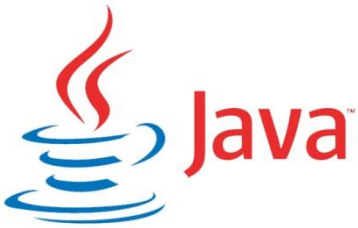
Lógica de **P**rogramação

Unidade 17 – Criando um Jogo



QI ESCOLAS E FACULDADES
Curso Técnico em Informática

SUMÁRIO

INTRODUÇÃO

Nesta unidade criaremos um jogo utilizando a linguagem Java para treinarmos associação entre classes e objetos sobre outra perspectiva.

O JOGO

O jogo que criaremos será o JanKenPon (famoso Pedra X Papel X Tesoura). Participam dois jogadores.

> *Para jogar, os dois jogadores dizem "Jan ken pon!" e, ao falar "pon", mostram simultaneamente a mão com uma dessas formas. A* ***pedra ganha da tesoura*** *(pois é capaz de quebrá-la),* ***a tesoura ganha do papel*** *(pois pode cortá-lo) e o* ***papel ganha da pedra*** *(pois pode embrulhá-la).*
> *** http://www.aprendendojapones.com/2008/06/07/jankenpon/*

Faremos o usuário jogar contra o computador. Assim, o jogador escolhe sua jogada e o computador escolhe a dele (via sorteio). Comparamos as jogadas de acordo com as regras e apontamos o vencedor (ou empate). Cada jogada pode ser 1 (Pedra), 2 (Papel) ou 3 (Tesoura).

O PROJETO

Podemos dividir nosso problema em duas classes: JanKenPon e Jogo. A primeira representa a jogada que cada um faz e a segunda controla o jogo. Pensemos em cada jogada e não no jogo; cada uma pode ter o valor 1, 2 ou 3. Após escolher a jogada, não podemos mais modifica-la, a não ser que iniciemos outra partida. Então teremos:

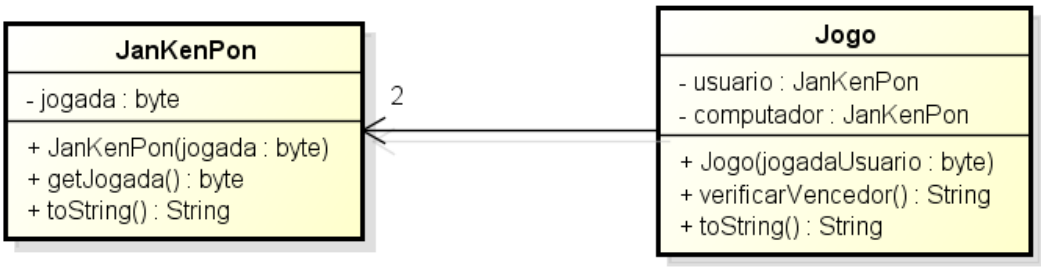
JanKenPon
- jogada:byte
+JanKenPon(jogada:byte) +getJogada():byte +toString():String

A jogada foi definida como *byte*, pois só poderá armazenar 1, 2 ou 3. O valor da jogada será definido no construtor e não poderá ser modificado (por isso temos o *get*, mas não o *set*). Por último o método toString, que deverá mostrar o nome da jogada conforme o número.

Em seguida, vamos montar a classe Jogo. O jogo é composto por dois jogadores: usuário e computador. Quando os dois jogam, verificamos o vencedor e exibimos. Portanto:

Jogo
- usuario: JanKenPon - computador: JanKenPon
+Jogo(jogadaUsuario: byte) +verificarVencedor():String +toString():String

Observe que os atributos usuario e computador são ambos da classe JanKenPon. Para criar um jogo, precisamos da jogada do usuário (construtor), pois a do computador é sorteada; precisamos do método que verifica o vencedor e o método *toString*. Observe também que não criamos os *gets* e *sets*, pois não há necessidade. Vamos representar o diagrama completo com a associação:



Classe JanKenPon

```
public class JanKenPon {
    private byte jogada;

    public JanKenPon(byte jogada){
        this.jogada = jogada;
    }

    public byte getJogada() {
        return jogada;
    }

    public String toString(){
        switch(this.jogada){
            case 1: return "Pedra";
            case 2: return "Papel";
            default: return "Tesoura";
        }
    }
}
```

Classe Jogo

```
1  public class Jogo{
2      private JanKenPon usuario;
3      private JanKenPon computador;
4
5      public Jogo(byte jogadaUsuario ){
6          /*definir jogada do usuario*/
7          this.usuario = new JanKenPon(jogadaUsuario);
8          /* sortear e definir a jogada do computador*/
9          byte sorteio = (byte)(Math.random()*3+1);
10         this.computador = new JanKenPon(sorteio);
11     }
12
13     public String verificarVencedor(){
14         byte usu = this.usuario.getJogada();
15         byte comp = this.computador.getJogada();
16         if(usu == comp){
17             return "Empate";
18         }else if(usu==1 && comp==3 || usu==2 && comp==1 || usu==3 && comp==2){
19             return "Usuário vence!";
20         }else{
21             return "Computador vence!";
22         }
23     }
24     public String toString(){
25         return "Usuário jogou " + this.usuario
26             +"\nComputador jogou " + this.computador;
27     }
28 }
```

Os dois atributos da classe são objetos da classe JanKenPon. Portanto, no construtor vamos instanciar cada um. A jogada do usuário vem da *Main* (argumento), mas a jogada do computador deve ser sorteada. Para isso usamos o método *random*() da classe *Main:*

```
byte sorteio = (byte)(Math.random()*3+1);
```

O método *random*() gera um número aleatório de 0 a 0,999999. Multiplicamos pelo número de possibilidades (3) e somamos 1 para anular o 0. O comando (*byte*) desconsidera as casas decimais e fica somente com um número inteiro.

O método verificarVencedor analisa primeiramente se deu empate (se ambas as jogadas forem iguais); depois verifica se o usuário venceu, tendo três possibilidades:

Usuário jogou	Computador jogou	Quem ganha?

Pedra (1)	Tesoura (3)	Usuário
Papel (2)	Pedra (1)	Usuário
Tesoura(3)	Papel (2)	Usuário

Caso contrário quem vence é o computador.

Classe Main

```
1  import java.util.Scanner;
2  public class Main{
3      public static void main(String args[]){
4          Scanner ler = new Scanner(System.in);
5          byte escolha;
6
7          do{
8              System.out.println("1 - Jogar");
9              System.out.println("0 - Sair");
10             escolha = ler.nextByte();
11
12             switch(escolha){
13                 case 1:
14                     System.out.println("ESCOLHA SUA JOGADA!");
15                     System.out.println("(1) Pedra");
16                     System.out.println("(2) Papel");
17                     System.out.println("(3) Tesoura");
18                     byte jogadaUsuario = ler.nextByte();
19                     Jogo j1 = new Jogo(jogadaUsuario);
20                     System.out.println(j1);
21                     System.out.println("Resultado: " + j1.verificarVencedor());
22                     break;
23                 case 0:
24                     System.out.println("JOGO ENCERRADO");
25             }
26         }while(escolha!=0);
27     }
28 }
```

Na classe *Main* apresentamos ao usuário 2 opções: Jogar ou encerrar. Se ele escolhe 1, o sistema solicita que ele escolha uma das três opções de jogo; após ele digitar, o sistema cria o jogo (linha 19), mostra o jogo (linha 20) e apresenta o resultado (linha 21).