Class SienaSedan

@Override métodos da interface CarroSedan

@Override

**public** **void** exibirInfoSedan() {

System.***out***.println("Modelo: Siena\nFábrica: Fiat\nCategoria:Sedan");

}

Class FabricaFiat

@Override métodos da interface

FabricaDeCarro

@Override

**public** CarroSedan criarCarroSedan() {

**return** **new** SienaSedan();

}

@Override

**public** CarroPopular criarCarroPopular() {

**return** **new** SienaPopular();

}

Class FabricaFord

@Override métodos da interface

FabricaDeCarro

<<FabricaDeCarro>>

+ CarroSedan criarCarroSedan();

+CarroPopular criarCarroPopular();

<<CarroSedan>>

+ void exibirInfoSedan();

<<CarroPopular>>

+ void exibirInfoPopular();

Class SienaPopular

@Override métodos da interface CarroPopular

@Override

**public** **void** exibirInfoPopular() {

System.***out***.println("Modelo: Siena\nFábrica: Fiat\nCategoria:Popular");

}

Tema: Abstract Factory

Origem: <https://github.com/MarcosX/Padr-es-de-Projeto>

Data da pesquisa: 23-11-2018

**Quando tem tipo mais específico CarroSedan e subtipo Siena. Por isso, que tem que haver o tipo CarroPopular**

**public** **class** Cliente {

**public** **static** **void** main(String[] args) {

FabricaDeCarro fabrica = **new** FabricaFiat(); //fiat

CarroSedan sedan = fabrica.criarCarroSedan(); //carro sena Sedan

CarroPopular popular = fabrica.criarCarroPopular();

sedan.exibirInfoSedan();

System.***out***.println();

popular.exibirInfoPopular();

System.***out***.println();

fabrica = **new** FabricaFord();

sedan = fabrica.criarCarroSedan();

popular = fabrica.criarCarroPopular();

sedan.exibirInfoSedan();

System.***out***.println();

popular.exibirInfoPopular();

}

}

Class Siena

@Override métodos da interface Carro

@Override

**public** **void** exibirInfo() {

System.***out***.println("Modelo: Siena\nFábrica: Fiat");

}

Class FabricaFord

@Override métodos da interface

FabricaDeCarro

Class FabricaFiat

@Override métodos da interface

FabricaDeCarro

@Override

**public** Carro criarCarro() {

**return** **new** Siena();

}

}

<<FabricaDeCarro>>

+ **Carro criarCarro();**

<<Carro>>

+ void exibirInfo ();

Class Fiesta

@Override métodos da interface Carro

@Override

**public** **void** exibirInfo() {

System.***out***.println("Modelo: Fiesta\nFabricante: Ford");

}

Tema: Factory Method

Origem: <https://github.com/MarcosX/Padr-es-de-Projeto>

Data da pesquisa: 23-11-2018

**Quando tem tipo genérico Carro e subtipo Siena –**

**observação:** Caso a fabrica Fiat tenha mais de um modelo no sistema, terá que ser construído método para esse novo modelo no sistema. Neste , caso é melhor receber como parâmetro uma String, que informa o modelo e o método ser direcionado para a espessividade. Exemplo:

@Override //Na interface FabricaDeCarro deverá ter a devida alteração

**public** Carro criarCarro(String modelo) {

if (modelo.equals(“Siena”)){

**return** **new** Siena();

} }

|  |
| --- |
| public class Cliente { |
|  | public static void main(String[] args) { |
|  | FabricaDeCarro fabrica = new FabricaFiat(); |
|  | Carro carro = fabrica.criarCarro(); |
|  | carro.exibirInfo(); |
|  | System.out.println(); |
|  |  |
|  | fabrica = new FabricaWolks(); |
|  | carro = fabrica.criarCarro(); |
|  | carro.exibirInfo(); |
|  | System.out.println(); |
|  |  |
|  | fabrica = new FabricaFord(); |
|  | carro = fabrica.criarCarro(); |
|  | carro.exibirInfo(); |
|  | System.out.println(); |
|  |  |
|  | fabrica = new FabricaChevrolet(); |
|  | carro = fabrica.criarCarro(); |
|  | carro.exibirInfo(); |
|  | } |
|  | } |