Fatec São Caetano do Sul Antonio Russo

Centro Paula Souza

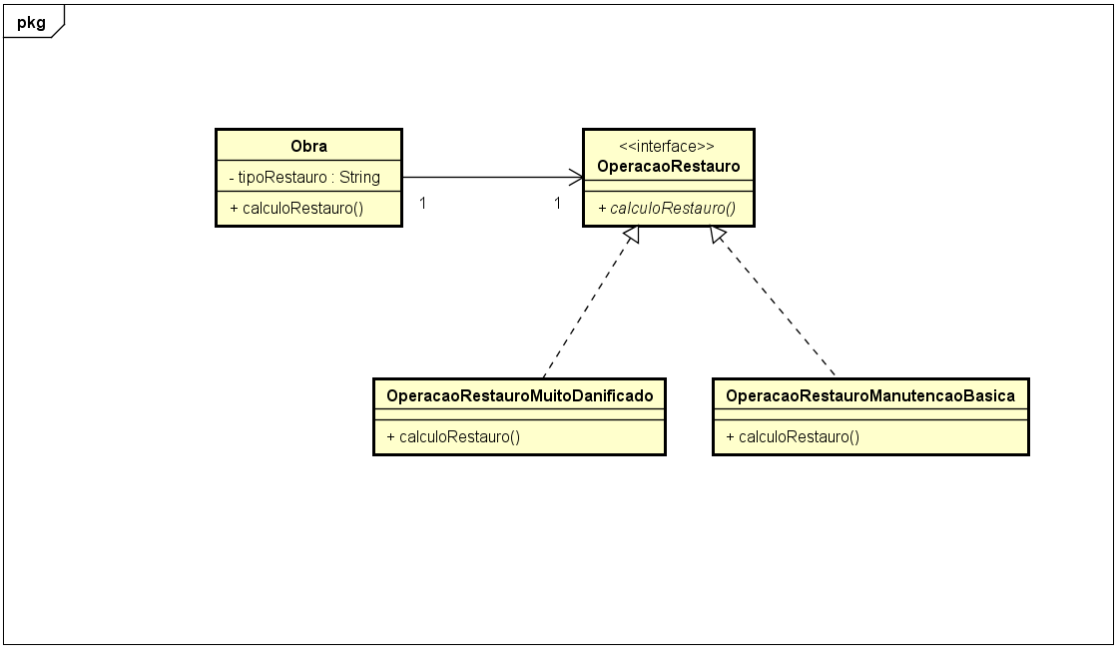
GOVERNO DE SÃO PAULO

Engenharia de Software 3

Lista 3

Andrei Lucas Gonçalves	1680481521025
Daniel Augusto Scordamaglio	1680481521030
Diego de Melo Gonzaga	1680481521036
Elias Kyoharu Sanai	1680481521016
Henrique Fernandes	141682263

1. Com base no diagrama de classes de projeto refinado nesta lista, modele o padrão de projeto Strategy. Qualo propósito desse padrão no diagrama?

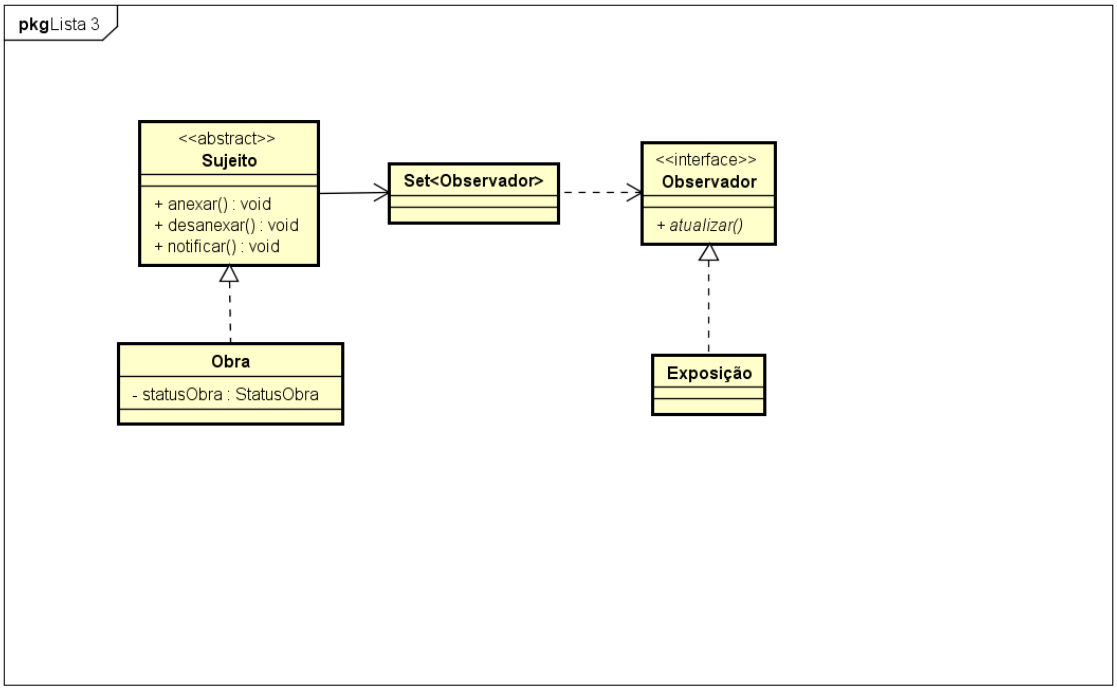


O propósito desse padrão no diagrama é poder alterar em tempo de execução um comportamento (nesse caso o método da Obra calculoRestauro())

2. Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar o padrão de projeto Strategy.

```
a. class Obra {
b.     private String tipoRestauro;
c.     private OperacaoRestauro operacao;
d.
e.     public double calculoRestauro() {
f.         return operacao.calculoRestauro(Obra obra);
g.     }
h. }
i.
j. interface OperacaoRestauro {
k.     double calculoRestauro(Obra obra);
l. }
m.
n. class OperacaoRestauroManutencaoBasica {
o.     public double calculoRestauro(Obra obra) {
p.         //Aqui ele realiza o cálculo
q.     }
r. }
```

3. Com base no diagrama de classes de projeto refinado nesta lista, modele o padrão de projeto Observer. Qual o propósito desse padrão no diagrama?

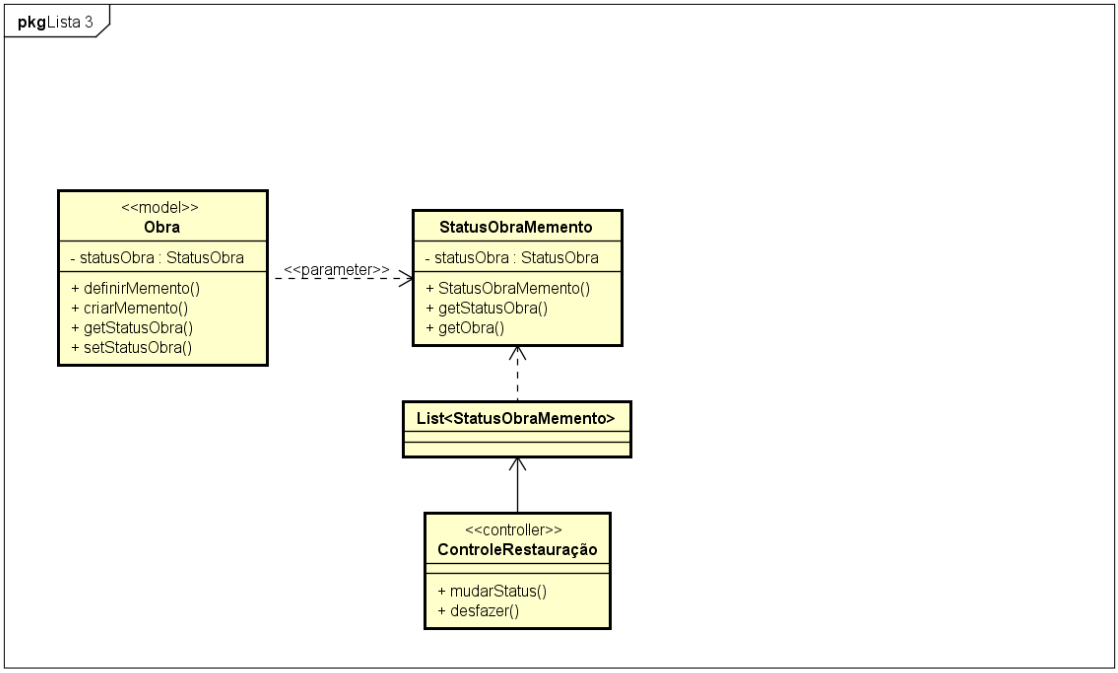


O propósito desse pattern é fazer com que uma classe (nesse caso a classe Exposição) fique Observando outra classe denominada Sujeito (Subject, nesse caso a classe Obra), quando a classe do Sujeito ocorre uma mudança de status (nesse caso do StatusObra), então ela notifica para todos os Observadores que ocorreu uma mudança de Status

4. Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar o padrão de projeto Observer.

```
class Sujeito {
    Set<Observador> observadores = new HashSet<Observador>();

    public void anexar(Observador observador) {
        observadores.add(observador);
    }

    public void desanexar(Observador observador) {
        observadores.remove(observador);
    }

    public void notificar() {
        for (Observador ob : observador) {
            ob.atualizar();
        }
    }
}

interface Observador {
    public void atualizar();
}

class Obra extends Sujeito {
    private StatusObra statusObra;
}

class Exposicao implements Observador {
    public void atualizar();
}
```

5. Com base no diagrama de classes de projeto refinado nesta lista, pesquise e modele o padrão de projeto Memento. Qual o propósito desse padrão no diagrama?



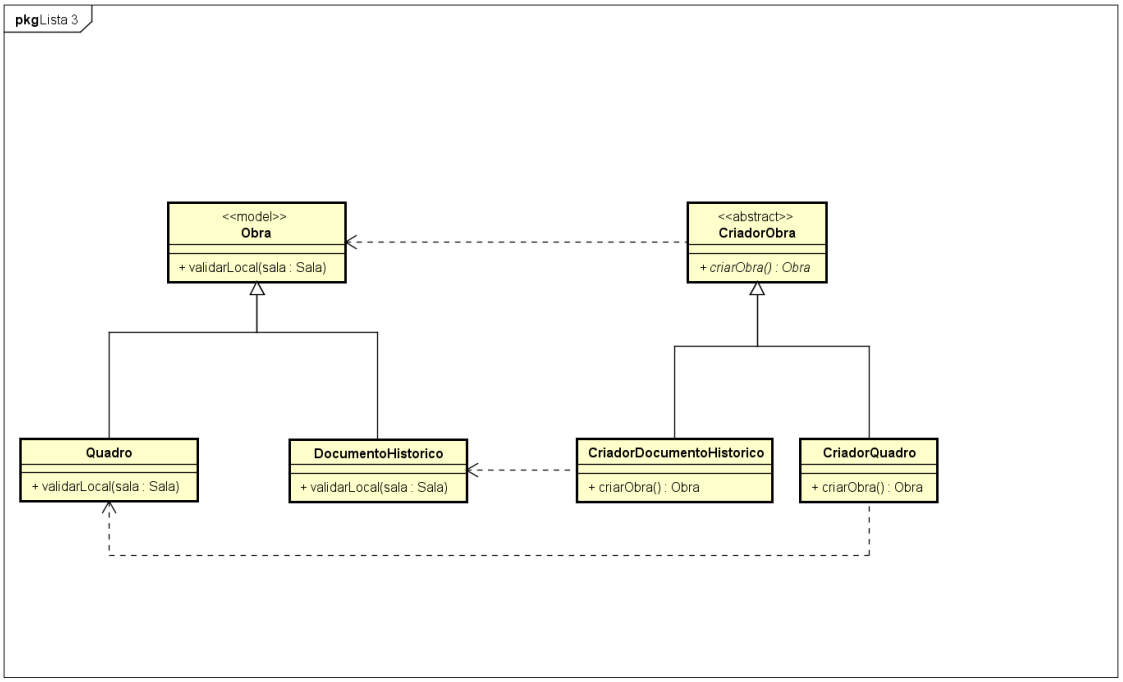
O propósito desse design pattern é nesse caso salvar o Status da Obra, vários status seguidos para que possa usar o comando desfazer, geralmente isso ocorre em alguma interface gráfica para que o usuário possa usar o famoso ctrl+z (no caso do Windows)

6. Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar o padrão de projeto Memento.

```
class Obra {
    private StatusObra statusObra;

    public void definirMemento(StatusObraMemento memento) {
        statusObra = memento.getStatusObra();
    }

    public StatusObraMemento criarMemento() {
        return new StatusObraMemento(this, statusObra);
    }

    public StatusObra getStatusObra() {
        return statusObra;
    }

    public void setStatusObra(StatusObra value) {
        statusObra = value;
    }
}

class StatusObraMemento {
    private StatusObra statusObra;
    private Obra obra;

    public StatusObraMemento(Obra obra, StatusObraMemento status) {
        this.obra = obra;
        statusObra = status;
    }

    public StatusObra getStatusObra() {
        return statusObra;
    }

    public Obra getObra() {
        return obra;
    }
}

class ControleRestauracao {
    List<StatusObraMemento> mementos = new
ArrayList<StatusObraMemento>();

    public void mudarStatus(Obra obra, StatusObraMemento novoStatus) {
        mementos.add(obra.criarMemento());
        obra.setStatusObra(novoStatus);
    }
```

```
    public void desfazer(Obra obra, StatusObraMemento novoStatus) {
        //Pega o ultimo
        StatusObraMemento ultimo = mementos.get(list.size() - 1);
        ultimo.getObra().definirMemento(ultimo);
        mementos.remove(ultimo);
    }
}
```

7. Com base no diagrama de classes de projeto refinado nesta lista, modele o padrão de projeto Factory Method. Qual o propósito desse padrão no diagrama?

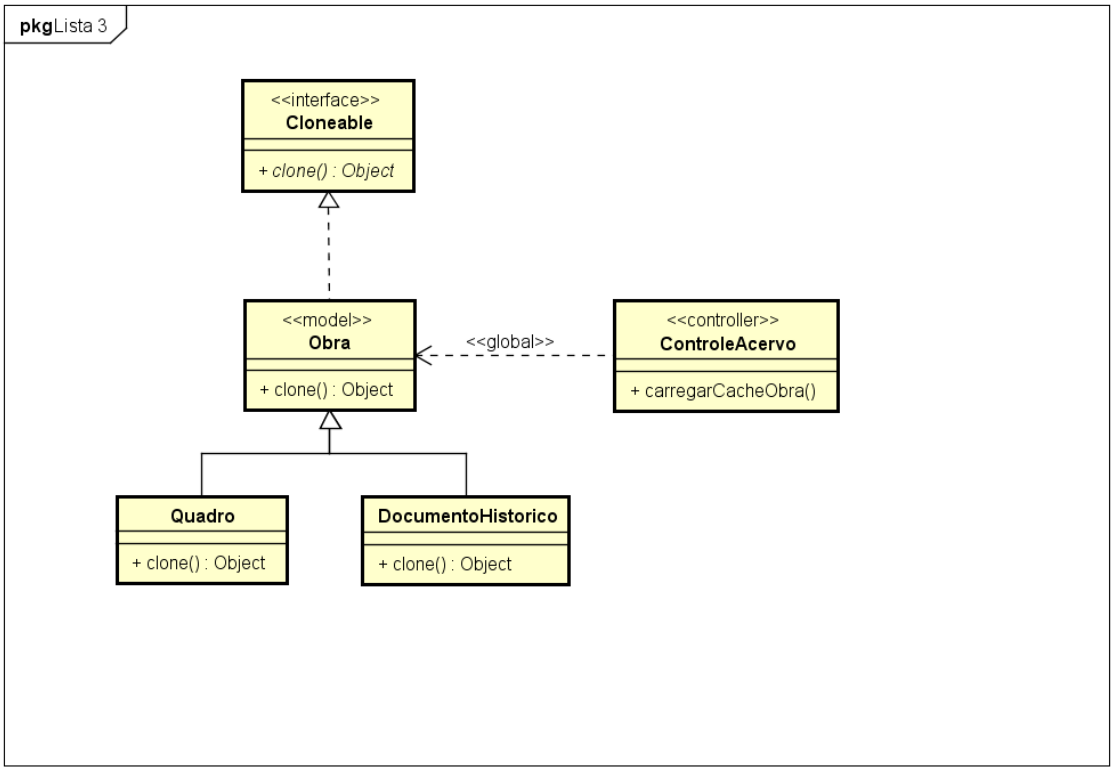


O propósito desse design pattern é remover a obrigação das classes consumidoras (nesse caso Obra) de instanciar a classe, assim reduzindo o acoplamento e também promovendo mais flexibilidade

8. Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar o padrão de projeto Factory Method.

```
class Obra {
    //Atributos e métodos da classe Obra
}

class Quadro extends Obra {
    //Atributos e métodos da classe Quadro
}

abstract class CriadorObra {
    public abstract Obra criarObra();
}

class CriadorQuadro extends CriadorObra {
    @Override
    public Obra criarObra() {
        return new Quadro();
    }
}
```

9. Com base no diagrama de classes de projeto refinado nesta lista, pesquise e modele o padrão de projeto Prototype. Qual o propósito desse padrão no diagrama?

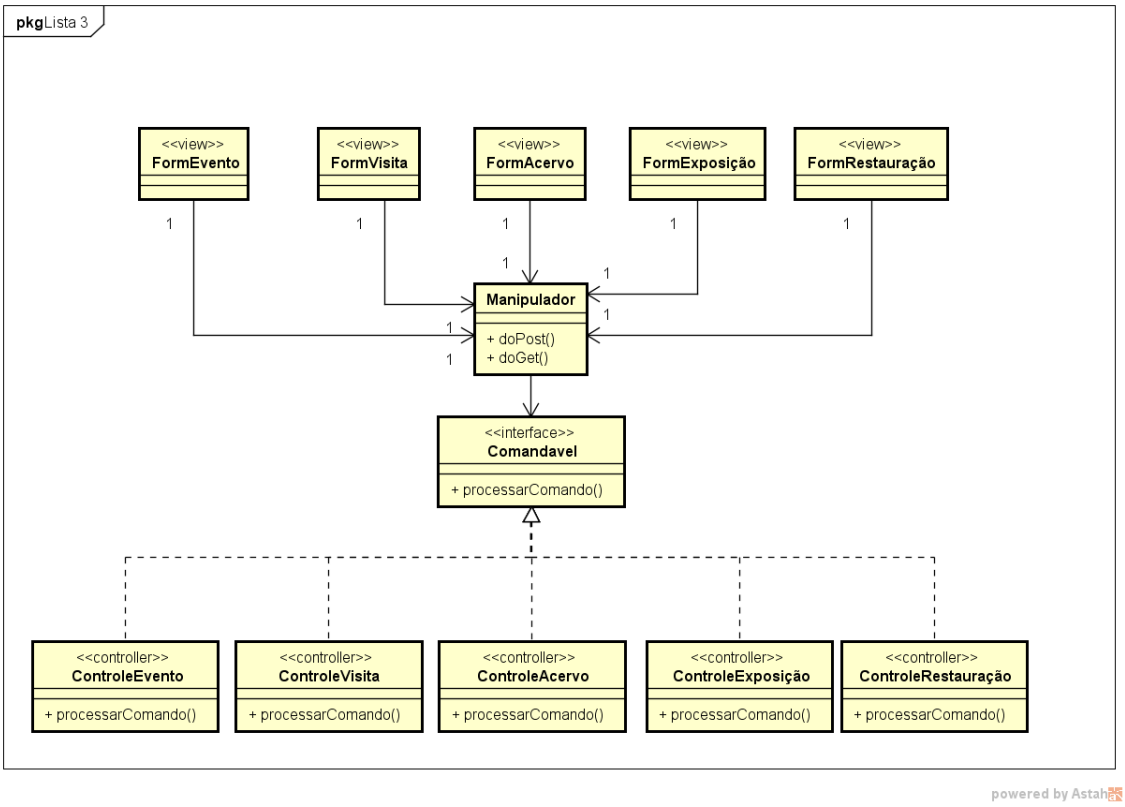


O propósito desse design pattern nesse diagrama é promover um protótipo (clone ou cópia, nesse caso clone) para que possa ser alterado sem alterar a instancia real, ou nesse caso, para armazenar como cache para usos posteriores (levando em consideração que a instancia não haverá mudanças nela)

10. Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar o padrão de projeto Prototype.

```
class Obra implements Cloneable {
    public Object clone() throws CloneNotSupportedException {
        return super.clone();
    }
}

class ControleAcervo {
    public List<Obra> carregarCacheObra() {
        //Aqui ele usaria alguma instancia global, framework ou algo do
genero para carregar as
        //instancias de Obra que precisam ser clonadas ou copiadas (nesse
caso clonadas), ou seja,
        //usar o design pattern prototype para realizar esse processo
    }
}
```

11. Com base no diagrama de classes de projeto refinado nesta lista, modele o padrão de projeto Front Controller. Qual o propósito desse padrão no diagrama?

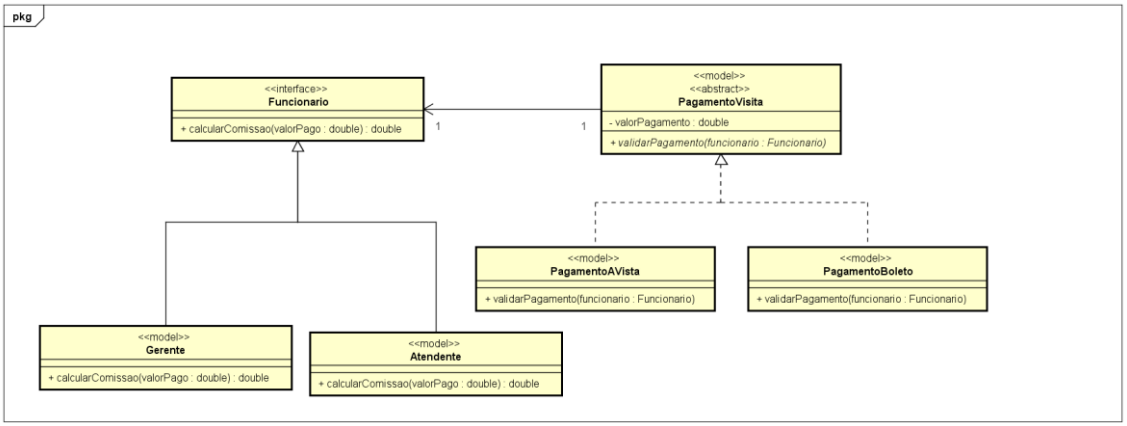


O propósito desse design pattern é promover uma interface para que assim as fronteiras acessem as controles (geralmente o uso é para web), nesse diagrama todas as fronteiras usariam métodos GET e POST para acessar e enviar as informações para as controles

12. Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar o padrão de projeto Front Controller.

```
class FormEvento {
    //Extends e implements dependendo da onde essa view está implementada
(desktop, web, mobile)
    //Atributos e Métodos do formulário
}

class Manipulador {
    //Geralmente chamada Handler
    public void doPost(String controllerName, String... parameters) {
        //Aqui chamaria o controlador específico dependendo do nome
requisitado
    }
    public void doGet(String controllerName, String... parameters) {
        //O mesmo do Post com exceção das características de Get
    }
}

interface Comandavel {
    Boolean processarComando(String... parameters);
}

class ControleEvento implements Comandavel {
    public Boolean processarComando(String... parameters) {
        //Aqui executaria o comando requisitado
    }
}
```

13. Com base no diagrama de classes de projeto refinado nesta lista, pesquise e modele o padrão de projeto Bridge. Qual o propósito desse padrão no diagrama?



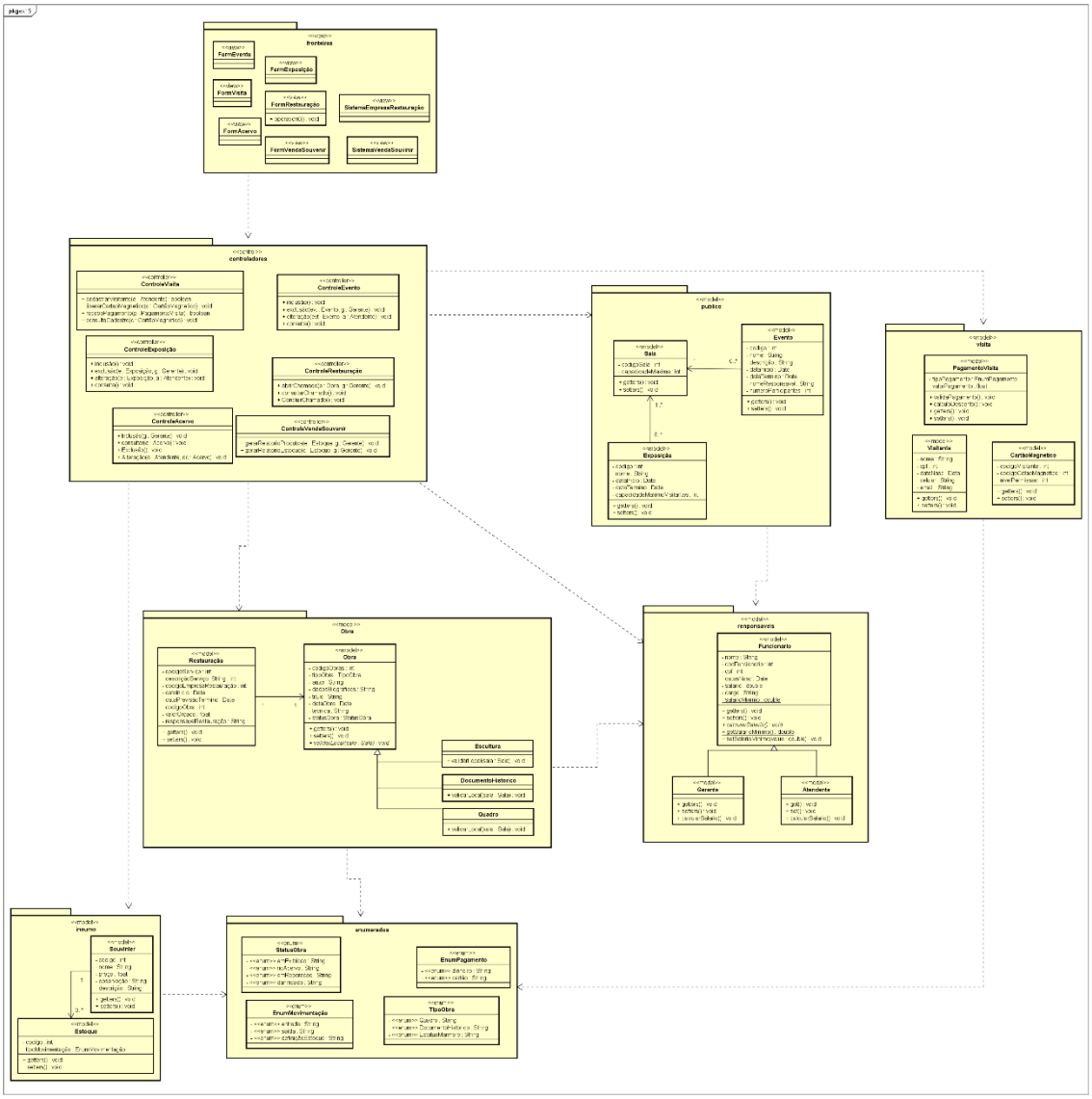
O propósito desse design pattern é reduzir o acoplamento entre a abstração e a implementação (nesse caso a abstração PagamentoVisita e a implementação Funcionario) onde por meio desse pattern realiza uma ponte (Bridge) entre essas classes

Observação: Nesse exercício a classe antes modelo Funcionario se tornou uma interface para que assim ela tivesse somente os métodos aplicaveis para o design pattern e ela na implementação poderia possuir muitos mais outros métodos

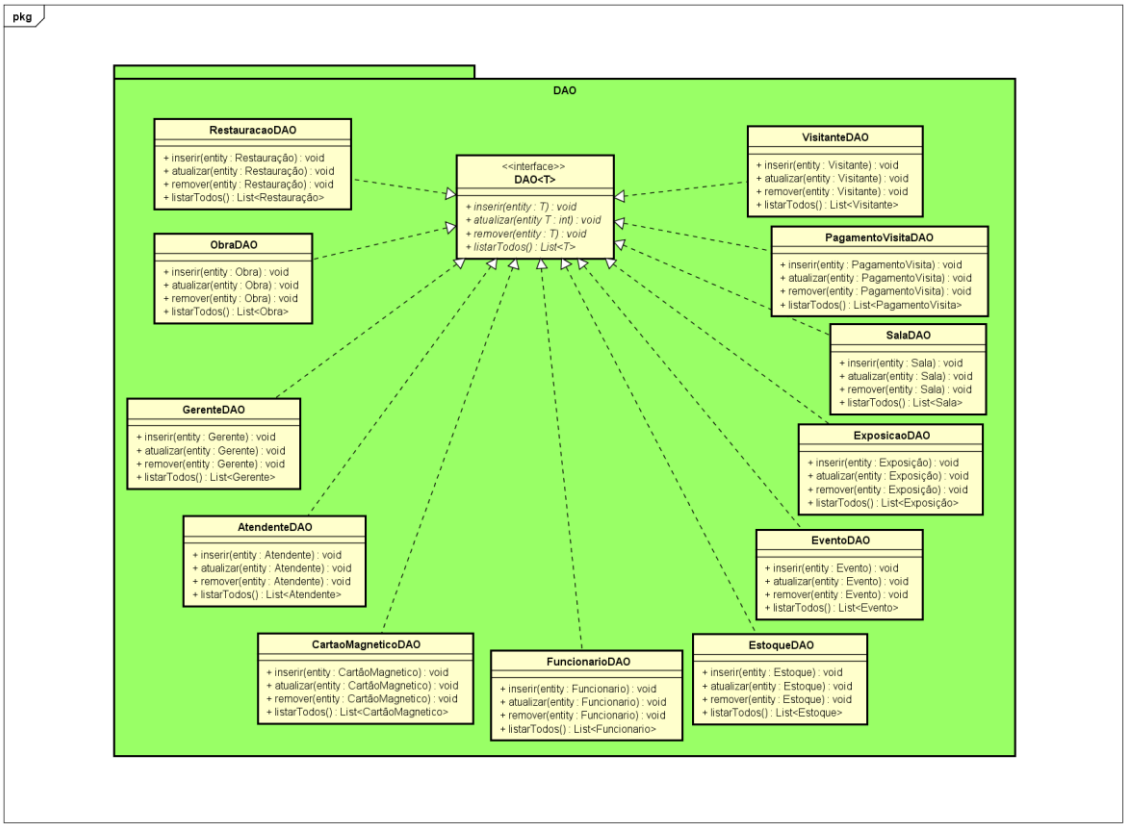
14. Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar o padrão de projeto Bridge.

```
interface Funcionario  {
    double calcularComisao(double valor);
}

class Gerente implements Funcionario {
    public double calcularComisao(double valor) {
        //Faria o calculo da comissao aqui
    }
}

class PagamentoVisita {
    protected double valorPagamento;
    protected Funcionario funcionario;

    public abstract Boolean validarPagamento(Funcionario funcionario);
}

class PagamentoAVista {
    @Override
    public Boolean validarPagamento(Funcionario funcionario) {
        double comissao =
funcionario.calcularCommisao(valorPagamento);
        //Outras validações adicionais
    }
}
```

15. Com base no diagrama de classes de projeto refinado nesta lista, modele os pacotes (subsistemas) e faça a alocação das classes em cada pacote. Cada pacote deve mostrar as classes detalhadas com atributos e métodos. Neste exercício, deve constar um pacote de classes de visão, um pacote de classes de controle, no mínimo três pacotes de classes de classes de modelo e um pacote de classes enumeradas.



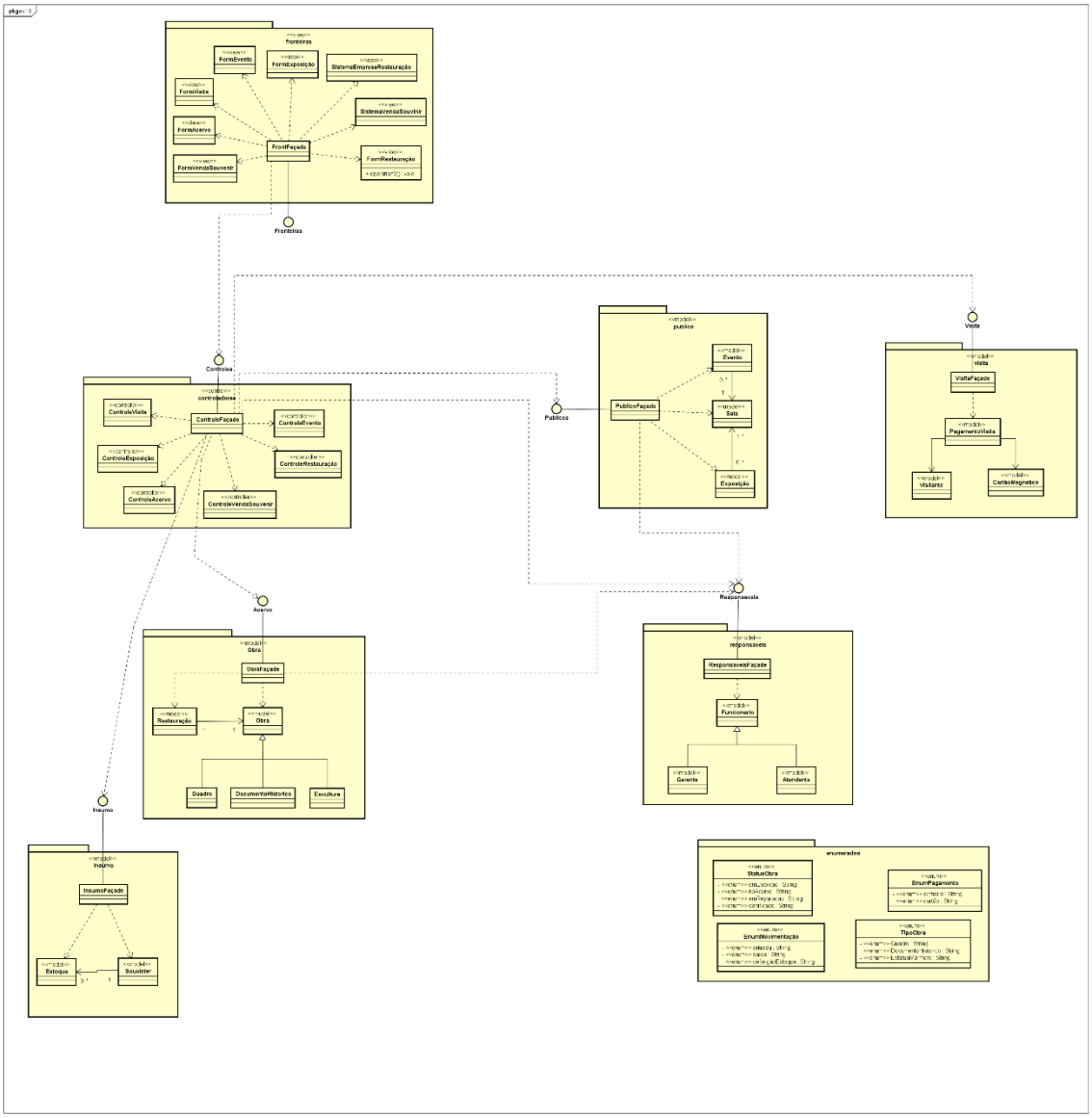
16. Construa o pacote de Persistência e faça a alocação das classes DAO no pacote. Este pacote deve mostrar as classes detalhadas com métodos.



17. Apresente a estrutura básica de código para implementar o pacote de Persistência (DAO).

```
interface DAO<T> {
    void inserir(T entity);
    void atualizar(T entity);
    void remover(T entity);
    List<T> listarTodos();
}

class ObraDAO implements DAO<Obra> {
    public void inserir(Obra entity) {
        //Faria conexão com banco e enviaria os dados
    }
    public void atualizar(Obra entity) {
        //Faria conexão com banco e enviaria os dados
    }
    public void remover(Obra entity) {
        //Faria conexão com banco e enviaria os dados
    }
    public List<Obra> listarTodos() {
        //Faria conexão com banco e receberia os dados
    }
}
```

18. Após a identificação dos pacotes (subsistemas) e alocação das classes, modele um diagrama de pacotes com os devidos relacionamentos, aplicando o padrão de projeto Façade no pacote de controle e nos pacotes de modelo. Neste diagrama, os pacotes devem mostrar somente os nomes das classes, sem a necessidade de apresentar os detalhes (atributos e/ou métodos) das classes.



19. Apresente a estrutura básica de código em JAVA, C# ou C++ para implementar o padrão de projeto Façade.

```
interface Controles {
    void liberarCartaoManetico(CartaoMagnetico cartao);
}


class ControleFaçade implements Controles {
    public void liberarCartaoManetico(CartaoMagnetico cartao) {
        //Pega a instancia do ControleVisita e chama o controle visita
    }
}

class ControleVisita {
    public void liberarCartaoMagnetico(CartaoMagnetico cartao) {
        //Realiza o método
    }
}
```

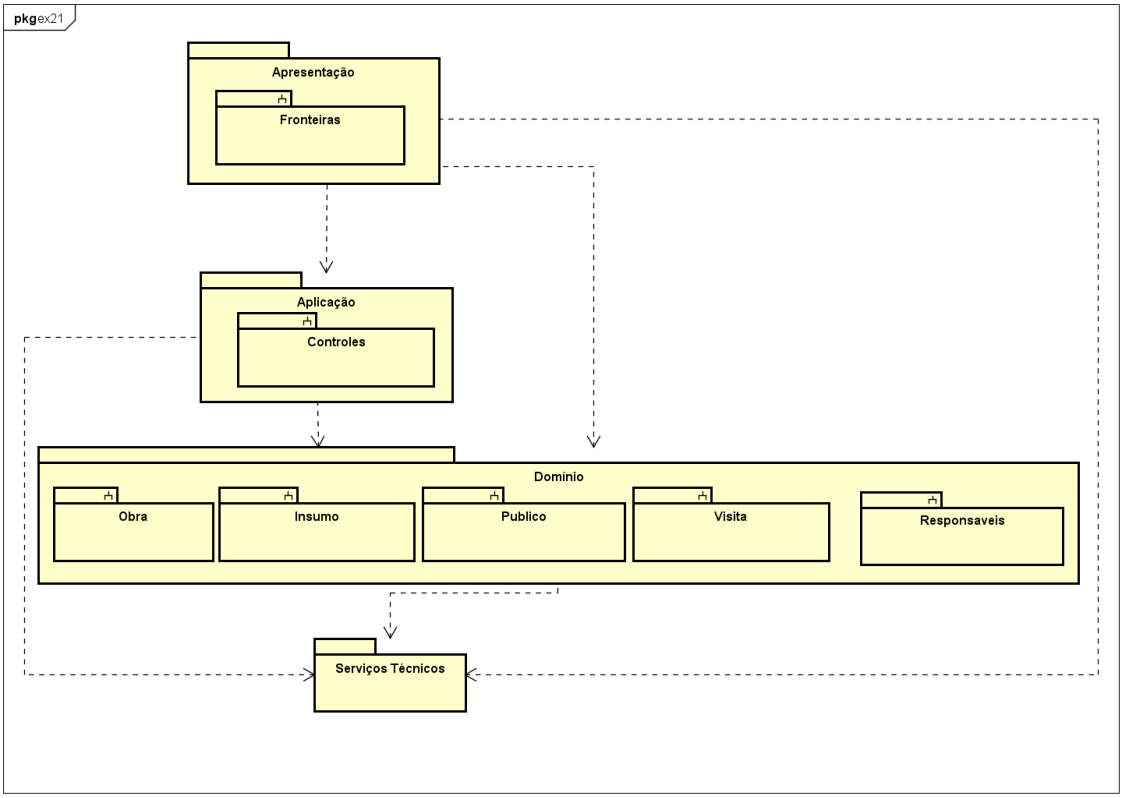
20. Apresente três diferenças conceituais entre os padrões de projeto Façade e Mediator. Justifique a tua resposta.

• Enquanto Mediator é um design pattern de comportamento, Façade é de estrutura

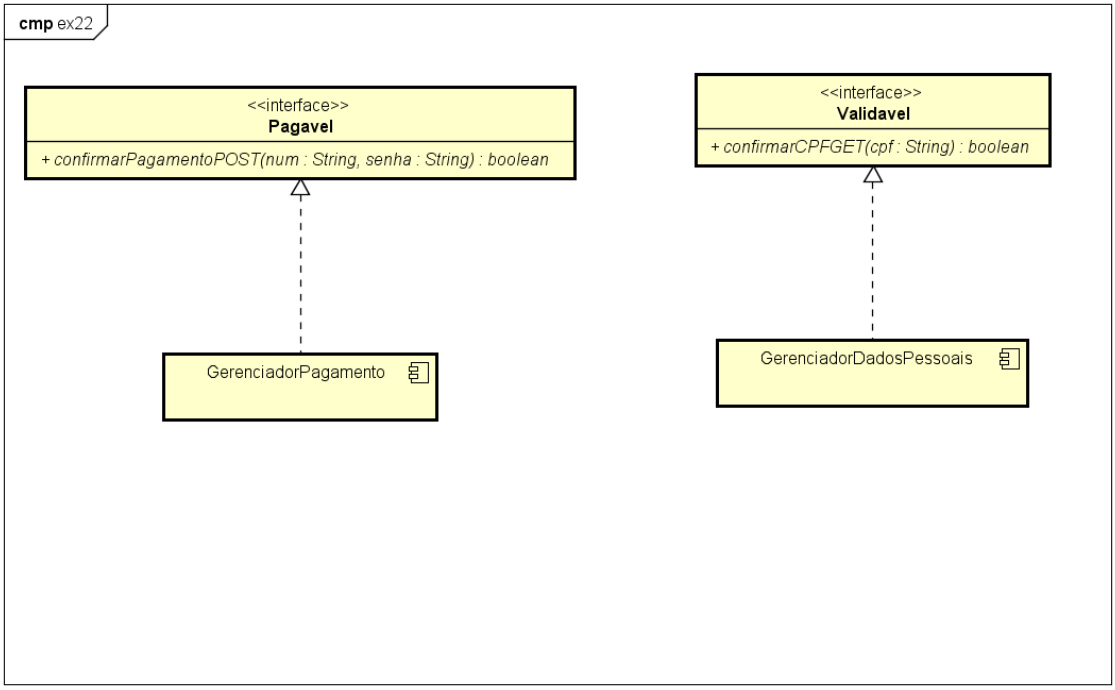
• Façade promove a ideia de subsistemas encapsulandos as classes para que não se saiba sobre tudo do sistema tornado a comunicação unilateral, enquanto no Mediator a ideia é encapsular os comportamentos coletivos independente de subsistemas, podendo haver comunicação bilateral

• Mediator tem maior probabilidade de reuso por conta que as classes podem estar num mesmo pacote, enquanto o Façade divide em subsistemas a lógica do programa, consequentemente são criado mais classes.

21. Faça a alocação dos pacotes (subsistemas) nas camadas de software apresentadas em aula. As camadas devem ser representadas no sentido vertical e com arquitetura aberta.



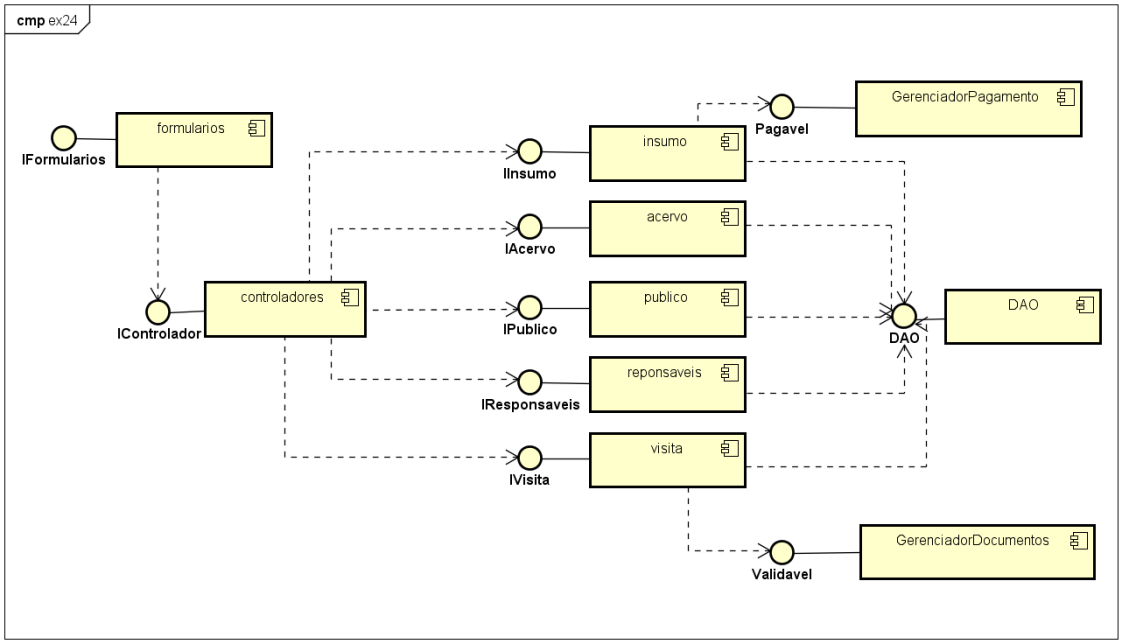
22. Modele dois componentes com interface expandida, sendo um para gerenciar o pagamento por meio de cartão e um para validar o CPF do hóspede. Vale ressaltar que esses componentes são serviços terceirizados e que podem ter sido desenvolvidos numa plataforma diferente.



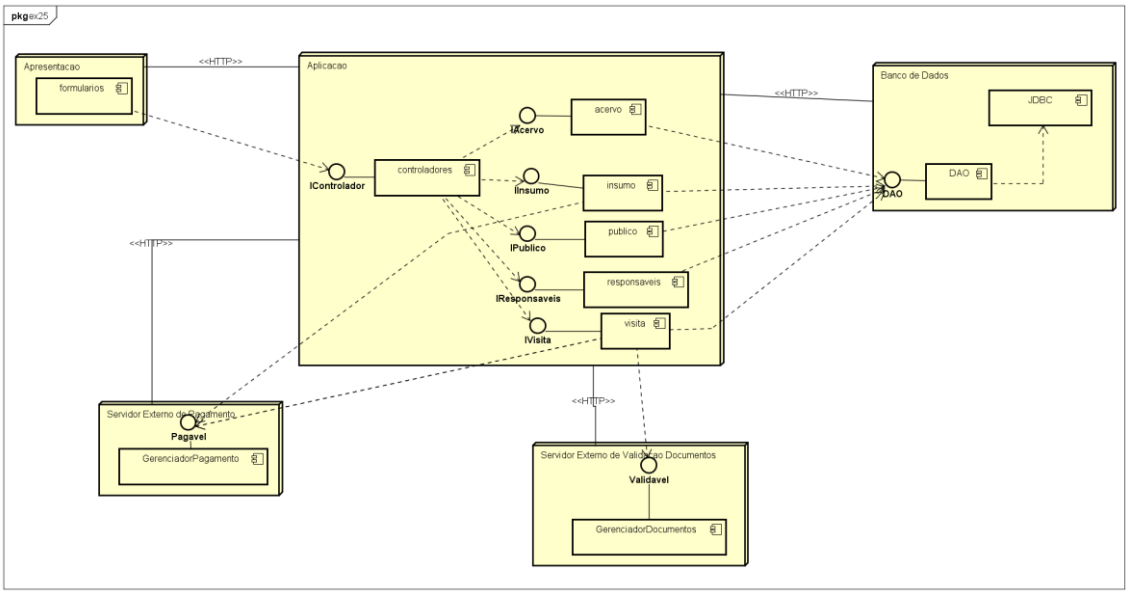
23. Apresente a estrutura básica de código da sua aplicação em JAVA, C# ou C++ consumindo o serviço de cada um dos dois componentes de terceiros por meio de web services REST. A apresentação do formato JSON na estrutura de código será necessária também.

```
class GerencidorPagamento implements Pagavel {
    public static Boolean confirmarPagamentoPOST(String num, String
senha);
}

class ControleVisita {
    public Boolean confirmarPagamento(String num, String senha) {
        String json = GerencidorPagamento.confirmarPagamentoPOST(num,
senha);

        if(JSON.parse(json)["status"] == "confirmado")
            return true;
        else
            return false;
    }
}

class GerencidorDadosPessoais implements Validavel {
    public static Boolean confirmarCPFGET(String cpf);
}

class DadosPessoaisUtil {
    public static Boolean confirmarCPFGET(String cpf) {
        String json = GerencidorDadosPessoais.confirmarCPFGET(cpf);

        if(JSON.parse(json)["status"] == "valido")
            return true;
        else
            return false;
    }
}
```

24. A partir da visão dos pacotes (subsistemas) e dos componentes de terceiros, construa o diagrama de componentes. Neste exercício, o pacote de classes enumeradas não precisa ser transformado para um componente e as classes de controle do pacote de controle podem ficar com seu respectivo pacote de classes de modelo, no mesmo componente.



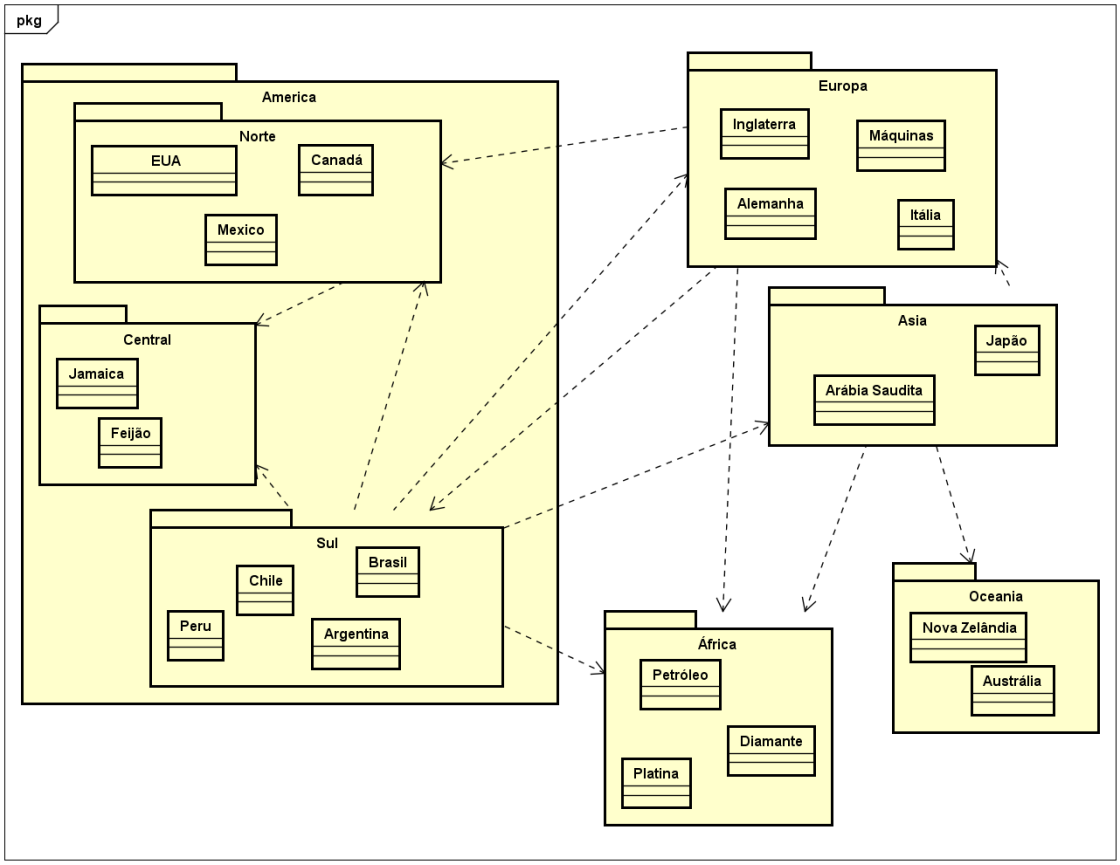
25. Com base na alocação dos pacotes (subsistemas) nas camadas de software e no diagrama de componentes, construa o diagrama de implantação distribuindo os componentes em seus respectivos nós. O seu projeto tem quantas camadas? Justifique a tua resposta.



O projeto contém 5 camadas, considerando duas externas que são utilizadas pela camada de Aplicação. Existe a camada de Apresentação que é utilizada para o cliente acessar a Aplicação. E também a camada do Banco de Dados que é onde os dados são guardados.

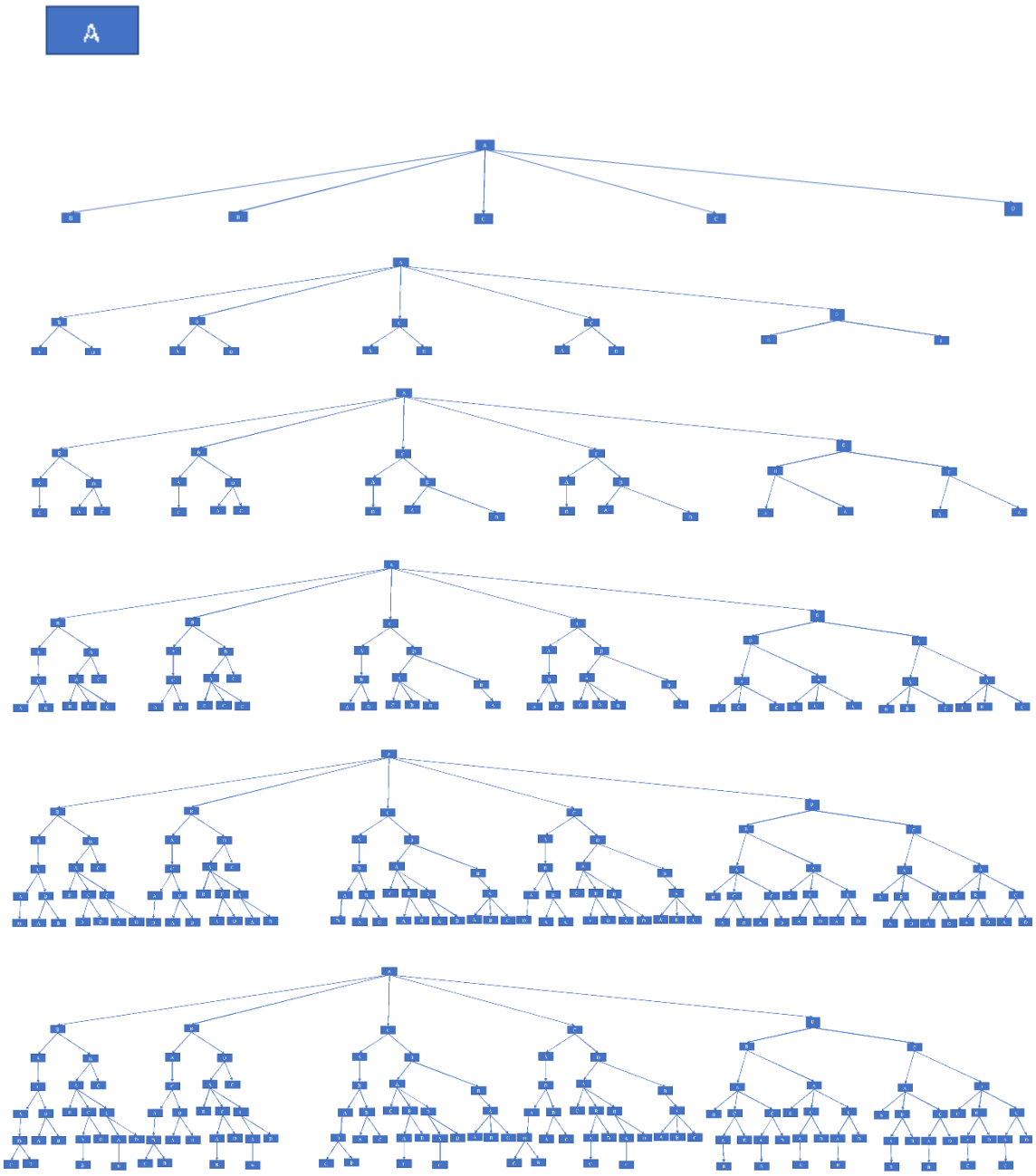
Parte B: Atividades de Abstração

Atv 1- Abstraia o Mapa Mundi e modele um diagrama de pacotes com os devidos relacionamentos. Somente o nome de cada classe alocada no devido pacote é suficiente para este exercício. Justifique a tua resposta.



Devido a globalização do mundo, cada país depende de outro para obter certos produtos. Tal como a Arábia Saudita exporta toneladas de Petróleo para diversos países em outros continentes, assim como a Alemanha exporta Máquinas. Cada país faz algum tipo de exportação, às vezes dois países exportam o mesmo produto, porém devido a proximidade podem ter clientes diferentes dentro e fora do continente.

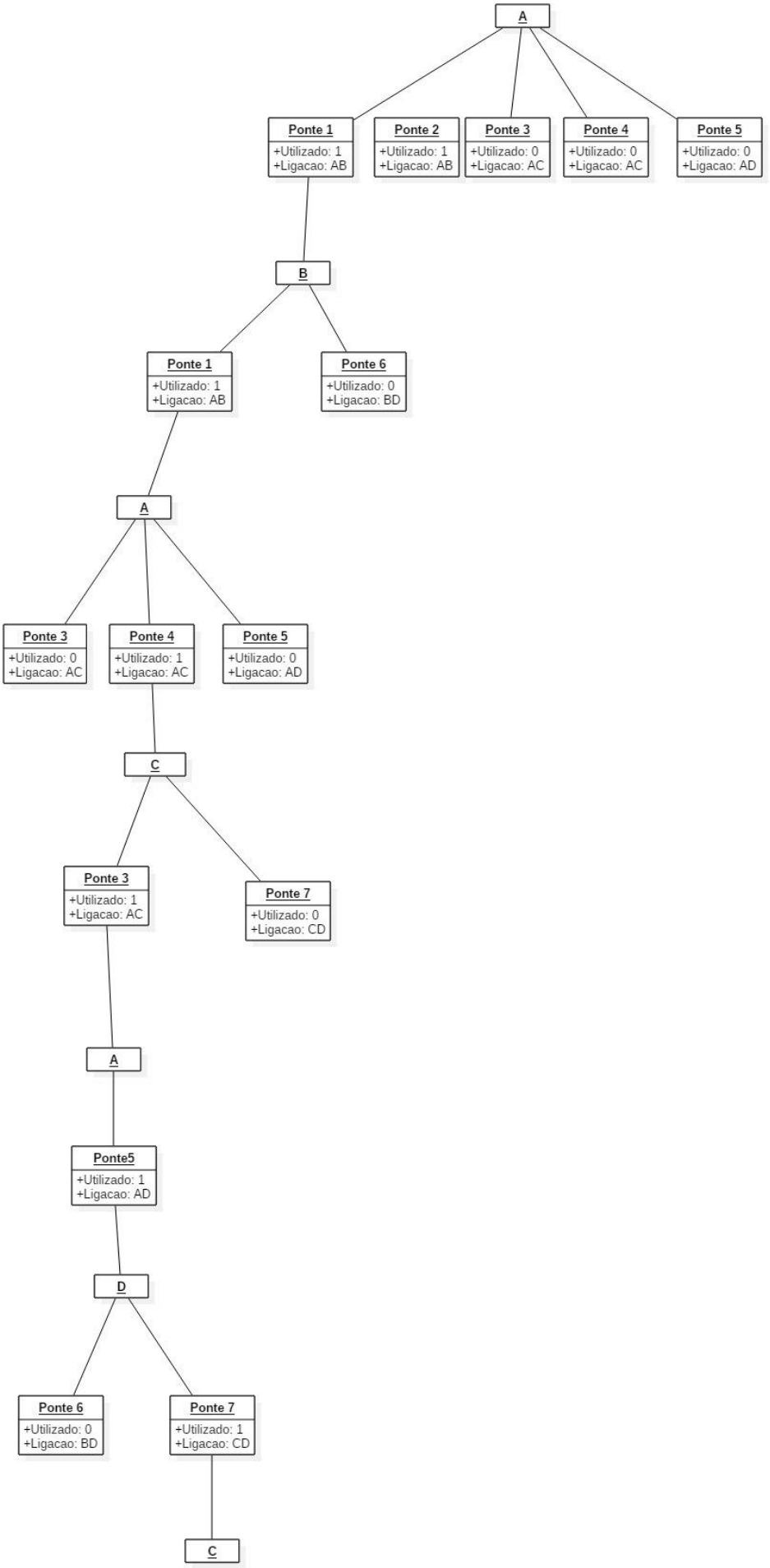
Atv 2- Abstraia o cenário e a imagem sobre a cidade de Caliningrado na Rússia. Escolha uma estrutura de árvore e represente graficamente como essa árvore trabalharia em tempo de execução para tentar resolver o problema. Justifique a tua resposta. Modele em paralelo um diagrama de objetos para representar a tentativa de resolução do problema em tempo de execução. Em 1736, Leonhard Euler ficou intrigado com um problema entre os habitantes de Königsberg (localizada ao leste da antiga Prussia chamada mais tarde de Caliningrado na Rússia). O rio que atravessa a cidade bifurca-se em torno de uma ilha e sete pontes atravessam a ilha conforme mostra a figura abaixo. O problema é decidir se uma pessoa poderia passar por toda a cidade cruzando cada ponte apenas uma vez. Tal solução poderia ser encontrada através de tentativa e erro, Euler todavia representou a situação por um grafo onde as pontes são as arestas e as partes em terra os vértices.



(IMAGENS ANEXADAS EM FORMATO ZIP PARA MELHOR VISUALIZAÇÃO)

Justificativa:

Cada ilha contém mais de uma ponte, portanto existem várias possibilidades para passar para outra utilizando pontes diferentes. A imagem em execução demonstra para cada descida uma possibilidade distinta, ou seja, um caminho único sem utilizar uma ponte que foi utilizada anteriormente.


A
Ponte 1
+Utilizado: 1
+Ligacao: AB
Ponte 2
+Utilizado: 1
+Ligacao: AB
Ponte 3
+Utilizado: 0
+Ligacao: AC
Ponte 4
+Utilizado: 0
+Ligacao: AC
Ponte 5
+Utilizado: 0
+Ligacao: AD
B
Ponte 1
+Utilizado: 1
+Ligacao: AB
Ponte 6
+Utilizado: 0
+Ligacao: BD
A
Ponte 3
+Utilizado: 0
+Ligacao: AC
Ponte 4
+Utilizado: 1
+Ligacao: AC
Ponte 5
+Utilizado: 0
+Ligacao: AD
C
Ponte 3
+Utilizado: 1
+Ligacao: AC
Ponte 7
+Utilizado: 0
+Ligacao: CD
A
Ponte5
+Utilizado: 1
+Ligacao: AD
D
Ponte 6
+Utilizado: 0
+Ligacao: BD
Ponte 7
+Utilizado: 1
+Ligacao: CD
C