```
================================================================================
FILE: src\main\java\org\coursera\lab\strategy\AbstractCar.java
================================================================================

package org.coursera.lab.strategy;

import org.coursera.lab.strategy.handling.Handling;

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

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

    public String getName() {
        return name;
    }

    public int getCost() {
        return cost;
    }

    public Handling getHandlingStrategy() {
        return handlingStrategy;
    }

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

    // Base cars have no decorators
    @Override
    public int getUndercoatCount() {
        return 0;
    }

    @Override
    public int getSeatCoverCount() {
        return 0;
    }

    @Override
    public int getServiceCount() {
```

```
        return 0;
    }
}
```

```
================================================================================
FILE: src\main\java\org\coursera\lab\strategy\Car.java
================================================================================

package org.coursera.lab.strategy;

import org.coursera.lab.strategy.handling.Handling;

public interface Car {

    String getName();
    String getType();
    void handle();
    int getCost();

    // Methods to track decorator counts
    int getUndercoatCount();
    int getSeatCoverCount();
    int getServiceCount();

}
```

```
================================================================================
FILE: src\main\java\org\coursera\lab\strategy\Convertible.java
================================================================================

package org.coursera.lab.strategy;

import org.coursera.lab.strategy.handling.RacingHandling;

public class Convertible extends AbstractCar {

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

    @Override
    public String getType() {
        return "Convertible";
    }
}
```

```
================================================================================
FILE: src\main\java\org\coursera\lab\strategy\Coupe.java
================================================================================

package org.coursera.lab.strategy;

import org.coursera.lab.strategy.handling.SportHandling;

public class Coupe extends AbstractCar {

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

    @Override
    public String getType() {
        return "Coupe";
    }
}



================================================================================
FILE: src\main\java\org\coursera\lab\strategy\Main.java
================================================================================

package org.coursera.lab.strategy;

import org.coursera.lab.strategy.decorator.SeatCoverDecorator;
import org.coursera.lab.strategy.decorator.ServiceDecorator;
import org.coursera.lab.strategy.decorator.UndercoatDecorator;

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 UndercoatDecorator(new ServiceDecorator(new Sedan())));  //
Decorations
        cars.add(new UndercoatDecorator(new Coupe())); // Decorations
```

```
        cars.add(new SeatCoverDecorator(new SeatCoverDecorator(new
Convertible()))); // Decorations
        // call the handle method for all of them
        for (Car c : cars) {
            System.out.print(c.getName() + " ");
            c.handle();
        }
    }
}
```

```
================================================================================
FILE: src\main\java\org\coursera\lab\strategy\Sedan.java
================================================================================

package org.coursera.lab.strategy;

import org.coursera.lab.strategy.handling.SafetyHandling;

public class Sedan extends AbstractCar {

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

    @Override
    public String getType() {
        return "Sedan";
    }
}
```

```
================================================================================
FILE: src\main\java\org\coursera\lab\strategy\decorator\SeatCoverDecorator.java
================================================================================

package org.coursera.lab.strategy.decorator;

import org.coursera.lab.strategy.Car;

public class SeatCoverDecorator implements Car {

    private final Car car;

    public SeatCoverDecorator(Car car) {
        if (car == null) {
            throw new IllegalArgumentException("Car cannot be null");
        } else if (car.getSeatCoverCount() >= 4) {
```

```
            throw new IllegalStateException("Car can only have up to 4 seat
covers");
        }
        this.car = car;
    }

    @Override
    public String getName() {
        return car.getName() + " (add seat cover)";
    }

    @Override
    public String getType() {
        return car.getType();
    }

    @Override
    public int getCost() {
        return car.getCost() + 250;
    } // Add cost for seat cover

    @Override
    public void handle() {
        car.handle();
    }

    @Override
    public int getUndercoatCount() {
        return car.getUndercoatCount();
    }

    @Override
    public int getSeatCoverCount() {
        return car.getSeatCoverCount() + 1;
    }

    @Override
    public int getServiceCount() {
        return car.getServiceCount();
    }

}
```

```
================================================================================
FILE: src\main\java\org\coursera\lab\strategy\decorator\ServiceDecorator.java
================================================================================

package org.coursera.lab.strategy.decorator;
```

```
import org.coursera.lab.strategy.Car;

public class ServiceDecorator implements Car {

    private final Car car;

    public ServiceDecorator(Car car) {
        if (car == null) {
            throw new IllegalArgumentException("Car cannot be null");
        } else if (car.getServiceCount() >= 2) {
            throw new IllegalStateException("Car can only have up to 2 services");
        }
        this.car = car;
    }

    @Override
    public String getName() {
        return car.getName() + " (add service visits)";
    }

    @Override
    public String getType() {
        return car.getType();
    }

    @Override
    public int getCost() {
        return car.getCost() + 400;
    }  // Add cost for service visits

    @Override
    public void handle() {
        car.handle();
    }

    @Override
    public int getUndercoatCount() {
        return car.getUndercoatCount();
    }

    @Override
    public int getSeatCoverCount() {
        return car.getSeatCoverCount();
    }

    @Override
    public int getServiceCount() {
        return car.getServiceCount() + 1;
    }

}
```

```
================================================================================
FILE: src\main\java\org\coursera\lab\strategy\decorator\UndercoatDecorator.java
================================================================================

package org.coursera.lab.strategy.decorator;

import org.coursera.lab.strategy.Car;

public class UndercoatDecorator implements Car {

    private final Car car;

    public UndercoatDecorator(Car car) {
        if (car == null) {
            throw new IllegalArgumentException("Car cannot be null");
        } else if (car.getUndercoatCount() >= 1) {
            throw new IllegalStateException("Car already has an undercoat");
        }
        this.car = car;
    }

    @Override
    public String getName() {
        return car.getName() + " (add undercoat)";
    }

    @Override
    public String getType() {
        return car.getType();
    }

    @Override
    public int getCost() {
        return car.getCost() + 500; // Add cost for undercoat
    }

    @Override
    public void handle() {
        car.handle();
    }

    @Override
    public int getUndercoatCount() {
        return car.getUndercoatCount() + 1;
    }

    @Override
    public int getSeatCoverCount() {
```

```
        return car.getSeatCoverCount();
    }

    @Override
    public int getServiceCount() {
        return car.getServiceCount();
    }

}
```

```
================================================================================
FILE: src\main\java\org\coursera\lab\strategy\handling\Handling.java
================================================================================

package org.coursera.lab.strategy.handling;

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

```
================================================================================
FILE: src\main\java\org\coursera\lab\strategy\handling\RacingHandling.java
================================================================================

package org.coursera.lab.strategy.handling;

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

```
================================================================================
FILE: src\main\java\org\coursera\lab\strategy\handling\SafetyHandling.java
================================================================================

package org.coursera.lab.strategy.handling;

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

```
    }
}
```

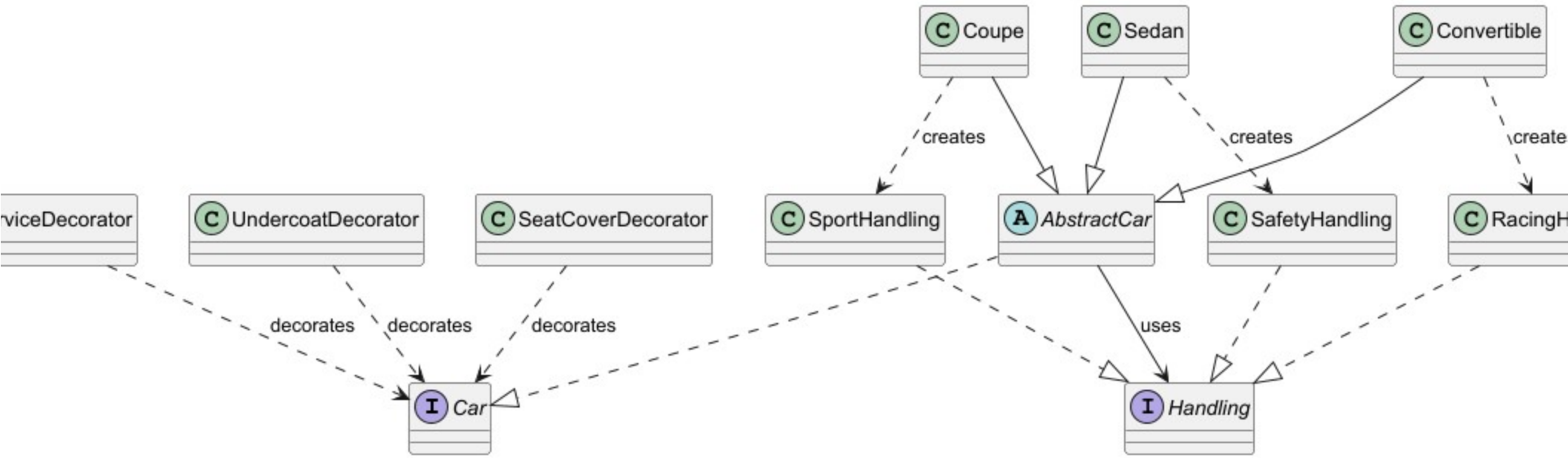
```
================================================================================
FILE: src\main\java\org\coursera\lab\strategy\handling\SportHandling.java
================================================================================

package org.coursera.lab.strategy.handling;

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

```
================================================================================
FILE: src\test\java\org\coursera\lab\strategy\MainTest.java
================================================================================

package org.coursera.lab.strategy;

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

import org.coursera.lab.strategy.decorator.SeatCoverDecorator;
import org.coursera.lab.strategy.decorator.ServiceDecorator;
import org.coursera.lab.strategy.decorator.UndercoatDecorator;
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());
    }

    @Test
    public void testDecoratorConvertible() {
        Car cv2 = new Convertible();
        cv2 = new UndercoatDecorator(cv2);
        cv2 = new SeatCoverDecorator(cv2);
        cv2 = new SeatCoverDecorator(cv2);
        cv2 = new ServiceDecorator(cv2);
        cv2 = new ServiceDecorator(cv2);
```

```
        assertEquals("Convertible 1 (add undercoat) (add seat cover) (add seat
cover) (add service visits) (add service visits)", cv2.getName());
        assertEquals(21800, cv2.getCost());
    }

    @Test
    public void testDecoratorLimit() {

        assertThrows(IllegalStateException.class, () -> {
            Car cv2 = new Convertible();
            cv2 = new UndercoatDecorator(cv2);
            cv2 = new UndercoatDecorator(cv2);
        });

        assertThrows(IllegalStateException.class, () -> {
            Car cv2 = new Convertible();
            cv2 = new SeatCoverDecorator(cv2);
            cv2 = new SeatCoverDecorator(cv2);
            cv2 = new SeatCoverDecorator(cv2);
            cv2 = new SeatCoverDecorator(cv2);
            cv2 = new SeatCoverDecorator(cv2);
        });

        assertThrows(IllegalStateException.class, () -> {
            Car cv2 = new Convertible();
            cv2 = new ServiceDecorator(cv2);
            cv2 = new ServiceDecorator(cv2);
            cv2 = new ServiceDecorator(cv2);
        });
    }

}
```

Main output:

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program
Files\JetBrains\IntelliJ IDEA Community Edition 2025.1.1.1\lib\idea_rt.jar=64750"
-Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -
classpath C:\Users\ck\Documents\dev\source\CSCA\csca-
java\5438\w1\learn\target\classes org.coursera.lab.strategy.Main
Sedan 1 (add service visits) (add undercoat) eases through turn
Coupe 2 (add undercoat) makes a tight turn
Convertible 3 (add seat cover) (add seat cover) skids through a turn

Process finished with exit code 0
```


Coupe
Sedan
Convertible
creates
creates
create
viceDecorator
UndercoatDecorator
SeatCoverDecorator
SportHandling
AbstractCar
SafetyHandling
RacingH
decorates
decorates
decorates
uses
Car
Handling