













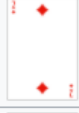











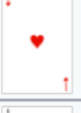

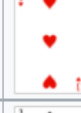
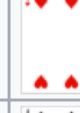

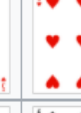




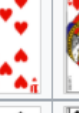
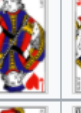
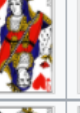















Ordenando Cartas

Descrição da parte 1

Na primeira fase desse trabalho você deve implementar uma **classe** chamada **Carta** que representa uma carta de baralho, lembrando que uma carta real sempre tem um **valor** e um **naipe** e em um baralho tradicional temos 52 cartas distintas, veja abaixo:

valor naipe	1	2	3	4	5	6	7	8	9	10	Valete	Dama	Rei
Paus:													
Ouros:													
Copas:													
Espadas:													

Baralho tradicional com 52 cartas

A **classe Carta** deve representar uma carta real e atender as seguintes especificações:

- Os atributos da **classe Carta** (**valor** e **naipe**) devem ser do tipo **String** e são **encapsulados**. Para isso, utilize o **modificador de acesso privado** para definir os atributos da classe. **Importante:** A classe Carta não é sensível ao caso, ou seja, para o atributo **valor** e **naipe**, “valete” e “paus” e “Valete” e “Paus” é a mesma carta.
- Deve existir um **construtor** que permita a um objeto dessa classe ser instanciado com valores específicos para os atributos da classe (**valor** e **naipe**).
- Deve existir um outro **construtor** sem parâmetros que define uma **carta curinga**. Uma carta curinga não tem **naipe** (que pode ser representado com uma **String** vazia), o atributo **valor** dessa carta deve ser preenchido com a **String** "Curinga" ou “curinga”.
- Deve existir um **método privado** que retorna o valor da carta, ou seja, um **valor inteiro**. Por exemplo:
 - Se a carta tem valor "As", o método retorna o número 1.
 - Se tem o valor "Valete", o método retorna o número 11.
 - Se tem o valor "Dama", o método retorna o número 12.
 - Se tem o valor "Rei", o método retorna o número 13.
 - Qualquer valor entre "2" e "10" deve retornar o número inteiro correspondente. No caso do "Curinga", o valor retornado deve ser o número 0.
- Deve existir um **método público** que retorna uma **String** com os atributos da carta no formato **valor** “ de ” **naipe**. Por exemplo: "As de espadas".
- Deve existir um **método público** que compara dois objetos da classe Carta pelo atributo **valor**. Na implementação do método uma das cartas ser passada de **forma implícita** (na chamada do método) e outra como parâmetro para o método, ou seja, de **forma explícita**.
 - Caso as duas cartas sejam iguais (tenham o mesmo valor) o método retorna 0.
 - Caso o valor da carta passada de forma implícita (na chamada do método) seja **menor** que o valor da carta passada de forma explícita o método retorna um **valor negativo**. Por exemplo a comparação entre “As” (implícito) e “Valete” (explícito) é **retornado um valor negativo**.
 - Caso o valor da carta passada de forma implícita (na chamada do método) seja **maior** que o valor da classe passada de forma explícita o método retorna um **valor positivo**. Por exemplo a comparação entre “Dama” (implícito) e “2” (explícito) é **retornado um valor positivo**.

Descrição da parte 2

Na segunda parte desse trabalho você deve escrever um programa (classe principal) com a função **main()** que lê um arquivo texto que tem armazenado uma sequência de cartas com valor e naipe, considere que no arquivo teremos no máximo **100 cartas**, ou seja, poderemos ter cartas repetidas na entrada. O seu programa deve ler as cartas do arquivo e armazená-las em **um vetor de objetos da classe Carta**.

Abaixo um exemplo de um arquivo texto com várias cartas:

```
As Espadas
Rei Ouros
Dama Espadas
2 Copas
3 Ouros
4 Copas
5 Copas
```

Depois de lido o arquivo e armazenado as cartas no vetor de objetos da **classe Carta**, o seu programa deve ordenar as cartas no vetor utilizando um algoritmo de ordenação visto nos semestres passados, e ao final o seu programa imprimir o vetor cartas ordenado na tela do computador, para entrada acima teríamos a seguinte saída:

```
As de Espadas
2 de Copas
3 de Ouros
4 de Copas
5 de Copas
Dama de Espadas
Rei de Ouros
```

Restrições do projeto

1. O programa deve estar bem documentado e implementado na **linguagem Java**, utilizando as boas práticas e conceitos vistos no paradigma orientada a objetos, tais como: atributos encapsulados (privados), passagem de parâmetro implícita e explícita, utilização de métodos.
2. Este trabalho pode ser desenvolvido **individualmente** ou **em duplas**, os nomes dos integrantes devem ser colocadas no arquivo fonte entregue, e nessa atividade siga também as **Orientações para Desenvolvimento de Trabalhos Práticos** disponível no **Moodle**.
3. A entrega do trabalho deve ser feita pelo **Moodle** (não serão aceitos trabalhos entregues via e-mail), basta que somente um dos integrantes da dupla entregue as classes do programa na linguagem Java. O trabalho será avaliado de acordo com os seguintes critérios:
 - Funcionamento do programa;
 - O quão fiel é o programa quanto à descrição do enunciado;
 - Indentação, comentários e legibilidade do código e clareza na nomenclatura de variáveis;
 - **Não é permitido** o uso de variáveis globais (**static**);
 - **Também não é permitido** o uso de classes prontas da Linguagem Java para representar o vetor de objetos da **classe Carta**. Tais como **List** ou **ArrayList**.