# Travel Planner

**Objective:**
Build a simple travel planner system using JavaScript ES6 classes that demonstrates object-oriented programming concepts like encapsulation, inheritance, and polymorphism. Exercise 1: The Location superclass

1.1 Create a `Location` class with a constructor that takes three parameters: `name`, `description`, and `activities` with a default value of an empty array.

1.2 Inside the constructor, assign `name`, `description`, and `activities` to instance variables using the `this` keyword.

1.3 Add a method `addActivity` which accepts an `activity` parameter. This method should use the `push` method to add the activity to the `activities` array.

1.4 Add a `removeActivity` method which takes `activity` as a parameter. Use the `indexOf` method to find the index of the `activity` in the `activities` array. If the index is found (greater than -1), use the `splice` method to remove the `activity` from the `activities` array.

1.5 Implement a `displayLocation` method that returns a string containing the location's `name`, `description`, and `activities`, formatted as required.

Exercise 2: The Origin and Destination subclasses

2.1 Create an `Origin` subclass that extends `Location`. In the constructor, add an additional parameter `dateOfDeparture`. Call the `super` function with the parameters `name`, `description`, and `activities`. Assign `dateOfDeparture` to an instance variable.

2.2 Similarly, create a `Destination` subclass of `Location`. Add an additional parameter `dateOfArrival` to the constructor, call the `super` function appropriately, and assign `dateOfArrival` to an instance variable.

Exercise 3: The Transport superclass

3.1 Implement a `Transport` class with a constructor taking `type`, `duration`, and `cost` as parameters and assigning them to instance variables.

3.2 Add `changeDuration` and `changeCost` methods, each taking a parameter (`newDuration` or `newCost`) and updating the appropriate instance variable.

3.3 Create a `displayTransport` method that returns a formatted string containing the transport's `type`, `duration`, and `cost`.

Exercise 4: The Flight, Train, and Car subclasses

4.1 For each transport mode, create a subclass (`Flight`, `Train`, and `Car`) of `Transport`.

4.2 Each subclass constructor should take additional parameters specific to that transport mode (`flightNumber` for `Flight`, `trainNumber` for `Train`, `carMake` and `carModel` for `Car`). Call the `super` function appropriately and assign the additional parameters to instance variables.

Exercise 5: The Trip class

5.1 Create a `Trip` class with a constructor that initializes `origin` to null and `destinations` and `transports` to empty arrays.

5.2 Add a `setOrigin` method that takes `name`, `description`, and `dateOfDeparture`, creates a new `Origin` object, and assigns it to `this.origin`.

5.3 Implement an `addDestination` method that creates a new `Destination` object and adds it to the `destinations` array.

5.4 Write an `addTransport` method that takes `type`, `duration`, `cost`, and `additionalData` parameters. Use a switch statement to create the appropriate subclass of `Transport` based on the `type`, and add it to the `transports` array.

5.5 Finally, create a `displayTrip` method that returns a string containing all trip details by calling the appropriate display methods from the `origin`, `destinations`, and `transports` objects. Use a loop to iterate over the `destinations` and `transports` arrays.