**Car.java**

package factorypattern;

public abstract class Car {

public Car(CarType model) {

this.model = model;

arrangeParts();

}

private void arrangeParts() {

// Do one time processing here

}

// Do subclass level processing in this method

protected abstract void construct();

private CarType model = null;

public CarType getModel() {

return model;

}

public void setModel(CarType model) {

this.model = model;

}

}

**CarFactory.java**

package factorypattern;

public class CarFactory {

public static Car buildCar(CarType model) {

Car car = null;

switch (model) {

case SMALL:

car = new SmallCar();

break;

case SEDAN:

car = new SedanCar();

break;

case LUXURY:

car = new LuxuryCar();

break;

default:

// throw some exception

break;

}

return car;

}

}

CarType .java

package factorypattern;

public enum CarType {

SMALL, SEDAN, LUXURY

}

LuxuryCar.java

package factorypattern;

public class LuxuryCar extends Car {

LuxuryCar() {

super(CarType.LUXURY);

construct();

}

@Override

protected void construct() {

System.out.println("Building luxury car");

// add accessories

}

}

SedanCar.java

package factorypattern;

public class SedanCar extends Car {

SedanCar() {

super(CarType.SEDAN);

construct();

}

@Override

protected void construct() {

System.out.println("Building sedan car");

// add accessories

}

}

SmallCar.java

package factorypattern;

public class SmallCar extends Car {

SmallCar() {

super(CarType.SMALL);

construct();

}

@Override

protected void construct() {

System.out.println("Building small car");

// add accessories

}

}

TestFactorypatern.java

package factorypattern;

public class TestFactoryPattern {

public static void main(String[] args) {

System.out.println(CarFactory.buildCar(CarType.SMALL));

System.out.println(CarFactory.buildCar(CarType.SEDAN));

System.out.println(CarFactory.buildCar(CarType.LUXURY));

}

}