Noções de projeto orientado a objetos - camadas

Prof. Gustavo Wagner

(alterações)

Prof. Tiago Massoni

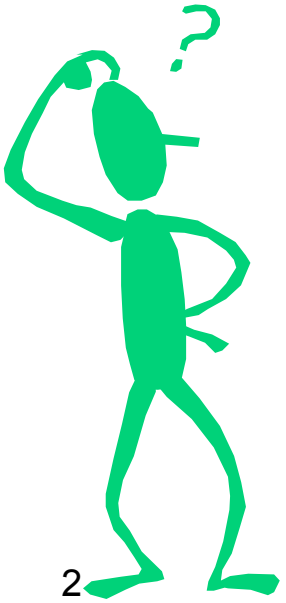
(Slides originais)

Sistemas de Informação

IESP

Tudo Bem se Vemos o Programa como uma Caixa Preta...

Funciona! Por que vou me preocupar?



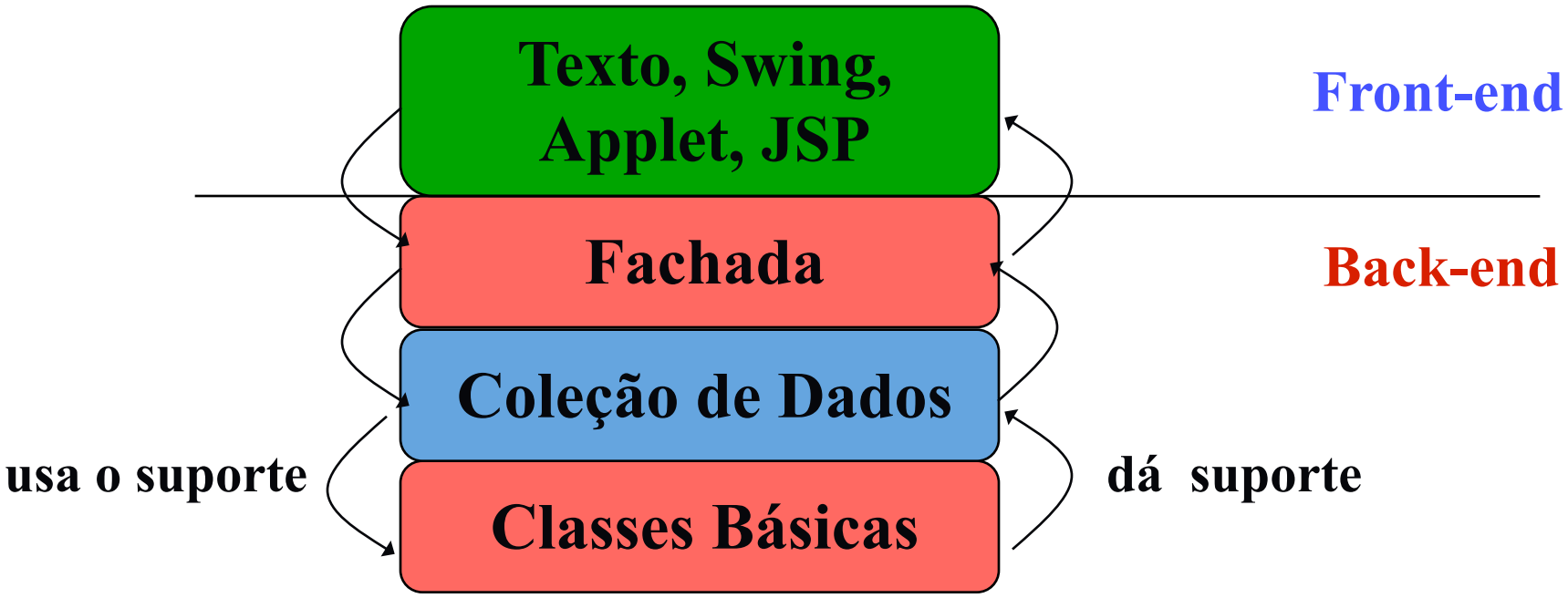
Mas Internamente...

Quando Eu Quiser Analisar e Modificar o Programa...

Péssimo Exemplo

```
public class Banco {
   ...
   public void CreditarEmConta() {
           //apresenta tela inicial;aguarda entrada de
   dados;
           //processa numero e valor fornecidos pelo
   usuario
           //checa se conta existe,senao mensagem de erro
           //se ok,busca conta em arquivo pelo numero
           //dada a conta,acresce valor ao saldo anterior
           //retorna mensagem de sucesso da operacao
   }
}
```

Melhor Ver o Programa como um Bolo (em Camadas)!



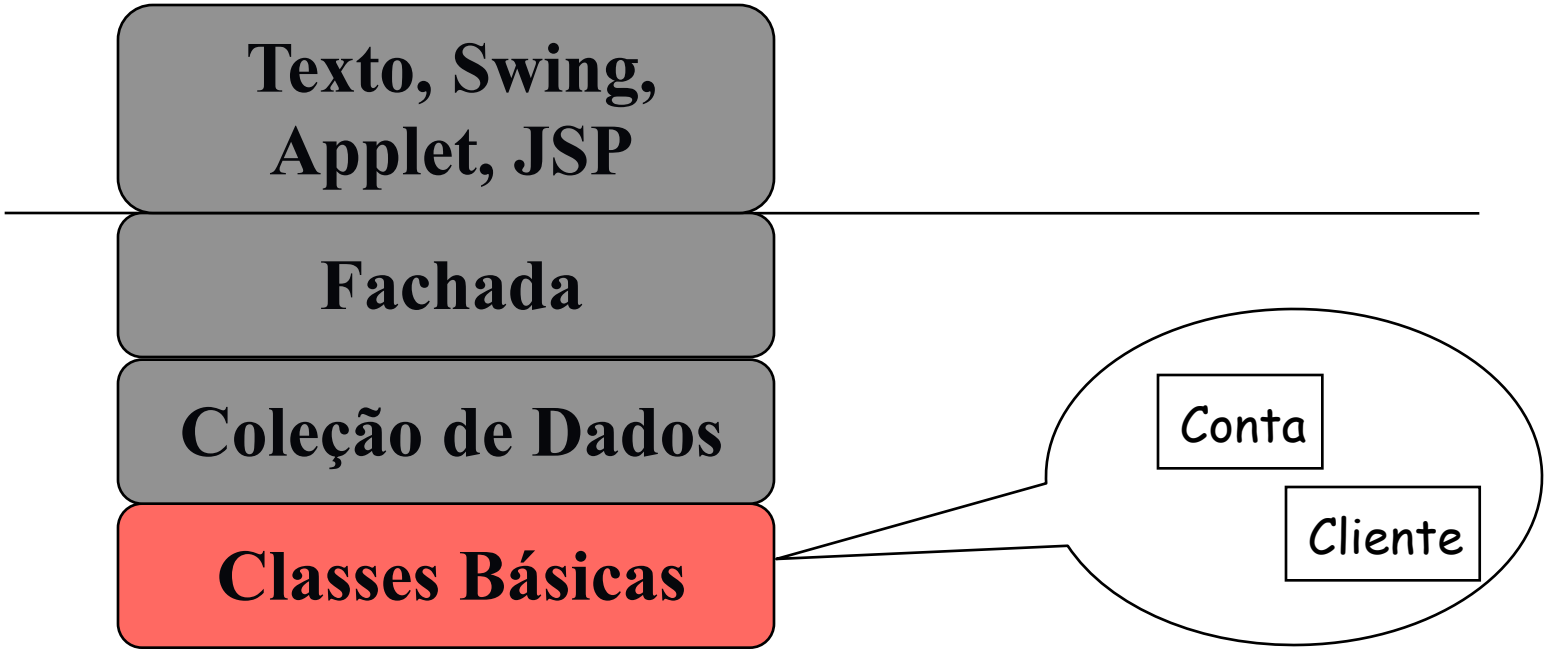
Definição

- Camadas definem um conjunto de classes com preocupações em comum
- Separação de Código
  - GUI (Interface Gráfica)
    - Apresentação da aplicação
    - Entrada e saída de dados
  - Negócio
    - Código inerente à aplicação sendo desenvolvida
  - Dados
    - Código de acesso e manipulação de dados, exclusivamente

Benefícios do Sistema em Camadas

- Modularidade
  - Dividir para conquistar
  - Separação de conceitos
  - Reusabilidade e facilidade de mudança
- Mudanças em uma camada não afeta às outras
  - Funcionalidade plug and play
  - Várias GUIs para a mesma aplicação
  - Vários mecanismos de armazenamento para o mesmo sistema

Projeto em Camadas



Classes Básicas

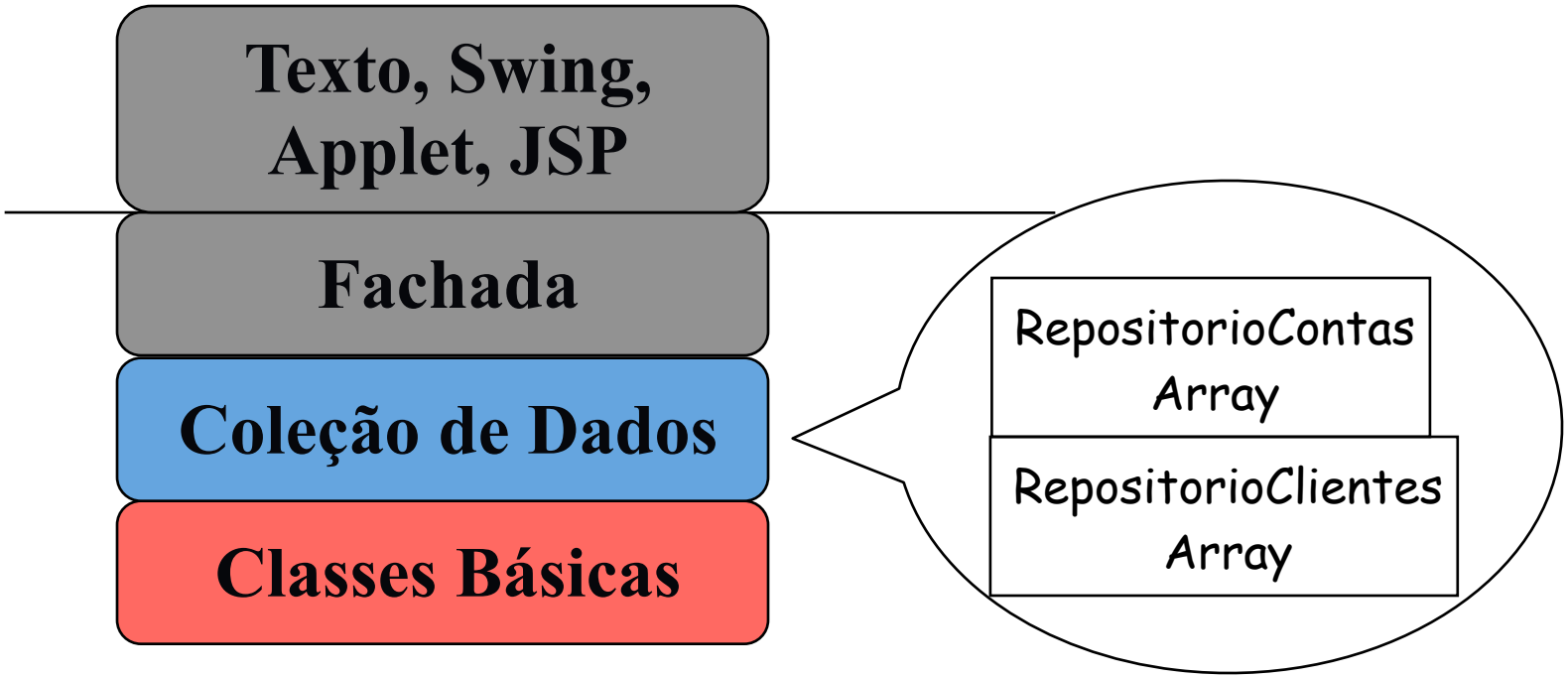
- Composta pelas classes elementares da aplicação
- Implementam serviços relacionados a um (1) objeto
- Possuem os métodos de acesso além de alguns serviços que atuam sobre os seus atributos
- Exemplos: Conta, Agencia, Cliente, Empregado, Venda, Aluno, Disciplina

Classes Básicas

```
public class Conta{
   private String numero;
   ...
   public Conta(String num){
       this.numero = num;
   }
   public void setNumero(String num){
       this.numero = num;
   }
   public void creditar(double valor){
       this.saldo += valor;
   }
```

Métodos de acesso poderiam ser usados para tudo, mas comportamentos complexos em um objeto são encapsulados na classe básica

Projeto em Camadas



Coleção de Dados (Repositório)

- Responsável por armazenar e recuperar objetos de UMA classe básica
- Definem estrutura para armazenar estes objetos: lista, array, etc...
- Preocupa-se APENAS com detalhes de armazenamento e recuperação dos objetos: inserir, remover, procurar
- Nome padrão: RepositorioClasseBasicaArmazenamento
  - Exemplo: RepositorioContasArray

Coleção de Dados

Estrutura escolhida, array!

```
public class RepositorioContasArray{

    private Conta[] contas;
    private int proximaLivre;

    public RepositorioContasArray(int tamanho) {
        contas = new Conta[tamanho];
        proximaLivre = 0;
    }
    public void inserir(Conta c){
        contas[proximaLivre] = c;
        proximaLivre++;
    }
```

Tamanho passado ao construtor

Não testa se já existe a conta??

Coleção de Dados

```
public Conta procurar(String numero){
    for (int i=0;i < proximaLivre;i++){
       if (contas[i].getNumero().equals(numero))
             return contas[i];
    }
    return null;
}
```

Este código vai ficar repetitivo para remover, verificar existência...

Coleção de Dados

```
private int procurarIndice(String numero){
    for (int i=0;i < proximaLivre;i++){
      if (contas[i].getNumero().equals(numero))
                  return i;
    }
    return -1;
}

public Conta procurar(String numero){
    int indice = this.procurarIndice(numero);
    if (indice != -1)
       return contas[indice];
    else return null;
}
```

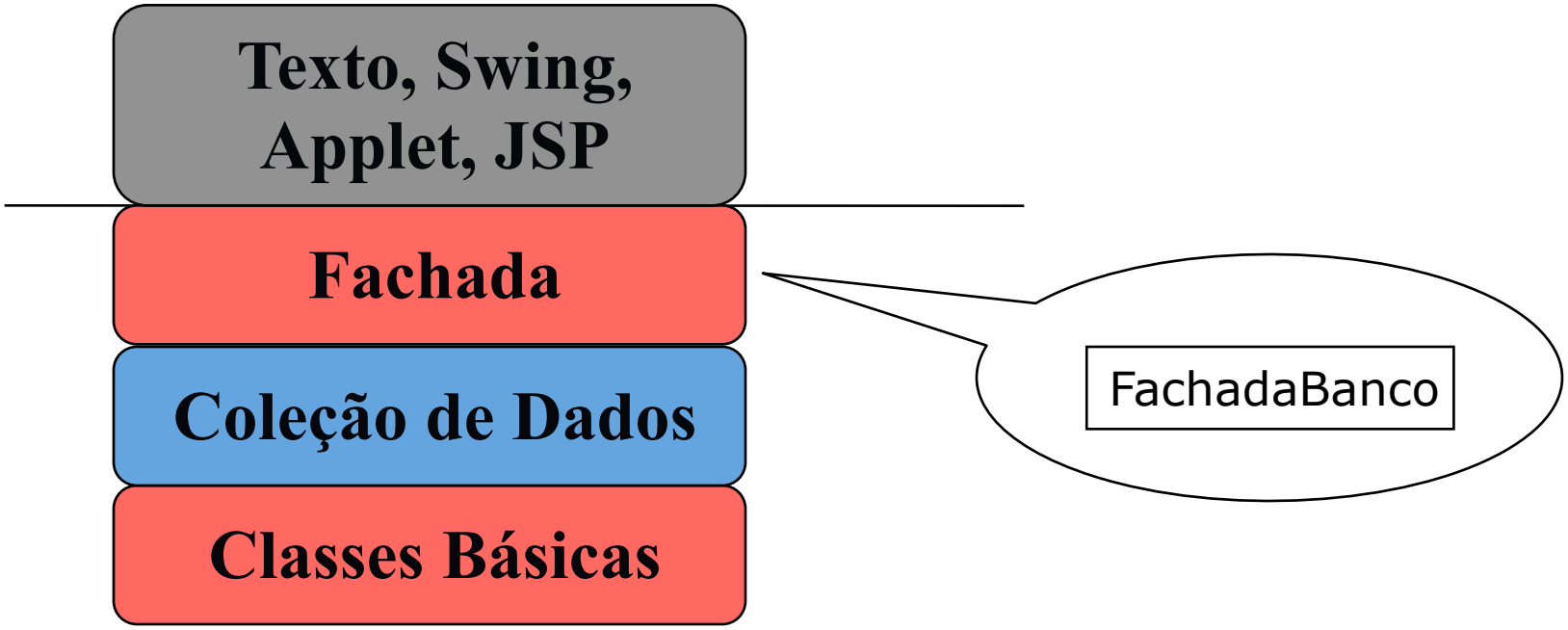
Método privado

Reutilização chamando o método

Coleção de Dados

```
public void atualizar(Conta c){
   int indice = this.procurarIndice(numero);
   if (indice > -1){
       contas[indice] = c;
   }
}

public boolean existe(String numero){
   if (this.procurarIndice(numero) == -1)
       return false
   else
       return true;
}
```

Projeto em Camadas



Fachada

- Realiza as operações e testes de negócio
  - Ex: teste 'não é possível inserir duas contas com o mesmo número'
  - Ex: 'creditar um valor em uma conta da coleção dado o número da mesma'
- Realiza operações críticas de negócio envolvendo diferentes subsistemas (críticas de restrição de integridade).
  - Ex: não é possível inserir contas para clientes que ainda não foram cadastrados
- Utiliza os serviços do repositório para lhe dar suporte ao armazenamento dos objetos

Fachada

- Reúne todas as funcionalidades (métodos) de todas as coleções de dados combinadas
  - Centralização do acesso aos serviços do sistema
- Inicializa todos os "subsistemas"
- Pergunta?
  - Suponha que o sistema agora possui uma outra regra de negócio que não permite que contas com saldo inicial menor que 50.0 reais sejam cadastradas. Onde implementar essa nova regra?

Fachada

Inicialização do sistema

```
public class FachadaBanco{
 private RepositorioContasArray repContas;
 private RepositorioClientesArray repClientes;

 public FachadaBanco(){
   repContas = new RepositorioContasArray(100);
   repClientes = new RepositorioClientesArray(100);
 }

 public void cadastrarCliente (Cliente cli){
   boolean achou = repClientes.existe(cli.getCpf());
   if (!achou){
      repClientes.inserir(cli);
   }
 }
```

Regra de negócio realizada utilizando o serviço da camada abaixo

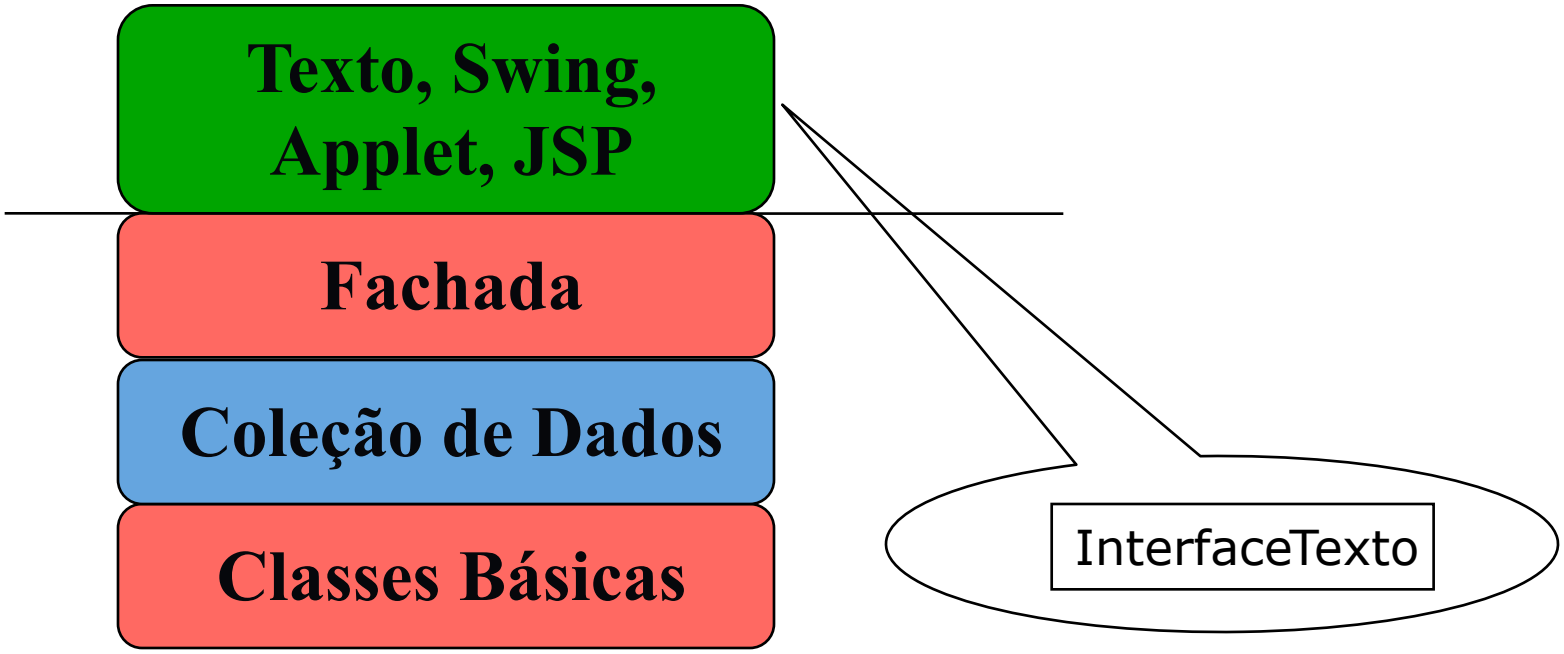
Fachada

```
public void cadastrarConta (Conta c) {
  boolean achou = repContas.existe(c.getNumero());
  if (!achou){
     boolean existeCliente =

           repClientes.existe(c.getTitular().getCpf());
     if (existeCliente)
           repContas.inserir(c);
  }
}

public void debitar (String num, double valor) {
     Conta c = repContas.procurar(num);
     c.debitar(valor);
}

...//como seria o transferir?
```

Regra de negócio envolvendo vários repositórios

Agora passamos o número (serviço do sistema)

Projeto em Camadas



Interface "Gráfica" (GUI)

- Classe que faz a intermediação do usuário com o sistema
- Prepara entrada do usuário
- Possui uma referência ao objeto Fachada, para pedir serviços que o usuário precisa
- Classe que vai ser "executada"
  - Possui um método main()

Interface "Gráfica"

```
public class InterfaceTexto{

 public static void main(String [] args){
   FachadaBanco f = new FachadaBanco();
   ...
   Cliente titular = f.procurarCliente(cpf);
   Conta novaConta = new Conta(numero,0.0,titular);
   f.cadastrarConta(novaConta);
   ...
  }
}
```

Aula Prática
Camadas

Objetivos da prática

- Organizar um sistema bancário em camadas
- Implementar lógica de negócio e armazenamento de dados para uma interface texto de um usuário do banco

Atividades

- Implementar Repositorios de array
  - Para armazenar contas e clientes
  - Implementar os métodos, com a ajuda do método privado procurarIndice()
  - Inserir, atualizar, procurar, existe
    - Ignorar possíveis erros

Atividades

- Implementar FachadaBanco
  - Serviços indicados, utilizam chamadas aos repositórios
  - Construtor deve iniciar todos os repositórios
  - Ignorar possíveis erros
    - Apenas evitar NullPointerException
- Checar se todos os erros de compilação foram resolvidos
  - Incrementar a interface texto com entradas de usuário
    - Lista de funcionalidades como no exercício passado
    - Oferecer opções que a Fachada oferece

Um outro possível modelo em Camadas

GUI →