Write a Java program to create a class called "Vehicle" with attributes for make, model, and year. Create subclasses "Car" and "Truck" that add specific attributes like trunk size for cars and payload capacity for trucks. Implement a method to display vehicle details in each subclass.

Sample Solution:

Java Code:

Vehicle.java

```java
// Define the Vehicle class
public class Vehicle {

    // Attribute for make of the vehicle
    private String make;

    // Attribute for model of the vehicle
    private String model;

    // Attribute for year of the vehicle
    private int year;

    // Constructor to initialize Vehicle object
    public Vehicle(String make, String model, int year) {
        this.make = make;
        this.model = model;
        this.year = year;
    }

    // Method to display vehicle details
    public void displayDetails() {
        System.out.println("Vehicle Details:");
        System.out.println("Make: " + make);
        System.out.println("Model: " + model);
        System.out.println("Year: " + year);
    }

    // Getter method for make
    public String getMake() {
        return make;
    }
```

```
        // Getter method for model
        public String getModel() {
            return model;
        }

        // Getter method for year
        public int getYear() {
            return year;
        }
    }
```

The abov
the vehicl

- Attril
- Con:
- Meth
- Gett
  mod

This class s.

Car.java

```
    // Define the Car subclass that extends Vehicle
    class Car extends Vehicle {

        // Attribute for trunk size of the car
        private double trunkSize;

        // Constructor to initialize Car object
        public Car(String make, String model, int year, double trunkSize) {
            super(make, model, year); // Call the constructor of the superclass
            this.trunkSize = trunkSize;
        }

        // Override the displayDetails method to include trunk size
        @Override
        public void displayDetails() {
```

```java
            super.displayDetails(); // Call the superclass method
            System.out.println("Trunk Size: " + trunkSize + " cubic feet");
        }

        // Getter method for trunk size
        public double getTrunkSize() {
            return trunkSize;
        }

        // Setter method for trunk size
        public void setTrunkSize(double trunkSize) {
            if (trunkSize > 0) {
                this.trunkSize = trunkSize;
            } else {
                System.out.println("Trunk size must be positive.");
            }
        }
    }
```

This Java code defines a 'Car' subclass that extends the Vehicle class, adding an additional attribute and behavior specific to cars:

- Attribute: trunkSize to store the size of the car's trunk in cubic feet.
- Constructor: Initializes the 'Car' object with make, model, year, and trunk size. It calls the constructor of the Vehicle superclass to initialize the common attributes.
- Method to display details (displayDetails): Overrides the displayDetails method from the Vehicle class to include the trunk size. It first calls the superclass method to display common vehicle details, then adds the trunk size.
- Getter and Setter methods: getTrunkSize() returns the trunk size, and setTrunkSize() updates the trunk size if the value is positive.

This subclass enhances the Vehicle class by adding a specific feature for cars and overriding the method to display complete car details.

Truck.java

```java
    // Define the Truck subclass that extends Vehicle
    class Truck extends Vehicle {

        // Attribute for payload capacity of the truck
        private double payloadCapacity;
        // Constructor to initialize Truck object
        public Truck(String make, String model, int year, double payloadCapacit
            super(make, model, year); // Call the constructor of the superclass
            this.payloadCapacity = payloadCapacity;
        }

        // Override the displayDetails method to include payload capacity
        @Override
        public void displayDetails() {
            super.displayDetails(); // Call the superclass method
            System.out.println("Payload Capacity: " + payloadCapacity + " tons'
        }

        // Getter method for payload capacity
        public double getPayloadCapacity() {
            return payloadCapacity;
        }

        // Setter method for payload capacity
        public void setPayloadCapacity(double payloadCapacity) {
            if (payloadCapacity > 0) {
                this.payloadCapacity = payloadCapacity;
            } else {
                System.out.println("Payload capacity must be positive.");
            }
        }
    }
```

The above Java code defines a 'Truck' subclass that extends the Vehicle class, adding an attribute specific to trucks:

- Attribute: payloadCapacity to store the truck's payload capacity in tons.
- Constructor: Initializes the Truck object with make, model, year, and payload capacity. It calls the constructor of the Vehicle superclass to initialize the common attributes.
- Method to display details (displayDetails): Overrides the displayDetails method from the Vehicle class to include the payload capacity. It first calls the superclass method to display common vehicle details, then adds the payload capacity.
- Getter and Setter methods: getPayloadCapacity() returns the payload capacity, and setPayloadCapacity() updates the payload capacity if the value is positive.

This subclass extends the Vehicle class by adding a specific feature for trucks and overriding the method to display complete truck details.

Main.java

```java
// Main class to test the Vehicle, Car, and Truck classes
public class Main {
    public static void main(String[] args) {
        // Create a Car object
        Car car = new Car("Suzuki", "Swift", 2015, 15.1);
        car.displayDetails(); // Display car details

        // Create a Truck object
        Truck truck = new Truck("Ford", "F-150", 2016, 3.5);
        truck.displayDetails(); // Display truck details
    }
}
```

The above Java code defines a Main class with a main method used to test the Vehicle, Car, and Truck classes:

- Car Object Creation: An instance of the Car class is created with the make "Suzuki", model "Swift", year 2015, and trunk size 15.1 cubic feet. The displayDetails method is called to print the details of the car.
- Truck Object Creation: An instance of the Truck class is created with the make "Ford", model "F-150", year 2016, and payload capacity 3.5 tons. The displayDetails method is called to print the details of the truck.

This code tests the functionality of the Vehicle, Car, and Truck classes by creating instances of each and displaying their details.

Sample Output:

```
Vehicle Details:
Make: Suzuki
Model: Swift
Year: 2015
Trunk Size: 15.1 cubic feet
Vehicle Details:
Make: Ford
Model: F-150
Year: 2016
Payload Capacity: 3.5 tons
```

Java Code Editor: