students must be picked up by a parent, guardian, or a family member over the age of 13.'

Save the instance to a constant variable named lorraineHansbury.

Hint

Use the following syntax to create a new instance:

const myInstance = new myClass(arg1, arg2, arg3);

**15.**

Call .quickFacts() on the lorraineHansbury instance.

**16.**

The principal of Lorraine Hansbury needs a substitute teacher for the day.

Call .pickSubstituteTeacher() on School, and pass the following array as an argument:

['Jamal Crawford', 'Lou Williams', 'J. R. Smith', 'James Harden', 'Jason Terry', 'Manu Ginobli'];

Hint

Use the following syntax to call a static method:

ClassName.staticMethod(arg);

**17.**

Create a HighSchool instance with the following properties:

* Name: 'Al E. Smith'
* Number of Students: 415
* Sports Teams: ['Baseball', 'Basketball', 'Volleyball', 'Track and Field']

Save the instance to a constant variable named alSmith.

Hint

Use the following syntax to create a new instance:

const myInstance = new myClass(arg1, arg2, arg3);

**18.**

Get the value saved to the sportsTeams property in alSmith.

**19.**

Nice work! If you’ve made it this far, you have a strong understanding of class syntax and inheritance.

If you would like to continue working on this project, we have listed some avenues to build on your progress.

* Add more properties to each class (averageTestScores, schoolOverview, etc.)
* Create a class called SchoolCatalog that holds a collection of schools. Create an instance of SchoolCatalog for primary, middle, and high schools.