MC302
Primeiro semestre de 2017

Laboratório 8

Professores: Esther Colombini (esther@ic.unicamp.br) e Fábio Luiz Usberti (fusberti@ic.unicamp.br)
PEDs: (Turmas ABCD) Elisangela Santos (ra149781@students.ic.unicamp.br), Lucas Faloni (lucasfaloni@gmail.com), Lucas David (lucasolivdavid@gmail.com), Wellington Moura (wellington.tylon@hotmail.com)
PEDs (Turmas EF) Natanael Ramos (naelr8@gmail.com), Rafael Arakaki (rafaelkendyarakaki@gmail.com)
PAD: (Turmas ABCD) Igor Torrente (igortorrente@hotmail.com)
PAD: (Turmas EF) Bleno Claus (blenoclaus@gmail.com)

---

1 Objetivo

O objetivo deste laboratório consiste em praticar os conceitos de classes abstratas, interfaces e classes internas.

2 Atividade

Iremos utilizar o código do laboratório 7, com o intuito de adapta-lo, buscando deixa-lo mais coeso e menos acoplado.

1. Crie um novo projeto Java com o nome Lab8.
2. Reutilize o projeto do Lab7, para isso basta copiar e colocar as estrutura interna de um projeto no outro.

2.1 Classes Abstratas

`https://www.caelum.com.br/apostila-java-orientacao-objetos/classes-abstratas/`

2.2 Interfaces

`https://www.caelum.com.br/apostila-java-orientacao-objetos/interfaces/`

2.3 Classes Internas

`http://www.java2s.com/Tutorial/Java/0100__Class-Definition/0200__Nested-Classes.htm`

2.4 Novos Pacotes

Crie os pacotes: base.controle; base.service; base.serviceImpl.

3 Questões

3.1 Parte 1

Responda as seguintes questões em um **arquivo texto** e submeta junto ao código no Moodle:

- Tendo em vista que não faz sentido que as classes Carta e Magia sejam instanciadas. Como podemos garantir que ambas nunca serão instanciadas?
- Como podemos obrigar que as classes que herdam da classe Carta implementem o método usar(Carta)? Faça as alterações no seu código. Quais as consequências dessa alteração?

3.2 Novas classes

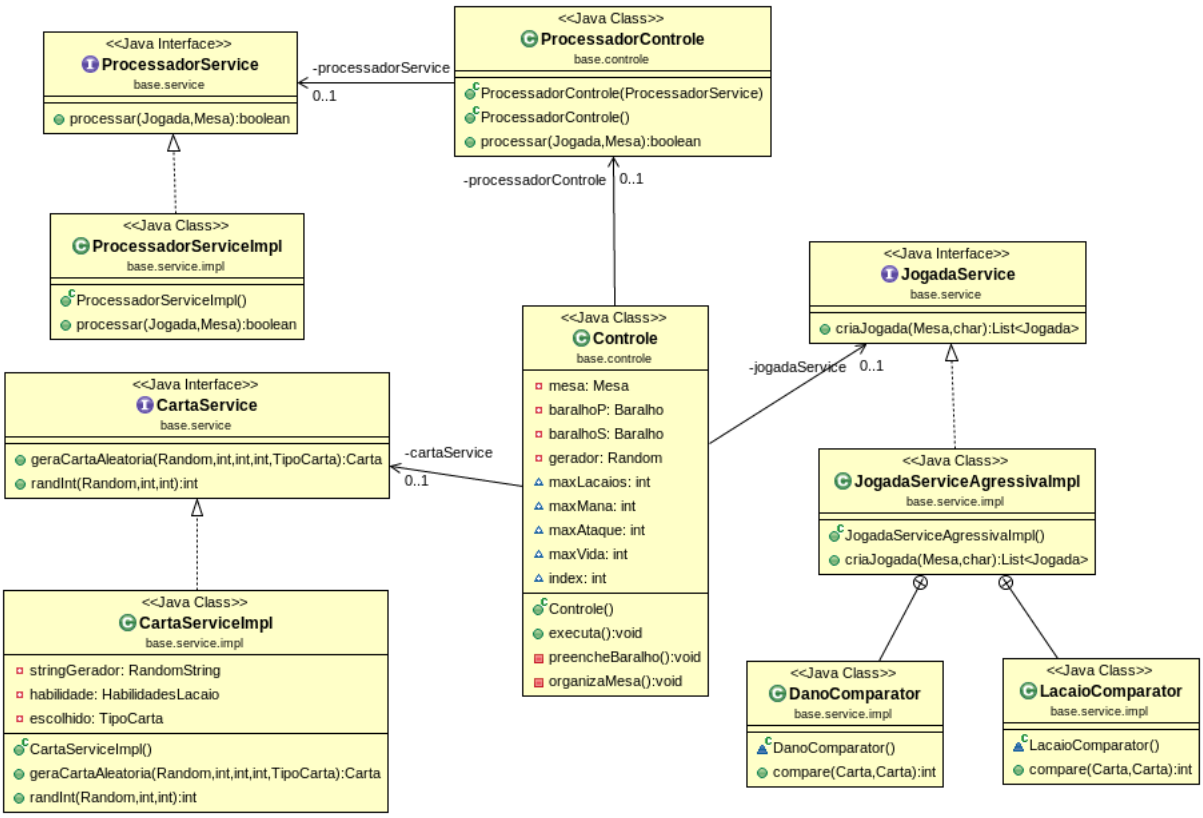

Figura 1: Organização de classes

- Vamos criar a estrutura de classes/interfaces apresentadas na Figura **??**.
- A classe Controle deve possuir a lógica de execução, que estava contida na Main. Veja a subseção **??** uma implementação dessta classe.
- A Main apenas instanciará a classe Controle e chamará o método executar() : void.

- Deve-se criar uma interface ProcessadorService, refatore o nome da classe ProcessadorJogada para ProcessadorServiceImpl e realoque-a para o pacote base.service.impl, como é ilustrado na Figura **??**, faça com que ela implemente a interface, como sugerido.
- A interface CartaService será usada para retirar a lógica, de gerar cartas aleatórias, da classe Util. A classe Util necessita apenas das constantes, os métodos buffar não são requeridos em outras classes.
- A interface JogadaService será utilizada para criar novas jogadas.
- As duas classes DanoComparator e LacaioComparator são classes internas de JogadaServiceAgressivaImpl, ambas implementam a interface Comparator<Carta>. Ambas são utilizadas na implementação de uma estratégia agressiva.

3.3 Classe Controle

```java
package base.controle;

import static util.Util.MAO_INI;
import static util.Util.MAX_CARDS;

import java.util.List;
import java.util.Random;

import base.Baralho;
import base.Jogada;
import base.Mesa;
import base.cartas.TipoCarta;
import base.service.CartaService;
import base.service.JogadaService;
import base.service.impl.CartaServiceImpl;
import base.service.impl.JogadaServiceAgressivaImpl;
import base.service.impl.ProcessadorServiceImpl;

public class Controle {

    private Mesa mesa;
    private Baralho baralhoP;
    private Baralho baralhoS;
    private Random gerador;
    private ProcessadorControle processadorControle;
    private CartaService cartaService;
    private JogadaService jogadaService;

    int maxLacaios = 10;
    int maxMana = 2;
    int maxAtaque = 6;
    int maxVida = 6;
    int index;

    public Controle() {
        this.baralhoP = new Baralho();
        this.baralhoS = new Baralho();
        this.mesa = new Mesa();
        gerador = new Random();
        cartaService = new CartaServiceImpl();
        jogadaService = new JogadaServiceAgressivaImpl();
    }

    public void executa() {
```

```
45
46              preencheBaralho();
47
48              organizaMesa();
49
50              processadorControle = new ProcessadorControle(new ProcessadorServiceImpl())
                    ;
51
52              List<Jogada> jogadas = jogadaService.criaJogada(mesa, 'P');
53              for (Jogada jogada : jogadas) {
54                  if (processadorControle.processar(jogada, mesa)) {
55                      System.out.println("###### " + jogada.getAutor() + " venceu!");
56                      break;
57                  }
58              }
59
60          }
61
62          private void preencheBaralho() {
63              baralhoP.preencheAleatorio(gerador, MAX_CARDS, maxMana, maxAtaque, maxVida)
                    ;
64              baralhoS.preencheAleatorio(gerador, MAX_CARDS, maxMana, maxAtaque, maxVida)
                    ;
65          }
66
67          private void organizaMesa() {
68              for (int i = 0; i < maxLacaios; i++) {
69                  mesa.getLacaiosP()
70                          .add(cartaService.geraCartaAleatoria(gerador, maxMana,
                                maxAtaque, maxVida, TipoCarta.LACAIO));
71                  mesa.getLacaiosS()
72                          .add(cartaService.geraCartaAleatoria(gerador, maxMana,
                                maxAtaque, maxVida, TipoCarta.LACAIO));
73              }
74
75              for (int i = 0; i < MAO_INI; i++) {
76                  mesa.getMaoP().add(baralhoP.comprar());
77                  mesa.getMaoS().add(baralhoS.comprar());
78              }
79
80              mesa.getMaoS().add(baralhoS.comprar());
81          }
82      }
```

3.4 Parte 2

Responda as seguintes questões em um **arquivo texto** e submeta junto ao código no Moodle:

- Qual a vantagem dos controladores conhecerem apenas as interfaces dos serviços?
- Caso queira criar uma nova estratégia de jogada, além da agressiva, como você procederia? Como o uso de Interfaces garante que sua nova estratégia será executada no controlador?
- Com relação às classes internas, utilizadas em JogadaServiceAgressivaImpl, qual a vantagem desse tipo de abordagem?

4 Submissão

Para submeter a atividade, utilize o *Moodle* (`https://www.ggte.unicamp.br/ea`). Crie um arquivo texto com as respostas para cada item da seção tarefas e as saídas geradas pelo código. Compacte o código-fonte contido no diretório **src** juntamente com arquivo de respostas no formato .zip ou similar e nomeie-o **Lab8-000000.zip**, trocando '000000' pelo seu número de RA. Submeta o arquivo na seção correspondente para esse laboratório no *Moodle* da disciplina MC302.

Certifique-se de entregar um código compilável, incluindo todas as subpastas dos pacotes em prol de facilitar a correção.

Datas de entrega

- Dia **23/05** Turma **ABCD** até às 23:55h
- Dia **19/05** Turma **EF** até às 23:55h