Torneio de Truco
(Sistema Especialista)

Prof. Dr. Luciano Antonio Digiampietri

Escola de Artes, Ciências e Humanidades da USP

Roteiro

- Contexto Educativo
- Descrição do Jogo de Truco
- Objetivo
- Descrição do Projeto *TorneioDeTruco*
- Exemplo de Resultado

Contexto Educativo

- O objetivo desta apresentação é propor um trabalho aos alunos da disciplina de Inteligência Artificial.
- O trabalho consiste na implementação de um jogador de truco que irá competir com os demais *bots* implementados pela turma.

Descrição do Jogo

- Utilizaremos o Jogo de Truco jogado entre (apenas) duas pessoas como referência para este projeto.
- A seguir descrevemos:
  - Os Termos;
  - As Regras;
  - Os Aspectos de Implementação.

Termos

- **PARTIDA** - Jogo valendo 12 pontos, conseguidos através das "mãos".
- **MÃO** - Fração da partida, vale 1 ponto e poderá ter seu valor aumentado a 3, 6, 9 e até 12 pontos, é disputada em melhor de 3 rodadas.
- **RODADA** - É a fração da mão, em cada rodada, os jogadores mostram uma carta.
- **TRUCO** - Para pedir o aumento do valor da "mão" para 3.
- **SEIS** - Para pedir o aumento do valor da "mão" para 6.
- **NOVE** - Para pedir o aumento do valor da "mão" para 9.
- **DOZE** - Para pedir o aumento do valor da "mão" para 12.
- **EMPATAR** - Quando as cartas jogadas pelos jogadores numa determinada rodada, tem o mesmo valor.
- **MÃO-DE-ONZE** - Quando uma das equipes conseguir chegar a 11 pontos na partida.
- **MÃO-DE-FERRO** - É a mão-de-onze especial, quando todos conseguem chegar a 11 pontos na partida.
- **MANILHAS** - São as 4 maiores cartas do jogo, as únicas dentre as 40 que não podem empatar, serão elas: 4 de paus, sete de copas, ás de espadas e sete de ouros.

Termos e Regras

- **Hierarquia das Cartas:**
  - Em ordem decrescente, começa pelas 4 manilhas: Zap (4 de paus), 7 Copas, Espadilha (ás de espadas) e 7 Ouros. As demais cartas também em ordem decrescente: 3, 2, Ás, K, J, Q, 7, 6, 5 e 4.
- **Aumentar o valor do jogo:** durante qualquer momento você poderá aumentar o valor do jogo (pedir truco, seis, nove ou doze) desde que você não tenho sido o último jogador a aumentar o jogo na mão atual e desde que nenhum dos jogadores possua 11 pontos.

Regras

- **Ganhando a mão:** um jogador ganhará os pontos da mão se:
  - Ganhar duas rodadas da mão;
  - Ganhar a primeira rodada e empatar a segunda;
  - Ganhar a segunda após ter empatadao a primeira;
  - Ganhar a última rodada da mão após ter empatado as duas primeiras.
- **Mão-de-onze:**
  - A mão-de-onze acontece quando um dos jogadores atinge 11 pontos na partida. O jogador que tiver 11 pontos irá decidir se irá ou não jogar, se não, o outro time ganhará um ponto, se sim, a mão valerá 3 pontos e não poderá ter seu valor aumentado.
- **Mão-de-ferro:**
  - A mão-de-ferro acontece quando ambos jogadores estão com 11 pontos na partida. Quem vencer a mão de ferro ganha o jogo. Ninguém poderá trucar durante a mão de ferro.

Aspectos de Implementação

- Cada carta receberá um valor de 1 a 10 (1 para os quatros, 2 para os cincos ... 9 para os dois e 10 para os três), as manilhas receberão valores 11 (sete de ouros), 12 (ás de espadas), 13 (sete de copas) e 14 (zap – quatro de paus). Com isso, será mais fácil de implementar o sistema.

Objetivo

- O objetivo deste trabalho é desenvolver uma ou duas classes que implementem a seguinte interface de um jogador de truco:

```
public interface Jogador {
    boolean aceitaMaoDeOnze(LogAtualDoJogo estadoDoJogo);
    Jogada aceitaAumento(LogAtualDoJogo estadoDoJogo);
    Jogada jogar(LogAtualDoJogo estadoDoJogo);
    String nomeDoJogador();
}
```

Descrição da Interface (1)

- Jogada:
  - é um tipo enumerável que pode assumir os seguintes valores: JogarCarta1, JogarCarta2, JogarCarta3, AceitarAumento, SolicitarAumento, AbandonarRodada. Em um dado momento de uma rodada, o jogador poderá ter de zero a três cartas que estarão num arranjo de objetos do tipo Carta. JogarCarta1 significa jogar a carta que está na posição 0 (zero) do arranjo, jogarCarta2 significa jogar a carta que está na posição 1 do arranjo e assim por diante.

Descrição da Interface (2)

- O método **aceitaMaoDeOnze** deve retornar true se o jogador aceitar a mão de onze e false caso contrário.
- O método **aceitaAumento** pode retornar três opções: Jogada.AceitarAumento; Jogada.SolicitarAumento (ou seja, aumentar ainda mais a mão); ou Jogada.AbandonarRodada caso não queira aceitar o aumento.
- O método **jogar** poderá retornar as seguintes opções: Jogada.AbandonarRodada, Jogada.JogarCarta1, Jogada.JogarCarta2, Jogada.JogarCarta3, Jogada.SolicitarAumento.
- O método **nomeDoJogador** deve retornar um nome para o jogador que o seu grupo implementou.

Descrição da Interface (3)

- A Classe LogAtualDoJogo possui os seguintes métodos (todos já implementados) que serão usados para a tomada de decisão:
  - getCartasJogadasPeloAdversario()
  - getCartasJogadasPorVoce()
  - getCartasNaSuaMao()
  - getNumeroDaMaoAtual()
  - getNumeroRodadaAtual()
  - getPlacarDoAdversario()
  - getQuantidadeDeVezesQueAdversarioAceitouAumento()
  - getQuantidadeDeVezesQueAdversarioAumentou()
  - getQuantidadeDeVezesQueVoceAumentou()
  - getSeuPlacar()
  - getValorAtualDaRodada()
  - isAdversarioJaJogouNestaRodada()
  - isEuComecoEstaMao()
  - isFuiOUltimoAAumentar()

Descrição da Interface (4)

- O método **getCartasNaSuaMao**() retorna um arranjo de 0 a 3 Cartas, estas cartas estarão ordenadas em ordem crescente de valor (para facilitar seus cálculos).
- Note que, se você possuir apenas uma carta, você receberá um arranjo de apenas uma posição (sua carta estará na posição zero do arranjo) e as opções: JogarCarta2 e JogarCarta3 não farão sentido.

Testando a Implementação

- Para testar seu jogo, utilize a classe ExecutaTorneio, substituindo um dos jogadores exemplos pelo jogador do seu grupo.
- Há duas classes de jogadores exemplos que podem ser utilizadas para auxiliar no desenvolvimento de seu sistema especialista.

- A seguir, o projeto TorneioDeTruco será resumido.

Projeto Torneio de Truco

- Contém as classes básicas para a execução de um torneio de truco entre qualquer quantidade de *bots* cadastrados.
- Um torneio é composto de uma série de duelos (jogador X contra jogador Y) cada um composto por diversas partidas. Ganha um duelo o jogador que ganhar o maior número de partidas dentro daquele duelo.

Classe ExecutaTorneio

- É a classe executável do projeto.
- Nelas os jogadores são colocados num arranjo de jogadores e o número de partidas por duelo é definido.
- Esta classe executa os duelos entre todas as combinações de jogadores (dois a dois) e imprime o resultado do torneio (ordenado por vitórias em duelos).

Classe JogadorExemplo 2

- O *bot* JogadorExemplo1 possui todos os métodos aleatórios e por isso não será explicado.
- O *bot* JogadorExemplo2 possui uma lógica simples no método jogar e é aleatório nos outros métodos.

Classe JogadorExemplo 2

- Método **aceitaAumento**: implementação aleatória, com apenas uma regra simples no final:

```
public Jogada aceitaAumento(LogAtualDoJogo estadoDoJogo) {
    Random rand = new Random();
    int teste = rand.nextInt(100);
    if (teste < 45) return Jogada.AbandonarRodada;
    else if (teste>=45 && teste<90){
        return Jogada.AceitarAumento;
    }
    if (estadoDoJogo.getValorAtualDaRodada()>=9) return
    Jogada.AceitarAumento;
    return Jogada.SolicitarAumento;
}
```

Classe JogadorExemplo 2

- Método **aceitaAumento**: implementação total aleatória:

```
public boolean aceitaMaoDeOnze(LogAtualDoJogo estadoDoJogo) {
    Random rand = new Random();
     int teste = rand.nextInt(100);
     if (teste < 90) return true;
    return false;
}
```

Classe JogadorExemplo 2

- Método **jogar**:
  - Solicita aumento se o valor médio das cartas na mão for maior ou igual a 6.
  - Se o adversário já tiver jogado, joga a menor carta que ganhe da carta do adversário.
  - Caso contrário, joga qualquer carta.

Método jogar

```
public Jogada jogar(LogAtualDoJogo estadoDoJogo) {
  Carta[] cartasNaMao;
  int valores = 0;
  cartasNaMao = estadoDoJogo.getCartasNaSuaMao();
  for (int i=0;i<cartasNaMao.length;i++){
    valores+=cartasNaMao[i].getValor();
  }
  if (valores>0) valores=valores/cartasNaMao.length;

  if (valores >= 6 && estadoDoJogo.getValorAtualDaRodada()<3 &&
      estadoDoJogo.getPlacarDoAdversario() <11 && estadoDoJogo.getSeuPlacar()<11 &&
      !estadoDoJogo.isFuiOUltimoAAumentar()) {
    return Jogada.SolicitarAumento;
  }

...
```

Método jogar (continuação 1)

```
if (estadoDoJogo.isAdversarioJaJogouNestaRodada()){
  int rodadaAtual = estadoDoJogo.getNumeroRodadaAtual();
  int valorDaCartaDoAdv =
    estadoDoJogo.getCartasJogadasPeloAdversario()[rodadaAtual-1].getValor();
  for (int i=0;i<cartasNaMao.length;i++){
    if (cartasNaMao[i].getValor() > valorDaCartaDoAdv){
        if (i==0) return Jogada.JogarCarta1;
        else if (i==1) return Jogada.JogarCarta2;
        else if (i==2) return Jogada.JogarCarta3;
    }
  }
}
...
```

Método jogar (continuação 2)

```
else{
    Random rand = new Random();
    int i = rand.nextInt(cartasNaMao.length);
    if (i==0) return Jogada.JogarCarta1;
    else if (i==1) return Jogada.JogarCarta2;
    else if (i==2) return Jogada.JogarCarta3;
  }
return Jogada.JogarCarta1;
}
```

Exemplo de Resultado

Abaixo, segue o resultado do torneio executado apenas entre 4 *bots* implementados pelo professor.

Duelos realizados: 6. Partidas realizadas: 6006.

Vitórias do Time 'Exemplo1': 17.92%; 0 duelo.

Vitórias do Time 'Exemplo2': 41.49%; 1 duelo.

Vitórias do Time 'Exemplo3': 64.50%; 2 duelos.

Vitórias do Time 'Exemplo4': 76.09%; 3 duelos.

http://www.uspleste.usp.br/digiampietri/jogos/