Engenharia de Software 2
Aula 07 - Modelagem com base na UML

"In theory, theory and practice are the same. In practice, they are not."
— Albert Einstein

1 Preparação

Na atividade de hoje os grupos irão trabalhar sobre a implementação do sistema modelado na atividade anterior.

Foi modelado um Sistema de Locação de Veículos [1], levando-se em consideração os seguintes requisitos:

1. A empresa tem uma grande frota de carros de passeio, sendo que esses carros apresentam diferentes marcas e modelos. Eventualmente um carro pode ser retirado da frota devido a algum acidente grave ou simplesmente por ter sido considerado velho demais para o padrão da empresa e tenha sido vendido. Da mesma forma, a empresa eventualmente renova a frota, sendo necessário, portanto, estar sempre mantendo o cadastro de veículos da empresa.
2. Os clientes dirigem-se à empresa e solicitam o aluguel de veículos, No entanto, primeiramente é necessário cadastrá-los, caso ainda não possuam cadastro ou atualizar o cadastro caso seus dados tenham sido alterados.
3. Depois de ter se identificado e, se necessário, cadastrado o cliente escolherá o carro que deseja alugar. O valor do aluguel depende da categoria do carro, do número de diárias utilizadas e da quilometragem percorrida. Existem 5 categorias de veículos:
   - Categoria A - Hatch Compacto.
   - Categoria B - Sedan Intermediário.
   - Categoria C - SUV Compacto.
   - Categoria D - Sedan Executivo.
   - Categoria E - SUV Executivo.
4. Durante o processo de locação o cliente deve informar por quanto tempo pretende utilizar o carro. Deve-se registrar o veículo como locado, registrar a data e a hora da retirada e a quilometragem que o carro se encontra no momento da retirada.
5. Quando o cliente devolve o carro deve-se definir o veículo como devolvido, registrar a data e a hora da devolução e a quilometragem em que se encontra.

1.1 Diagrama de Classes Atualizado

A Figura 1 apresenta o diagrama de classes atualizado para o Sistema de Locação de Veículo. Esse diagrama já utiliza normas atualizadas para nome de métodos e atributos.

[1] Adaptado de um exemplo publicado no livro UML2 - Uma Abordagem Prática de G. Guedes [1].

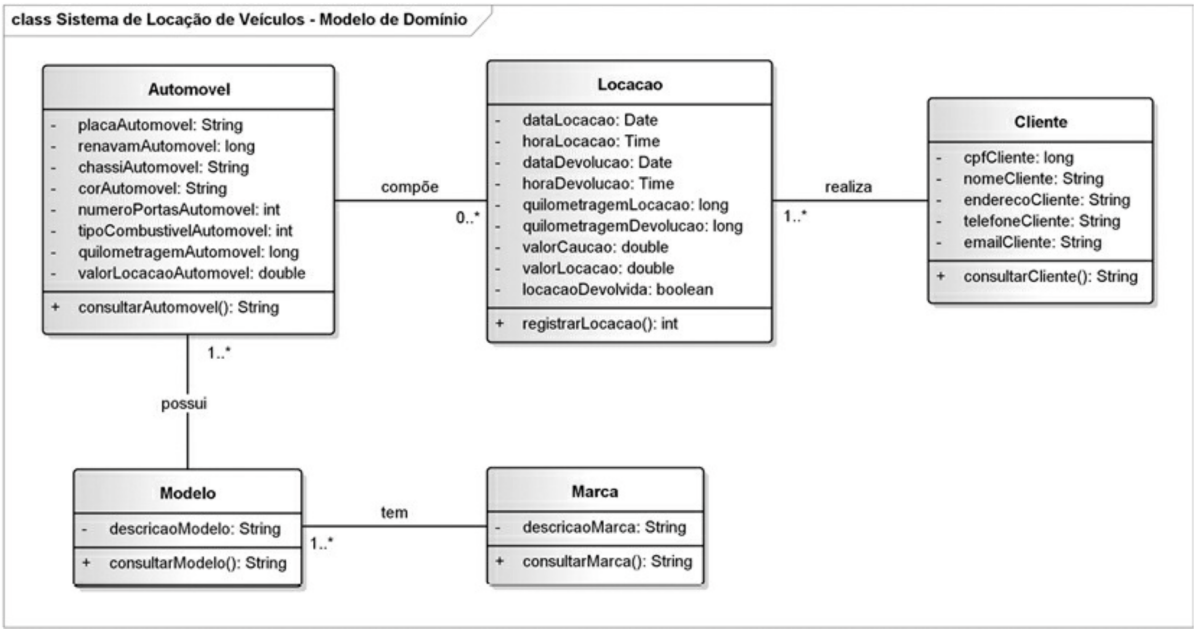

Figura 1: Diagrama de Classes Atualizado

1.2 Implementação do Modelo de Domínio

A seguir são apresentadas as classes em Java que implementam o modelo de domínio apresentado na sessão anterior. Essas classes foram implementadas a partir de uma versão anterior do diagrama de classes apresentado, portanto os nomes de atributos e métodos não estão seguindo o protocolo padrão do Java.

```java
public class Automovel {
    private String placa;
    private String cor;
    private int nroportas;
    private int tipo_combustivel;
    private long quilometragem;
    private long renavam;
    private String chassi;
    private double valor_locacao;

    // atributos das Associacoes
    private Modelo modelo;

    public Automovel(String placa, String cor, int nroportas, int tipo_combustivel, long quilometragem, lon
        this.placa = placa;
        this.cor = cor;
        this.nroportas = nroportas;
        this.tipo_combustivel = tipo_combustivel;
        this.quilometragem = quilometragem;
        this.renavam = renavam;
        this.chassi = chassi;
        this.valor_locacao = valor_locacao;
    }

```

```
25      public void setQuilometragem(long quilometragem) {
26          this.quilometragem = quilometragem;
27      }
28
29      public void setValor_locacao(double valor_locacao) {
30          this.valor_locacao = valor_locacao;
31      }
32
33      public double getValor_locacao() {
34          return valor_locacao;
35      }
36
37      public String conAuto(){
38          return this.toString();
39      }
40
41      public Modelo getModelo() {
42          return modelo;
43      }
44
45      public void setModelo(Modelo modelo) {
46          this.modelo = modelo;
47      }
48
49      @Override
50      public String toString() {
51          return "Automovel{" + "placa=" + placa + ", cor=" + cor + ", nroportas=" + nroportas + ", tipo_comb
52      }
53  }
```

```
1   import java.sql.Time;
2   import java.sql.Date;
3
4   public class Locacao {
5       private Date dt_locacao;
6       private Time hora_locacao;
7       private Date dt_devolucao;
8       private Time hora_devolucao;
9       private long quilometragem;
10      private double valor_caucao;
11      private double valor_locacao;
12      private int devolvido;
13
14      public Locacao(Date dt_locacao, Time hora_locacao, long quilometragem, double valor_caucao) {
15          this.dt_locacao = dt_locacao;
16          this.hora_locacao = hora_locacao;
17          this.quilometragem = quilometragem;
18          this.valor_caucao = valor_caucao;
19      }
20
21      public void setDt_devolucao(Date dt_devolucao) {
22          this.dt_devolucao = dt_devolucao;
23      }
24
```

```
25      public void setHora_devolucao(Time hora_devolucao) {
26          this.hora_devolucao = hora_devolucao;
27      }
28
29      public void setValor_locacao(double valor_locacao) {
30          this.valor_locacao = valor_locacao;
31      }
32
33      public void setDevolvido(int devolvido) {
34          this.devolvido = devolvido;
35      }
36
37      public int regLoc(){
38          // Nao sei o que esse metodo deveria fazer...
39          return 1;
40      }
41
42      @Override
43      public String toString() {
44          return "Locacao{" + "dt_locacao=" + dt_locacao + ", hora_locacao=" + hora_locacao + ", dt_devolucao
45      }
46  }
```

```
1   public class Cliente {
2       private long cpf_cli;
3       private String nom_cli;
4       private String end_cli;
5       private String tel_cli;
6       private String email_cli;
7
8       public Cliente(long cpf_cli, String nom_cli, String end_cli, String tel_cli, String email_cli) {
9           this.cpf_cli = cpf_cli;
10          this.nom_cli = nom_cli;
11          this.end_cli = end_cli;
12          this.tel_cli = tel_cli;
13          this.email_cli = email_cli;
14      }
15
16      public String conCli(){
17          return this.toString();
18      }
19
20      @Override
21      public String toString() {
22          return "Cliente{" + "cpf_cli=" + cpf_cli + ", nom_cli=" + nom_cli + ", end_cli=" + end_cli + ", tel
23      }
24  }
```

```
1   public class Marca {
2       private String descricao;
3
4       public Marca(String descricao) {
5           this.descricao = descricao;
```

```
    }

    public String conMarca(){
        return this.toString();
    }

    @Override
    public String toString() {
        return "Marca{" + "descricao=" + descricao + '}';
    }
}
```

```
public class Modelo {
    private String descricao;

    // Atributos das Associacoes
    private Marca marca;

    public Modelo(String descricao) {
        this.descricao = descricao;
    }

    public String conModelo(){
        return this.toString();
    }

    public Marca getMarca() {
        return marca;
    }

    public void setMarca(Marca marca) {
        this.marca = marca;
    }

    @Override
    public String toString() {
        return "Modelo{" + "descricao=" + descricao + '}';
    }
}
```

2 Atividade

Nesta atividade as equipes vão realizar ajustes nas classes do modelo de domínio apresentadas. Para adaptá-las ao diagrama de classes e aos padrões da linguagem Java. O código fonte das classes estará disponível no Moodle.

A avaliação se dará em vários níveis de conclusão do projeto dependendo do esforço dedicado pelas equipes. Procure apresentar telas da execução do programa para facilitar a avaliação.

Níveis de Avaliação: (Indique no relatório até que nível vocês chegaram)

- Mínimo: As classes implementadas de acordo com o modelo e um programa principal funcionando para demonstrar que os requisitos são atendidos mesmo que parcialmente. Não há necessidade de implementar interface gráfica ou persistência de dados. Mas o programa precisa ser funcional, mesmo que parcialmente. Sugestão, implementar uma classe Principal para executar os testes necessários para demonstrar os requisitos implementados.
- Intermediário: Os requisitos do projeto precisam estar implementados integralmente e pelo menos um item avançado precisa estar implementado, podendo ser a Interface Gráfica ou a Persistência de Dados (no caso da persistência de dados pode ser apenas parcial, por exemplo, apenas para os Clientes). Nos dois casos as equipes devem pesquisar como esses itens avançados são implementados em Java e colocar o que foi pesquisado no relatório.
- Avançado: Os dois itens avançados devem ser implementados, mesmo que parcialmente.

3 Entregável

O relatório da aula de hoje deve conter os seguintes itens:

1. Implementação das classes de domínio ajustadas.
2. Implementação dos itens avançados, caso seja feito.

Referências

[1] G.T.A. Guedes. *UML 2 - Uma Abordagem Prática*. Novatec Editora, 2018.