Develop simple java and JS based program to show is-a, has-a, uses-a relationship

Java:

```
class Transport {
  String category;

  Transport(String category) {
    this.category = category;
  }

  void displayDetails() {
    System.out.println("Category: " + this.category);
  }
}

class Automobile extends Transport {
  String make;

  Automobile(String category, String make) {
    super(category);
    this.make = make;
  }

  void displayDetails() {
    super.displayDetails();
    System.out.println("Make: " + this.make);
  }
}

class ParkingLot {
  Transport[] vehicles;

  ParkingLot(Transport[] vehicles) {
    this.vehicles = vehicles;
  }

  void displayAllVehicles() {
    System.out.println("All Vehicles in Parking Lot:");
    for (Transport vehicle : vehicles) {
      vehicle.displayDetails();
    }
  }
}

class RepairTech {
  ParkingLot lot;

  RepairTech(ParkingLot lot) {
    this.lot = lot;
  }

  void repairTransport(String category) {
```

```
        System.out.println("Repairing vehicle of category: " + category);
        for (Transport vehicle : lot.vehicles) {
            if (vehicle.category.equals(category)) {
                System.out.println("Vehicle of category " + category + " repaired.");
                return;
            }
        }
        System.out.println("No vehicle of category " + category + " found in the parking lot.");
    }
}

public class Main {
    public static void main(String[] args) {
        Transport car1 = new Automobile("Car", "Toyota");
        Transport car2 = new Automobile("Car", "Honda");
        Transport bike1 = new Transport("Bike");
        Transport[] vehicles = {car1, car2, bike1};
        ParkingLot parkingLot = new ParkingLot(vehicles);
        RepairTech repairTech = new RepairTech(parkingLot);
        parkingLot.displayAllVehicles();
        repairTech.repairTransport("Car");
        repairTech.repairTransport("Truck");
    }
}
```

Java script

```
class Vehicle {
    constructor(type) {
        this.type = type;
    }
    displayInfo() {
        console.log("Type: " + this.type);
    }
}

class Car extends Vehicle {
    constructor(type, brand) {
        super(type);
        this.brand = brand;
    }
    displayInfo() {
        super.displayInfo();
```

```
        console.log("Brand: " + this.brand);
    }
}

class Garage {
    constructor(vehicles) {
        this.vehicles = vehicles;
    }
    displayAllVehicles() {
```

```
        console.log("All Vehicles in Garage:");
        this.vehicles.forEach(vehicle => {
            vehicle.displayInfo();
        });
    }
}

class Mechanic {
    constructor(garage) {
        this.garage = garage;
    }
    repairVehicle(type) {
        console.log("Repairing vehicle of type: " + type);
        for (let vehicle of this.garage.vehicles) {
            if (vehicle.type === type) {
                console.log("Vehicle of type " + type + " repaired.");
                return;
            }
        }
        console.log("No vehicle of type " + type + " found in the garage.");
    }
}

const main = () => {
    const car1 = new Car("Car", "Toyota");
    const car2 = new Car("Car", "Honda");
    const bike1 = new Vehicle("Bike");
    const vehicles = [car1, car2, bike1];
    const garage = new Garage(vehicles);
    const mechanic = new Mechanic(garage);
    garage.displayAllVehicles();
```

```
    mechanic.repairVehicle("Car");
    mechanic.repairVehicle("Truck");
};
main();
```