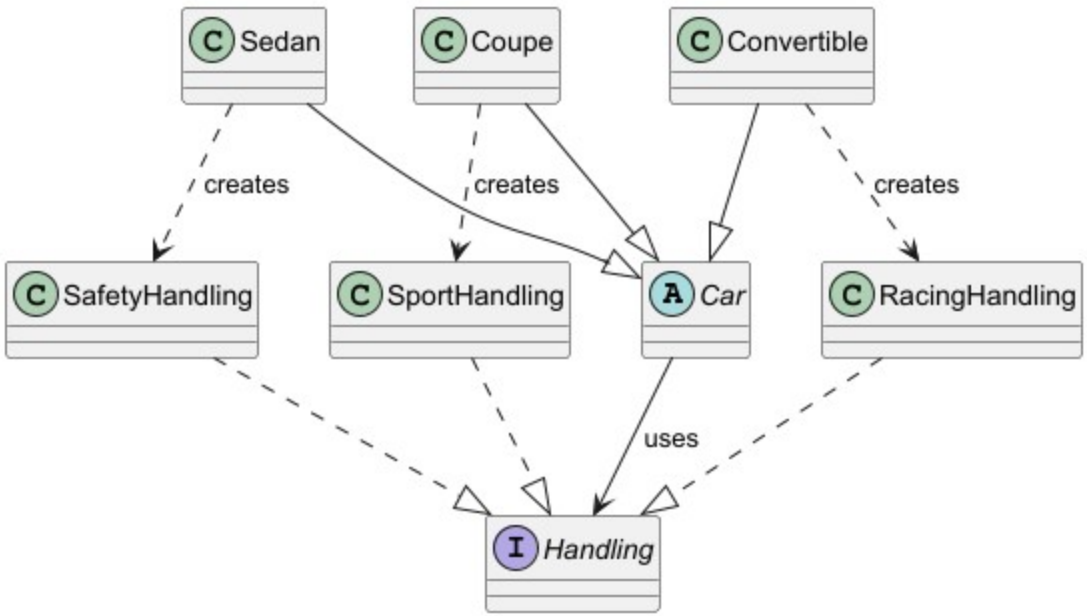
```
/******* Main.java ******/

package org.coursera.lab.strategy;

import java.util.ArrayList;

/*
    Starting code with Car class for Strategy exercise
    You should refactor to use Strategy for handling
    Keep automatic car object naming as is
    Bruce Montgomery 10/12/24
*/

public class Main {
    public static void main(String[] args) {
        // Create a list of the different car objects
        ArrayList<Car> cars = new ArrayList<Car>();
        cars.add(new Sedan());
        cars.add(new Coupe());
        cars.add(new Convertible());
        // call the handle method for all of them
        for (Car c : cars) {
            System.out.print(c.name + " ");
            c.handle();
        }
    }
}

abstract class Car {
    String name;
    int cost;
    Handling handlingStrategy;
    protected static int carCounter = 0;

    Car(int cost, Handling handlingStrategy) {
        carCounter++;
        name = getType() + " " + carCounter;
        this.cost = cost;
        this.handlingStrategy = handlingStrategy;
    }

    abstract String getType();

    public void handle() {
        System.out.println(handlingStrategy.handle());
    }
}

/******* Sedan.java ******/
package org.coursera.lab.strategy;
```

```
public class Sedan extends Car {

    public Sedan() {
        super(10000, new SafetyHandling());
    }

    @Override
    String getType() {
        return "sedan";
    }
}


/******* Coupe.java ******/
package org.coursera.lab.strategy;

public class Coupe extends Car {

    public Coupe() {
        super(15000, new SportHandling());
    }

    @Override
    String getType() {
        return "coupe";
    }
}

/******* Convertible.java ******/
package org.coursera.lab.strategy;

public class Convertible extends Car {

    public Convertible() {
        super(20000, new RacingHandling());
    }

    @Override
    String getType() {
        return "convertible";
    }
}

/******* Handling.java ******/
package org.coursera.lab.strategy;

public interface Handling {
    default String handle() {
        return "has undefined handling";
    }
}
```

```
/******* RacingHandling.java ******/
package org.coursera.lab.strategy;

public class RacingHandling implements Handling {
    @Override
    public String handle() {
        return "skids through a turn";
    }
}

/******* SafetyHandling.java ******/
package org.coursera.lab.strategy;

public class SafetyHandling implements Handling {
    @Override
    public String handle() {
        return "eases through turn";
    }
}

/******* SportHandling.java ******/
package org.coursera.lab.strategy;

public class SportHandling implements Handling {
    @Override
    public String handle() {
        return "makes a tight turn";
    }
}

/******* MainTest.java ******/
package org.coursera.lab.strategy;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

public class MainTest {

    @Test
    public void strategyTEst() {
        assertEquals("eases through turn", new Sedan().handlingStrategy.handle());
        assertEquals("makes a tight turn", new Coupe().handlingStrategy.handle());
        assertEquals("skids through a turn", new
Convertible().handlingStrategy.handle());
    }

}
```


Sedan
Coupe
Convertible
creates
creates
creates
SafetyHandling
SportHandling
Car
RacingHandling
uses
Handling