MC322
Segundo semestre de 2018

Laboratório 3

Professor: Fábio Luiz Usberti (fusberti@ic.unicamp.br)
PED Luis Henrique Pauleti Mendes (luishpmendes@gmail.com)
PAD: Pedro Barros Bastos (p204481@dac.unicamp.br)

---

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 chamado Svoyi Koziri[1]. Nesta atividade o principal foco será a construção das classes **Deck** e **Util**. Crie um projeto chamado Lab3. No projeto crie três pacotes com o seguinte formato: **com.seuPrimeiroNome.util**, **com.seuPrimeiroNome.base** e **com.seuPrimeiroNome.driver**. Cole a classe do **Card.java** dentro do projeto no pacote base.

3 Classe Deck

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

- cards (array de objetos da classe **Card**)
- size (número inteiro)
- rnd (atributo **estático** da classe Random[2])

Um exemplo da classe é dado a seguir.

```
1 public class Deck {
2     private static Random rnd = new Random();
3     private Card[] cards;
4     private int size;
5 }
```

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

```
1     public Deck() {
2         this.cards = new Card[10];
3         this.size = 0;
4     }
```

[1] https://en.wikipedia.org/wiki/Svoyi_Koziri

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

A classe Deck possuirá dois métodos, um que adiciona e outro que remove cartas: addCard() e removeCard(). Como convenção, iremos adicionar e remover cartas sempre do final do array cards. O código desses métodos é dado a seguir:

```
    public void addCard (Card card) {
        this.cards[this.size] = card;
        this.size++;
    }

    public Card removeCard() {
        this.size--;
        return this.cards[this.size];
    }
```

A classe Deck também possuirá um método shuffle() 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:

```
    public void shuffle() {
        for (int i = 1; i < this.size; i++) {
            int j = Deck.rnd.nextInt(i + 1); // Sorteia um numero no intervalo [0, i]
            if (j != i) {
                Card aux = this.cards[i];
                this.cards[i] = this.cards[j];
                this.cards[j] = aux;
            }
        }

        // Comandos para imprimir as cartas em ordem reversa aqui
    }
```

A utilização do atributo rnd para sortear um número dentro de um intervalo não é a única utilidade da classe Random, para mais informações consulte a documentação[3].

Após embaralhar as cartas, o método shuffle 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.

4 Classe Util

Crie uma classe chamada Util dentro do pacote **util**. Nessa classe implemente dois novos métodos sobrecarregados **modify** da classe Card. Esses métodos devem ser estáticos e substituirão os métodos antigos (apague os antigos na classe Card). As assinaturas dos métodos devem ser como segue:

```
    public static void modify (Card card, String suit);
    public static void modify (Card card, String suit, char rank);
```

5 Collection

A linguagem Java disponibiliza para seus programadores um conjunto de classes e interfaces chamado Collection[4].

[3] https://docs.oracle.com/javase/8/docs/api/java/util/Random.html
[4] https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html

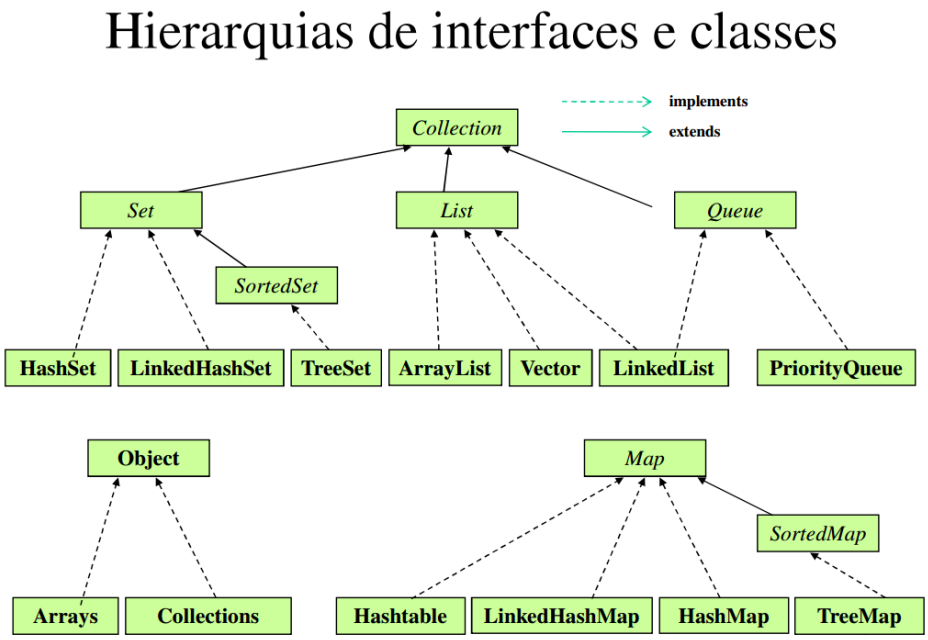

Figura 1: Hierarquia Collection.

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

- 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 DeckArrayList. Nessa classe iremos utilizar ArrayList[5] ao invés de array. O único atributo da classe será um List de objetos Card. Veja mais abaixo como é a classe e seu construtor.

```
public class DeckArrayList {
    private List<Card> cards;

    public DeckArrayList() {
        this.cards = new ArrayList<Card>();
    }
}
```

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.

```
    public static final int MAX_CARDS = 30;
```

Programe os métodos addCard, removeCard e shuffle. Verifique no método addCard se o baralho não chegou no limite (30 cartas) antes de adicionar uma nova. No método shuffle não utilize mais o algoritmo anterior, embaralhe os objetos do List com o método **shuffle** da classe Collections[6]. 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.

6 Classe Main

Crie uma nova classe Main no pacote **driver** e escolha a opção para gerar automaticamente o método main[7].

Na função main instancie três objetos arbitrários da classe Card e um novo objeto de Deck. Adicione as cartas no deck e chame o método shuffle. 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 DeckArrayList.

Somente para o objeto DeckArrayList, execute pelo menos uma vez o método removeCard e mostre a carta que foi removida.

Também na função main, utilize os dois métodos modify da classe Util para modificar duas cartas de sua escolha. Imprima o estado das cartas modificadas.

7 Tarefas

Faça a documentação Javadoc de todas as classes e 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 Deck, o que acontece se adicionarmos mais de 10 cartas? Que tipo de erro o Java acusará? (Se não souber, teste!)

[5] https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

[6] https://docs.oracle.com/javase/8/docs/api/java/util/Collections.html

[7] Dica: 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.

2. No método removeCard de Deck, 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 **rnd** não é inicializado no construtor da classe Deck? 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 Deck com ArrayList e não com vetor?

8 Submissão

Para submeter a atividade utilize o Moodle (`https://www.ggte.unicamp.br/ea`). Crie um arquivo .txt 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 MC322.

Data de entrega

- Dia **05/09** até às 23:59h