Vitor Manoel Vidal Braz

Exercício – Fábrica

Governador Valadares

2025

Documento Oficial da Aplicação Java – Sistema Fábrica VitorCar

1. Objetivo da Aplicação

O objetivo desta aplicação é desenvolver um sistema completo de cadastro e gerenciamento de **Pessoas** e **Veículos** em uma fábrica automotiva, utilizando os fundamentos de **Programação Orientada a Objetos (POO)** na linguagem Java. O projeto busca consolidar conhecimentos como encapsulamento, abstração, uso de métodos acessores (get/set), relacionamento entre objetos e aplicação prática de validações e simulações administrativas.

---

2. Estrutura da Aplicação

Classes principais:

- `Fabrica` – Classe principal que contém o método `main` e toda a lógica do sistema.
- `Pessoa` – Representa o cidadão, com atributos pessoais e uma lista de veículos associados.
- `Carro` – Representa o veículo, com características como modelo, placa e ano de fabricação.

Cada classe é modularizada e aplica boas práticas de encapsulamento e reutilização de código.

---

3. Requisitos Atendidos

Requisito	Descrição	Implementação
Cadastro de Pessoa e Carro	Permite cadastrar usuários e seus respectivos veículos pela interface principal.	Implementado com validações robustas e associação entre objetos.
Proteção dos Atributos	Os atributos das classes estão `private` para garantir encapsulamento.	Implementado via modificadores de acesso e métodos acessores.
Uso de Getters e Setters	Todos os atributos são acessados e modificados por métodos `get` e `set`.	Aplicado em todas as classes auxiliares.
Exemplo de Erro de Acesso	A tentativa de acessar diretamente um atributo `private` é mostrada como exemplo comentado no `main`.	Presente no início do método `main` como código comentado.

---

4. Visão Geral das Classes

Classe Pessoa.java

- **Atributos:** `nome`, `idade`, `cpf`, `carros`
- **Responsabilidade:** Representa uma pessoa com CPF único e lista de veículos cadastrados.
- **Métodos:** Construtor, `getNome`, `setNome`, `getIdade`, `setIdade`, `getCpf`, `setCpf`, `adicionarCarro`, `getCarros`

Classe Carro.java

- **Atributos:** `modelo`, `placa`, `anoFabricacao`
- **Responsabilidade:** Representa um carro com seus dados principais, incluindo um atributo numérico obrigatório.
- **Métodos:** Construtor, `get` e `set` para todos os atributos

Classe Fabrica.java (Main)

- **Responsabilidade:** Executa a aplicação principal, apresentando menus de navegação, cadastro, relatórios e administração. Coordena a interação entre as classes.

---

5. Funcionalidades Detalhadas do Sistema

Menu Principal

O sistema apresenta ao usuário um menu com as seguintes opções:

1. **Cadastrar Pessoa** – Solicita nome, idade e CPF. O CPF é validado com 11 dígitos e formatado com máscara `###.###.###-##`. Impede duplicidade e valida o nome apenas com letras.
2. **Cadastrar Carro** – Solicita CPF do proprietário. Verifica se o CPF está cadastrado. Para menores de idade, exige autorização de um responsável legal. A placa do carro deve seguir os formatos `ABC1234` ou `BRA1A23` e ser única. O ano do carro deve estar entre 1886 e o ano atual.
3. **Visualizar Dados** – Exibe um relatório completo com todas as pessoas, seus carros e, se for menor de idade, o responsável legal. Inclui histórico de manutenção.
4. **Área Administrativa** – Acesso restrito. Requer usuário "admin" e senha "1234". Permite:
    - Visualizar estatísticas do sistema
    - Simular manutenções (troca de óleo, pneu, revisão)
    - Listar todos os carros e pessoas
    - Remover registros por CPF
    - Verificar menores de idade cadastrados
5. **Créditos** – Apresenta dados do autor, curso e repositório.
6. **Sobre o Sistema** – Exibe informações técnicas do funcionamento, arquitetura e recursos usados no projeto.
7. **Sair** – Encerra o sistema com mensagem personalizada.– Encerra o sistema com mensagem personalizada.

Máscara do CPF

Ao cadastrar uma pessoa, o CPF digitado com 11 dígitos numéricos é automaticamente formatado para o padrão `000.000.000-00`, assegurando a padronização visual em todo o sistema.

Restrições para Menores de Idade

Caso a pessoa cadastrada tenha menos de 18 anos, ela **não poderá registrar um carro** até que um **termo de autorização seja preenchido** com:

- Nome e CPF do responsável legal
- Data de autorização válida (não futura)
- Confirmação do termo com exibição formatada no console

Após a autorização, o menor é liberado permanentemente para registrar veículos no sistema.

---

6. Exemplo de Erro de Acesso

Demonstração de erro de acesso direto a atributo privado, conforme exigido:

```
// Pessoa exemplo = new Pessoa("Teste", 20, "00000000000");
// exemplo.nome = "Outro nome"; // Erro proposital: 'nome' tem acesso privado
```

---

7. Funcionalidades Adicionais

Validação de Dados

- CPF único e formatado
- Nome sem números
- Idade entre 1 e 120
- Placa válida e única
- Ano de fabricação coerente

Controle de Menores

- Cadastro condicionado ao termo de responsabilidade
- Responsável autorizado uma única vez por menor

Interface e Estilo

- Limpeza de tela entre menus
- Mensagens informativas e acessíveis
- Estrutura de menu clara e funcional

Administração Protegida

- Login com usuário e senha
- Submenu exclusivo para gerenciar dados do sistema

Manutenção Veicular

- Registro de serviços por carro
- Histórico salvo por placa com data e tipo

Estatísticas do Sistema

- Número de pessoas e carros cadastrados
- Média de idade
- Carros mais antigos

---

8. Créditos

O sistema **Fábrica VitorCar** foi desenvolvido por Vitor Manoel sob a identidade criativa **VM SYSTEMS**, idealizada por Vitor Manoel Vidal Braz para representar seus projetos acadêmicos e criativos em Java. A VM SYSTEMS tem como missão padronizar o estilo visual dos sistemas desenvolvidos em terminal, oferecendo identidade, organização e uma experiência imersiva mesmo em ambientes sem interface gráfica.

O projeto foi desenvolvido em linguagem Java, utilizando os princípios da Programação Orientada a Objetos (POO), como parte de uma atividade prática da disciplina de POO no curso de Sistemas de Informação da Univale.foi desenvolvido em linguagem Java, utilizando os princípios da Programação Orientada a Objetos (POO), como parte de uma atividade prática da disciplina de POO no curso de Sistemas de Informação da Univale.

Informações do projeto:

- Nome do sistema: Fábrica VitorCar
- Repositório GitHub: https://github.com/vitormanoelvb/sistema-fabrica-vitorcar
- Desenvolvedor: Vitor Manoel Vidal Braz
- Orientador: Prof. Vitor Silva Ribeiro
- Universidade: Univale – Universidade Vale do Rio Doce
- Curso: Sistemas de Informação – 3º período
- Disciplina: Programação Orientada a Objetos
- Linguagem utilizada: Java
- VM ENGINE DEVELOPMENT: versão atual 1.5

A interface de console foi construída com um estilo próprio e padronizado, otimizando a experiência visual do usuário mesmo em ambiente de terminal.

Sobre o Motor VM ENGINE DEVELOPMENT

O **VM ENGINE DEVELOPMENT** é um motor gráfico de console desenvolvido exclusivamente por **Vitor Manoel Vidal Braz**.

Este motor surgiu inicialmente como **VM ENGINE**, com foco em criar **elementos visuais para vídeos do YouTube**, como introduções animadas, efeitos de texto e banners interativos. Com o tempo, sua estrutura gráfica baseada em blocos e transições foi adaptada para o terminal Java, ganhando robustez e novas funções voltadas ao desenvolvimento de software.

Com essa transição, o projeto foi reestruturado e passou a se chamar **VM ENGINE DEVELOPMENT**, sendo hoje utilizado em aplicações interativas de console, como o Sistema Fábrica VitorCar, já na sua **versão 1.5**.

Inicialmente projetado para sistemas de terminal, o motor surgiu durante a construção de sistemas acadêmicos para fins de apresentação, simulando uma interface visual elegante e organizada mesmo em modo texto.

Características principais:

- Tela de abertura com identidade visual centralizada
- Transições suaves entre menus via `Thread.sleep`
- Sistema de limpeza de tela por método padrão
- Layout simétrico de caixas e molduras com uso de caracteres ASCII
- Identidade visual padronizada para múltiplos projetos

Histórico de versões:

- **Versão 1.0**: Utilizada no Sistema de Gerenciamento de Biblioteca com árvore ABB (abril de 2025)
- **Versão 1.5**: Utilizada no Sistema Fábrica VitorCar com controle de menor, placa e CPF, e funcionalidades administrativas completas

O motor é constantemente aprimorado e adaptado a novos projetos da equipe, sempre com foco em clareza visual, praticidade de navegação e estilo próprio. Ele representa um diferencial de identidade. O **VM ENGINE DEVELOPMENT VERSÃO 1.5** foi otimizada para linguagens mais antigas de programação, assegurando o funcionamento adequado do Java no projeto acadêmico de **Vitor Manoel** – *Sistemas de Informação: Fábrica VitorCar*.

---

9. Bônus: Integração com Banco de Dados (MySQL)

Como aprimoramento adicional, o sistema Fábrica VitorCar também possui integração com banco de dados MySQL, permitindo que os dados de pessoas, veículos e autorizações sejam armazenados de forma persistente.

Estrutura do banco:

- Banco de dados: `fabrica_vitorcar`
- Tabelas principais:
  - `pessoa`: contém nome, idade e CPF
  - `carro`: contém modelo, placa, ano e referência à pessoa
  - `termo_autorizacao`: registra termos legais para menores
- **VIEW**: `vw_menores_autorizados` – exibe menores com dados do responsável
- **FUNCTION**: `tem_termo_autorizacao()` – retorna se CPF tem termo assinado
- **Procedures**:
  - `sp_cadastrar_pessoa()` – insere pessoa com verificação de CPF
  - `sp_cadastrar_carro()` – insere carro vinculado à pessoa

Essa integração é feita por meio das classes Java `PessoaDAO` e `ConexaoBD`, que utilizam JDBC para conectar e interagir com o banco.

10. Conclusão

A aplicação **Sistema Fábrica VitorCar** apresenta uma estrutura sólida, completa e extensível, construída com boas práticas de POO. Atende todos os critérios definidos na tarefa, oferecendo ainda:

- Interface intuitiva no console
- Simulação realista de processos administrativos
- Máscara de CPF e restrições legais integradas
- Mecanismos de segurança e validação eficientes

Este projeto demonstra domínio técnico de Java, criatividade na resolução de problemas e cuidado na experiência do usuário. Como sugestão futura, pode-se adaptar este sistema para uma aplicação gráfica (JavaFX) ou web (com Spring Boot) para ampliar sua usabilidade.