Programação Orientada por Objetos

2018/2019

Trabalho de Laboratório – Curso EI

3

Objetivos:
Polimorfismo.

Programa:
Pretende-se desenvolver um programa jogos de cartas. Um dos jogos a implementar é o jogo da sueca.

Regras de implementação:

- Criar a aplicação utilizando o IDE BlueJ.
- Implementar o código necessário e testar no fim de cada nível.
- Use as convenções de codificação adotadas para a linguagem Java (ver **Notas**).

Implementação:

Nível 1:

- Considere a seguinte classe **`Card`** e o tipo enumerado **`Suit`** (Naipe) que foram desenvolvidos inicialmente:

```
public class Card {
    private int number;
    private Suit suit;
    private int value;
    private boolean isFace;
    private String faceName;
}
```

```
public enum Suit {
    NONE, SPADES, HEARTS, DIAMONDS, CLUBS;

    @Override
    public String toString() {
        switch (this) {
            case SPADES:   return "espadas";
            case HEARTS:   return "copas";
            case DIAMONDS: return "ouros";
            case CLUBS:    return "paus";
            default:       return "";
        }
    }
}
```

- Foi decidido que a classe **`Card`** criada não servia porque tinha alguns problemas de coesão. Neste caso, estava a representar cartas que eram figuras e cartas que eram números e que se pretendiam diferenciar. Sendo assim, crie uma nova classe **`Card`** que irá servir de base à hierarquia de classes que se pretende implementar. Depois, derive as classes **`FaceCard`** que representa uma carta com uma figura (Valete, Dama ou Rei) e **`NumberedCard`** para as restantes cartas. Para distinguir as figuras crie um tipo enumerado **`FaceName`** que pode ter os valores **`JACK`**, **`QUEEN`** e **`KING`**. Inclua neste tipo o método **`toString()`** que devolve o nome da figura. Comece por incluir apenas os atributos nas classes criadas.
- Implemente na classe **`Card`** os seguintes construtores e métodos:
  - **`Card(Suit suit)`** – construtor que recebe apenas o naipe e inicia o atributo relacionado;
  - **`Card(Suit suit, int value)`** – Construtor que recebe e fornece os valores iniciais do naipe e da carta;
  - **`getValue`** os **`setValue`** – métodos seletor e modificador do atributo **`value`**;
  - **`getSuit`** – método seletor do atributo **`suit`**.
- Implemente na classe **`FaceCard`** os seguintes construtores e métodos:
  - **`FaceCard(FaceName face, Suit suit)`** – construtor que recebe qual a figura e o naipe e inicia os atributos relacionados;
  - **`FaceCard(FaceName face, Suit suit, int value)`** – construtor que recebe qual a figura, o naipe e o valor da carta e inicia os atributos relacionados;
  - **`getFaceName`** – método seletor do atributo **`faceName`**.

- Implemente na classe **NumberedCard** os seguintes construtores e métodos:
  - **NumberedCard(int number, Suit suit)** – construtor que recebe qual o número e o naipe e inicia os atributos relacionados;
  - **NumberedCard(int number, Suit suit, int value)** – construtor que recebe qual o número, o naipe e o valor da carta e inicia os atributos relacionados;
  - **getNumber** – método seletor do atributo **number**.
- Para testar as classes anteriores, crie num método estático **run** de uma classe **Program**, um "valete de paus" e um "três de ouros" e mostre o seu nome ou número e o naipe. Não necessita de adicionar nenhum outro método às classes criadas.

Nível 2:

- Para que seja mais simples obter o nome da carta, adicione o método **getName** às classes **FaceCard** e **NumberedCard**. Este método deve retornar um texto com o nome da carta. Se for uma figura é o nome da figura. Se for um número é o número por extenso. O número 1 deve retornar o texto "ás".
- Para testar os métodos criados, no método **run** definido anteriormente crie um **ArrayList** de cartas e adicione-lhe as cartas criadas. Adicione ainda um ás de paus. Depois, através de um ciclo **for-each** mostre os nomes de todas as cartas da coleção usando o método **getName** criado (usa-se o polimorfismo).

Nível 3:

- Redefina nas classes **FaceCard** e **NumberedCard** o método **toString()**. Não o implemente na classe **Card**. Este método deve retornar um texto com o nome e o naipe da carta. Por exemplo, "ás de ouros", "valete de paus".
- Para testar os métodos **toString()** criados, dentro do método **run** adicione ao **ArrayList** definido antes, mais um rei de copas e uma dama de espadas e mostre o nome de todas as cartas da coleção usando um ciclo **for-each** onde utiliza o método **toString** criado.

Nível 4:

- Pretendemos agora representar um baralho de cartas. Para isso crie a classe **Deck**. Esta classe deve possuir apenas um atributo **cards** que guarda as cartas do baralho. Use, neste caso, uma coleção **ArrayList**. Crie um construtor sem argumentos que cria a coleção usada e um segundo construtor que recebe uma lista de cartas e copia as cartas dessa lista para o atributo **cards**.
- Defina ainda na classe **Deck** os seguintes métodos:
  - **void addCard(Card card)** – adiciona uma carta ao baralho;
  - **boolean removeCard(Card card)** – retira uma carta do baralho, removendo-a da coleção. Se a carta existir e for removida com sucesso devolve **true**, caso contrário, devolve **false**;
  - **void clear()** – limpa o baralho, ficando sem cartas.
  - **String toString()** – devolve um texto com o nome e naipe de todas as cartas que estão no baralho, uma por linha.
- Para testar a classe criada, defina apenas um baralho no fim do método **run**. Este baralho deve conter todas as cartas que estavam na coleção de cartas definida nos níveis anteriores.

Nível 5:

- Como estava definido nos objetivos iniciais, pretendia-se usar o código criado para a implementação de um "jogo da Sueca". Sendo assim, quer-se criar apenas o baralho de cartas sem muitas funcionalidades. Para isso defina a classe **SuecaDeck** que representa um baralho de cartas para o jogo da Sueca. Nota: no nome da classe não foi traduzida a palavra "Sueca" porque este jogo é jogado principalmente por países lusófonos e a tradução direta não faria sentido (https://pt.wikipedia.org/wiki/Sueca_(jogo_de_cartas) .

- No construtor sem argumentos da classe **SuecaDeck**, limpe o baralho de cartas e crie todas as cartas do jogo da sueca. Neste caso, deve adicionar ao baralho as 40 cartas deste jogo que vão do Ás (1) ao 7, mais o rei, dama e valete para cada um dos naipes.
- Para testar, no método **run**, crie um baralho de cartas da sueca (um objeto da classe **SuecaDeck**) e mostre todas as cartas desse baralho no ecrã.
- Para simular o "dar cartas", crie o método **Card getRandomCard()** na classe **Deck**. Este método deve retirar aleatoriamente uma carta do baralho e devolvê-la ou devolver **null** se não existirem cartas. Use a classe **Random** do Java para selecionar a carta a retirar.
- Para testar, no método **run**, retire 32 cartas do baralho de cartas da sueca criado antes e mostre as cartas que restaram.

Notas:

Para os identificadores siga as convenções adotadas normalmente, em particular:

1) A notação **camelCase** para o nome das variáveis locais e identificadores de atributos e métodos.
2) A notação **PascalCase** para os nomes das classes.
3) Não utilize o símbolo '_', nem abreviaturas nos identificadores.