Programação Orientada a Objetos
Padrões de Projeto

Rodrigo Bonifácio

25 de junho de 2013

A autenticação do aluno dependia do estado:

- Estudantes sem acesso
- Estudantes sem ativação do código de acesso
- Estudantes com permissão de acessar o sistema
- Estudantes com acesso bloqueado

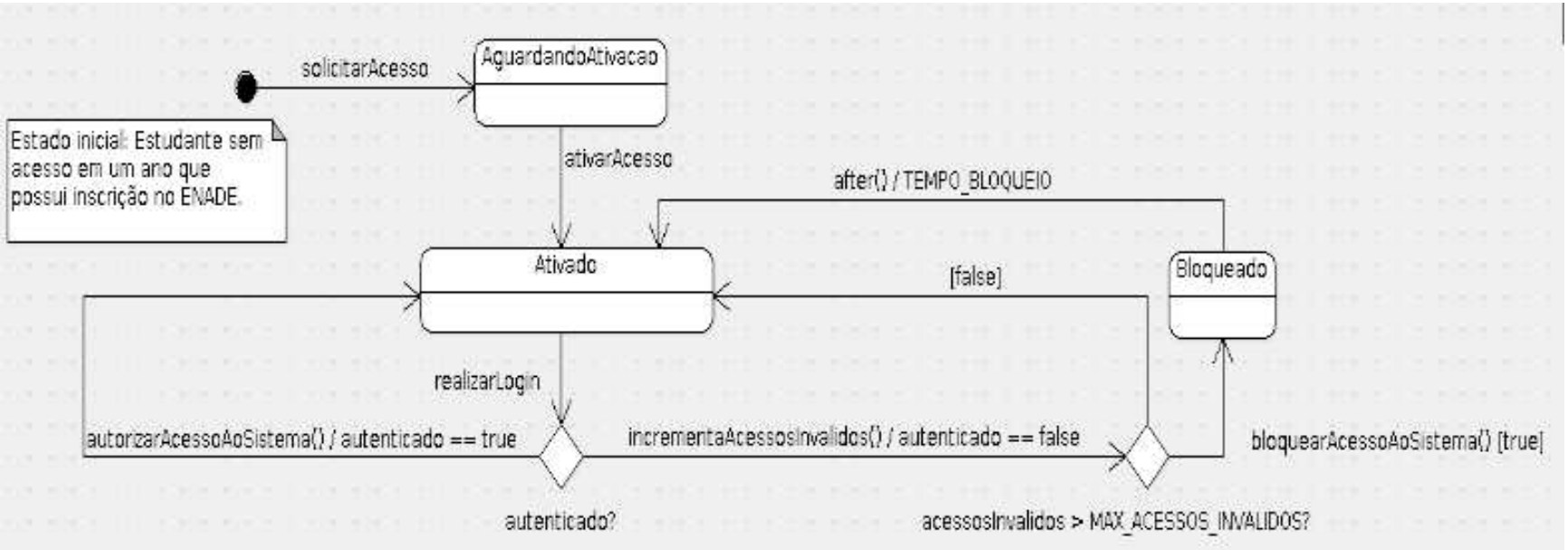
Projeto ENADE

A autenticação do aluno dependia do estado:

- Estudantes sem acesso
- Estudantes sem ativação do código de acesso
- Estudantes com permissão de acessar o sistema
- Estudantes com acesso bloqueado

Diferente operações podem ser realizadas.

- Solicitar acesso, ativar acesso
- Acessar o sistema (login)
- Alterar senha, ...


solicitarAcesso
AguardandoAtivacao
Estado inicial: Estudante sem acesso em um ano que possui inscrição no ENADE.
ativarAcesso
after() / TEMPO_BLOQUEIO
Ativado
[false]
Bloqueado
realizarLogin
autorizarAcessoAoSistema() / autenticado == true
incrementaAcessosInvalidos() / autenticado == false
bloquearAcessoAoSistema() [true]
autenticado?
acessosInvalidos > MAX_ACESSOS_INVALIDOS?

Como implementar essa “máquina de estados” de forma flexível?

Como implementar essa "máquina de estados" de forma flexível?

- Possibilidade de adicionar novos estados de forma flexível
- A lógica para checar o estado da autenticação do aluno deve estar modularizada, sem espalhamento.

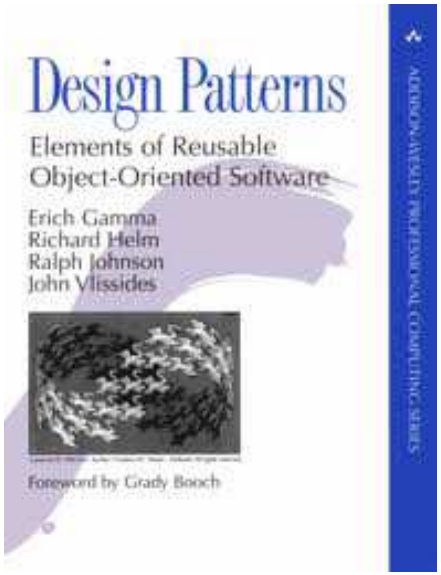
Como implementar essa "máquina de estados" de forma flexível?

- Possibilidade de adicionar novos estados de forma flexível
- A lógica para checar o estado da autenticação do aluno deve estar modularizada, sem espalhamento. Isso é uma característica de um bom design OO.

Sem termos resolvido algo parecido, essa é uma tarefa bem complicada . . .

a menos que alguém já tenha resolvido esse problema de forma elegante, e tal solução tenha sido bem documentada.

a menos que alguém já tenha resolvido esse problema de forma elegante, e tal solução tenha sido bem documentada. Como um catálogo de padrões



Livro da GoF

Catálogo de padrões de projeto organizados como:

- **Criacionais:** criação de objetos
- **Comportamentais:** implementação de algoritmos
- **Estruturais:** organização das classes de um subsistema

Livro da GoF

Catálogo de padrões de projeto organizados como:

- **Criacionais:** criação de objetos
- **Comportamentais:** implementação de algoritmos
- **Estruturais:** organização das classes de um subsistema

Visando flexibilidade e reúso do design OO

Voltando ao exemplo, vamos consultar a bíblia do design OO

Voltando ao exemplo, vamos consultar a bíblia do design OO parece que o padrão State é uma boa alternativa

Padrão State: Propósito

Propósito

Permite um objeto alterar o seu comportamento quando seu estado é modificado.

Padrão State: Propósito

Propósito

Permite um objeto alterar o seu comportamento quando seu estado é modificado.

- Excelente isso é exatamento o que precisamos!

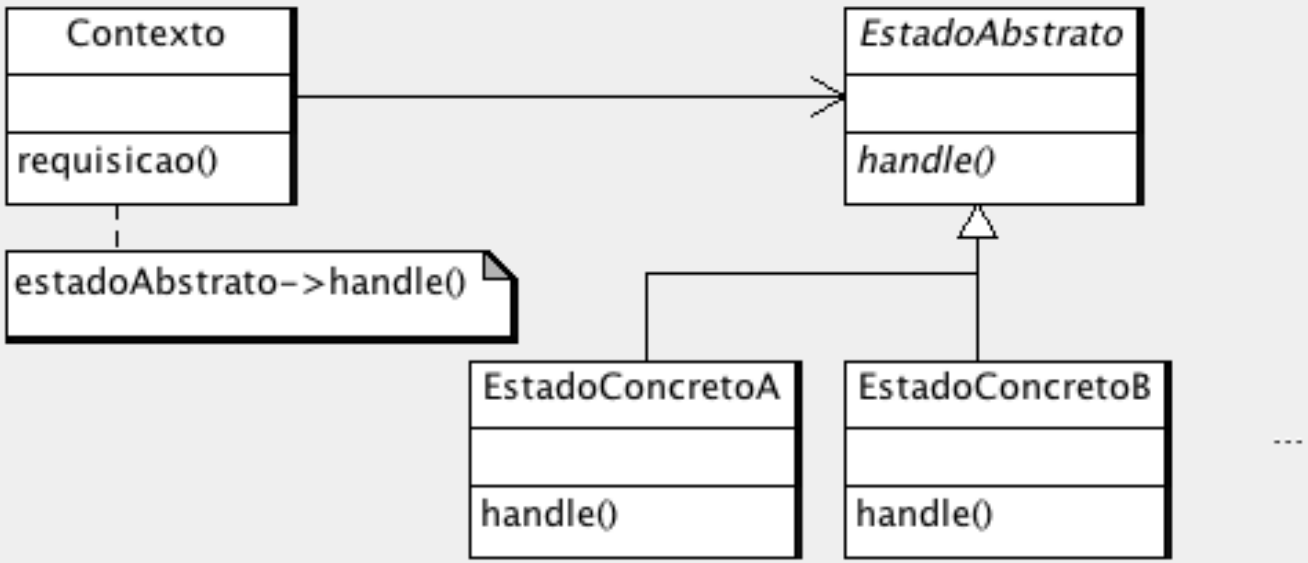
Padrão State: Motivação

As operações relacionadas com a autenticação dos estudantes dependem do estado da própria autenticação (aguardando ativação, bloqueada, . . .)

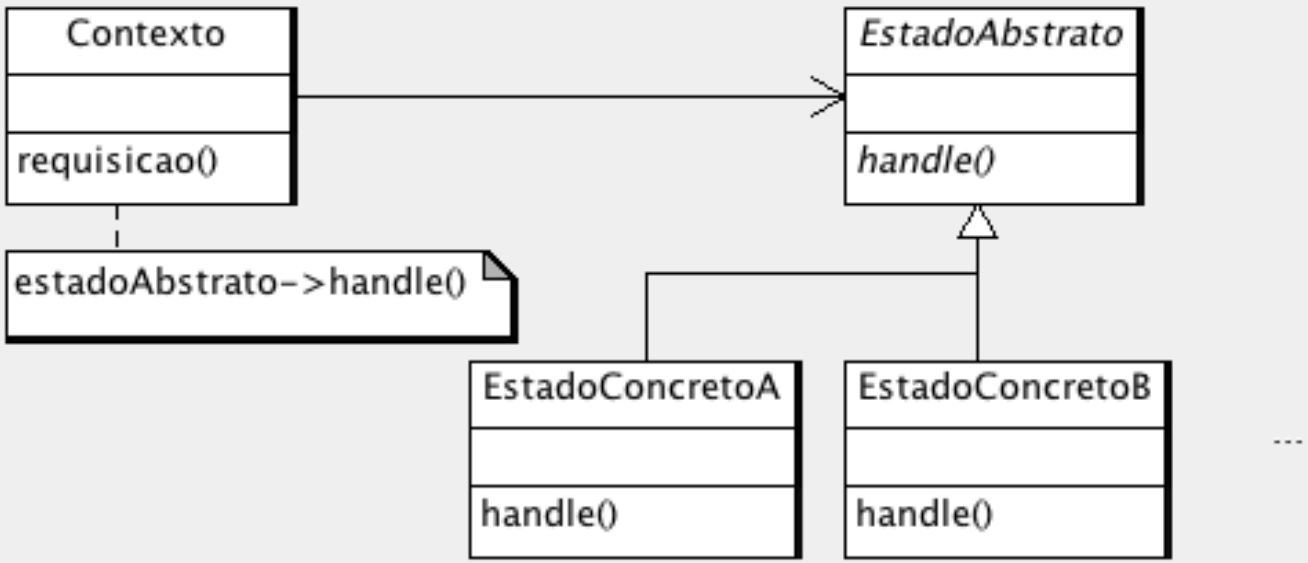
Aplicabilidade

- O comportamento do objeto depende do seu estado, e deve ser possível alterar tal comportamento durante tempo de execução.
- Operações têm muitas sentenças condicionais que referenciam o estado do objeto. Tal estado é usualmente representado por constantes enumeradas.

Padrão State: Estrutura e Participantes



Padrão State: Estrutura e Participantes



Padrão State: Colaborações

- *Contexto*, que serve como interface para os clientes, delega as requisições para o *estado concreto* atual do objeto.
- O *contexto* pode passar a si mesmo como argumento para as implementações do método *handle()* .
- O *contexto* ou os *estados concretos* decidem sobre quando uma transição deve ocorrrer.

Padrão State: Conseqüências

- Localização do comportamento que é específico de um estado
- Torna a transição de estados explícita, com uma classe implementado cada estado válido (em vez do uso de constantes)
- Aumenta a quantidade de classes da aplicação, o que de certa forma causa impacto na produtividade da equipe— mais classes a serem revisadas, testadas, ...

Padrão State: Decisões de Implementação

- Quem define a transição de estado?
- Criação e destruição de objetos

Padrão State: Decisões de Implementação

- Quem define a transição de estado?
- Criação e destruição de objetos
- Alternativa: implementação baseada em tabelas

Padrão State: Código de Exemplo:

AtivadoSenhaBloqueada.java:1 <No selected symbol>

```
package br.gov.inep.enadeies.workflow.cadastroEstudante;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

import br.gov.inep.enadeies.dao.AlunoDao;
import br.gov.inep.enadeies.entity.enade.Aluno;
import br.gov.inep.enadeies.entity.enade.LoginAluno;
import br.gov.inep.enadeies.exception.AutenticacaoAlunoException;
import br.gov.inep.enadeies.util.Constantes;
import br.gov.inep.enadeies.util.ResourceBundleUtil;
import br.gov.inep.enadeies.util.Utilitario;

/**
 * Implementa a transicao de estado em que o
 * acesso do estudante está ativado, mas com a senha
 * bloqueada.
 *
 * @author rbonifacio
 *
 */
public class AtivadoSenhaBloqueada extends Ativado {
    protected AtivadoSenhaBloqueada(LoginAluno acesso, AlunoDao alunoDao) {
        super(acesso, alunoDao);
    }

    @Override
    public Aluno autenticar(String senha) throws AutenticacaoAlunoException {
        Date dataAtual = new Date(System.currentTimeMillis());
        Date dataBloqueio = acesso.getDataAtualizacao();


        if(Utilitario.diferencaEmHoras(dataBloqueio, dataAtual) >= Constantes.TEMPO_BLOQUEIO_SENHA_ESTUDANTE) {
            acesso.desbloquear();
            return super.autenticar(senha);
        }

        throw new AutenticacaoAlunoException(ResourceBundleUtil.getMessage("estudante.login.cadastro.bloqueado", new String [] {
    }
}
```

Padrão State: Uso Comum e Padrões Relacionados

- Protocolos
- Games
- Processos de negócio, aplicações corporativas, ...

Padrão State: Uso Comum e Padrões Relacionados

- Protocolos
- Games
- Processos de negócio, aplicações corporativas, ...

Padrões relacionados

- Singleton
- Flyweight