

MC302
Primeiro semestre de 2017

Laboratório 3

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 na prática de módulos, métodos estáticos, variáveis estáticas e finais, arrays e a classe Random da biblioteca padrão do Java.

2 Atividade

Continuaremos trabalhando em classes baseadas no jogo de cartas de computador chamado Hearthstone¹. Nesta atividade o principal foco será a construção das classes **Baralho** e **Util**. Crie um projeto chamado Lab3. No projeto crie dois pacotes com o seguinte formato: **com.seuPrimeiroNome.util** e **com.seuPrimeiroNome.base**. Cole a classe do **CartaLacaio.java** dentro do projeto no pacote base.

3 Classe Baralho

Crie a classe Baralho no pacote **base** com os seguintes atributos:

- vetorCartas (array de objetos da classe **CartaLacaio**)
- nCartas (número inteiro)
- gerador (atributo **estático** da classe Random²)

Um exemplo da classe é dado a seguir.

```
1 public class Baralho {
2     CartaLacaio[] vetorCartas;
3     int nCartas;
4     static Random gerador = new Random();
5
6 }
```

Baralho.java

¹<http://us.battle.net/hearthstone/pt>

²É necessário importar a classe Random da biblioteca padrão do Java, para isso use: import java.util.Random (Existe o atalho Ctrl+Shift+O atalho no eclipse para imports faltantes.)

Crie um construtor para a classe Baralho da seguinte forma. Repare que o gerador é uma variável estática inicializada já em sua própria declaração.

```
1 public Baralho () {
2     vetorCartas = new CartaLacaio [10];
3     nCartas = 0;
4 }
```

Construtor_Baralho

A classe Baralho possuirá dois métodos, um que adiciona e outro que retira cartas: adicionarCarta() e comprarCarta(). Como convenção, iremos adicionar e remover cartas sempre do final do array vetorCartas. O código desses métodos é dado a seguir:

```
1 public void adicionarCarta (CartaLacaio card){
2     vetorCartas [nCartas] = card;
3     nCartas++;
4 }
5
6 public CartaLacaio comprarCarta () {
7     nCartas ---;
8     return vetorCartas [nCartas];
9 }
```

Adicionar_Comprar

A classe Baralho também possuirá um método para embaralhar as cartas. Para isso, utilize o algoritmo descrito a seguir: tendo como entrada um baralho (vetor de cartas), itere sobre seus índices de 0 a $n - 1$. Para cada posição i , escolha aleatoriamente uma posição j no intervalo fechado $[0, i]$ e troque as cartas das posições i e j . Esse algoritmo já encontra-se implementado, conforme segue:

```
1 public void embaralhar () {
2     int i, j;
3
4     for (i = 1; i < nCartas; i++){
5         j = gerador.nextInt(i+1); //Sorteia um numero dentre [0,i]
6         if (j != i){
7             CartaLacaio a = vetorCartas [i];
8             CartaLacaio b = vetorCartas [j];
9             vetorCartas [i] = b;
10            vetorCartas [j] = a;
11        }
12    }
13    //Comandos para imprimir as cartas em ordem reversa aqui
14 }
```

Embaralhar

A utilização do atributo gerador para sortear um número dentro de um intervalo não é a única utilidade da classe Random, para mais informações consulte a documentação³.

Após embaralhar as cartas, o método embaralhar deve imprimir as cartas na ordem reversa do array (de $n - 1$ até 0 no array). Em nossa convenção, essa será a ordem em que o usuário irá comprar as cartas. Para imprimir as cartas, utilize System.out.println(<objeto carta>). Esta chamada invocará o método toString() das cartas implementado no Lab1.

³<http://docs.oracle.com/javase/7/docs/api/java/util/Random.html>

4 Classe Util

Crie uma classe chamada Util dentro do pacote **util**. Nessa classe implemente dois novos métodos **buffar** da classe lacaio (os métodos devem incrementar/aumentar a vidaAtual e a vidaMax). Esses métodos devem ser estáticos e substituirão os métodos antigos (apague os antigos na classe CartaLacaio). As assinaturas dos métodos devem ser como segue:

```
1 public static void buffar(CartaLacaio lac, int a);  
2 public static void buffar(CartaLacaio lac, int a, int v);  
3 private static void alteraNomeFortalecido(CartaLacaio lac);
```

Buffar

5 Collection

A linguagem Java disponibiliza para seus programadores um conjunto de classes e interfaces chamado Collection⁴.

Hierarquias de interfaces e classes

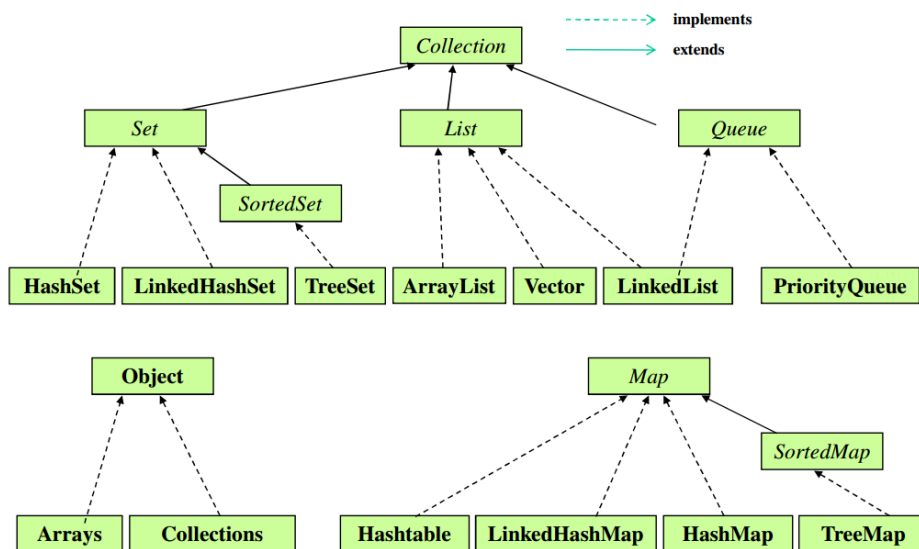


Figura 1: Hierarquia Collection.

Existem quatro tipos de coleções (interfaces) básicas como mostra a Figura 1:

⁴<https://docs.oracle.com/javase/7/docs/api/java/util/Collection.html>

- List: lista de objetos. Pode ter elementos repetidos
- Set: conjunto de objetos. Não pode ter elementos repetidos
- Queue: Introduzida a partir do Java 5. Ideal para implementação de Filas.
- Map: grupo de objetos que possuem um identificador (id) ou chave associado com cada objeto.

As palavras-chave *implements* e *extends* serão abordadas futuramente na disciplina.

Uma coleção é um grupo de objetos e precisamos saber qual coleção é mais adequada para um determinado requisito. As operações básicas que geralmente usamos com coleções são:

- Adicionar objetos
- Remover objetos
- Verifica se um objeto está na coleção
- Recuperar um objeto
- Iterar através da coleção acessando todos os objetos

As operações podem ter métodos sobrecarregados dando liberdade ao programador para utilizar o que mais lhe beneficia. Para saber como realizar as operações consulte a documentação da coleção que estiver usando.

Crie uma nova classe no pacote **base** chamada BaralhoArrayList. Nessa classe iremos utilizar ArrayList⁵ ao invés de array. O único atributo da classe será um ArrayList de objetos CartaLacaio. Veja mais abaixo como é a classe e seu construtor.

```

1 public class BaralhoArrayList {
2     private ArrayList<CartaLacaio> vetorCartas;
3
4     public BaralhoArrayList() {
5         vetorCartas = new ArrayList<CartaLacaio>();
6     }
7 }

```

BaralhoArrayList

O primeiro baralho foi limitado a 10 cartas com intuito de facilitar os testes deste laboratório. Contudo, um baralho futuramente será composto por 30 cartas. Crie uma variável estática e final inteira na classe Util com o valor 30.

```

1 public static final int MAX_CARDS = 30;

```

Constante

Programa os métodos adicionarCarta, comprarCarta e embaralhar. Verifique no método adicionarCarta se o baralho não chegou no limite (30 cartas) antes de adicionar uma nova. No método embaralhar não utilize mais o algoritmo anterior, embaralhe os objetos do ArrayList com o método **shuffle** da classe Collections⁶. Essa classe contém métodos estáticos que operam sobre coleções. Para imprimir as cartas em ordem reversa, verifique se a classe Collections possui algum método que possa te ajudar.

⁵<https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html>

⁶<https://docs.oracle.com/javase/7/docs/api/java/util/Collections.html>

6 Classe Main

Crie uma nova classe Main e escolha a opção para gerar automaticamente o método main⁷.

Na função main instancie três objetos arbitrários da classe CartaLacaio e um novo objeto de Baralho. Adicione as cartas no baralho e chame o método embaralhar. Verifique se as cartas estão sendo embaralhadas (pode ser necessário rodar algumas vezes para que ocorra alguma mudança na ordem). Repita esta tarefa mas agora utilizando a classe BaralhoArrayList.

Somente para o objeto BaralhoArrayList, execute pelo menos uma vez o método comprarCarta e mostre a carta que foi comprada.

Também na função main, utilize os dois métodos buffar da classe Util para fortalecer duas cartas de sua escolha. Imprima o estado das cartas fortalecidas.

7 Tarefas

Responda as questões a seguir sucintamente em um arquivo texto que deverá ser submetido juntamente com o código:

1. Considerando a implementação atual do método construtor de Baralho, o que acontece se adicionarmos mais de 10 cartas? Que tipo de erro o Java acusará? (Se não souber, teste!)
2. No método comprarCarta de Baralho, o que acontece se chamarmos este método com o baralho vazio? Que tipo de erro Java acusará? Por que este problema ocorre? (Se não souber, teste!)
3. Na declaração do atributo gerador, por que este atributo pode ser estático? Qual a diferença de escopo de um atributo “estático” para um “não-estático”?
4. Por que o atributo **gerador** não é inicializado no construtor da classe Baralho? Em nosso programa, quantas vezes o comando new Random() será executado?
5. Qual o benefício de criar a classe Util e utilizar os métodos estáticos e a variável final estática?
6. Quais os benefícios de implementar a classe Baralho com ArrayList e não com vetor?

Para pensar: Observe que neste momento nosso baralho aceita somente objetos do tipo CartaLacaio. Como poderíamos fazer uma classe Baralho que receba objetos do tipo CartaLacaio e CartaMagia?

8 Submissão

Para submeter a atividade utilize o Moodle (<https://www.ggte.unicamp.br/ea>). Crie um arquivo .pdf com as respostas para cada item da seção tarefas e as saídas geradas pelo código. Salve todos os arquivos desta atividade em um arquivo compactado e nomeie-o **Lab3-000000.zip** trocando '000000' pelo seu número de RA. Submeta o arquivo na seção correspondente para esse laboratório no moodle da disciplina MC302.

Datas de entrega

- Dia **03/04** Turma **ABCD** até às 23:55h
- Dia **31/03** Turma **EF** até às 23:55h

⁷Dica: Ctrl+espaço com o cursor na classe exibe opções de códigos que podem ser gerados automáticos com pré-visualização, a opção "main" também gera o método estático main.