# Desenvolvimento Dirigido por Testes (TDD)

Cursos de Verão 2008

www.agilcoop.org.br

Mariana & Paulo

AgilCoop

# O que é Desenvolvimento Dirigido por Testes?

**Dúvidas?**

# Testes *a priori*

- Conhecer *design*
- Testar
- Implementar

- Conseqüência: Testes não são esquecidos devido a falta de tempo, pressão ou estresse, etc

# O que é Desenvolvimento Dirigido por Testes?

*"Código limpo que funciona"*

-- Ron Jeffries

# O que é Desenvolvimento Dirigido por Testes?


- Ciclo em passos pequenos:
  1. Escreva um teste que falha
  2. Faça o teste passar rapidamente
  3. Refatore

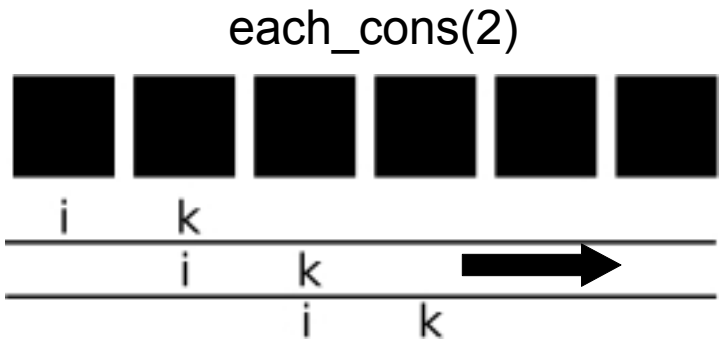
# Demonstração: Poker!

# Linguagem: Ruby

- Definindo uma classe: **class MinhaClasse end**
- Delimitador de métodos: **def meuMetodo end**
- Contrutor: **def initialize end**
- Criando uma instância: **Classe.new**
- Verificando um tipo de instância: **x.kind_of? Y**
- Constantes: **LETRA_MAIUSCULA = ...**
- Variaveis de instância: **@variavel**
- Falso: **false** e **nil**
- Métodos booleanos: **metodo?**

# + Linguagem: Ruby

- Getters/Setters: **attr_reader** e **attr_accessor**
  - Exemplo: **attr_reader :variavel**
- Equals (true ou false): **1 == 1** (= true)
- CompareTo (-1, 0, 1): **<==>**
- Append (add): **lista << elemento**
- Mapas: **{ chave => valor }**
- Alguns métodos auxiliares:
  - **each_cons(inteiro)**




  - string[indice, quantidade]: **“abc”[1,2]** (= “bc”)

# Arcabouço de Teste: rSpec

- describe Classe do ... end
- it “frase” do end
- kind_of? => be_kind_of
- dois_pares? => be_dois_pares
- variavel.should
- variavel.should_not

# Poker Simplificado

- Cartas: '2H 3S 4C 5D TD', 'JS QD KS AC 2S'
- Dois jogadores, 5 cartas (ordenadas) cada um
- Regras:
  - Maior carta (Jogo simples)
  - Par
  - 2 Pares
  - Trinca

# O que o programa deve fazer


- Entrada: 2 jogadores, 5 cartas cada um
  - Recebe as cartas como *strings* ordenadas por valor
  - Exemplo: '5O DC ... ' e '6E JP ...'
- Saída: quem venceu entre as duas mãos
  - Responde **Jogo1** ou **Jogo2**

# Tarefas pendentes

- Decidir quem ganhou no poker

# Jogos

- Cada jogo tem 5 cartas

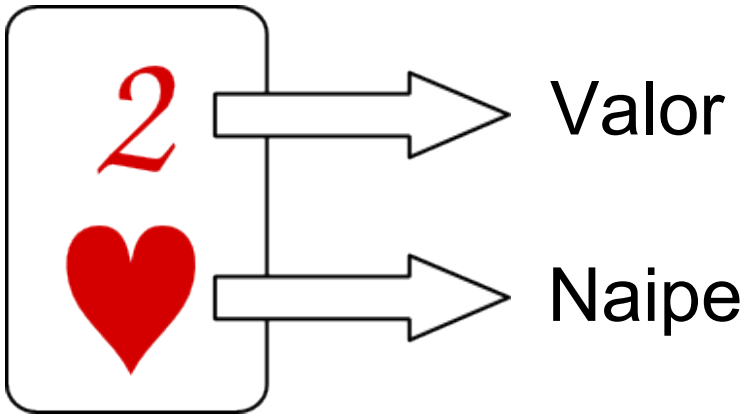

- Exemplo: '2E 5P 7C 8E DO'
- Existem diversos tipos de jogos, e queremos saber compará-los de acordo com as regras

# Tarefas pendentes

- Decidir quem ganhou no poker
- Jogo:
  - Deve ter 5 cartas
  - Deve comparar de acordo com as regras

# Cartas

- Cartas




- Naipes
  - E (espadas)
  - C (copas)
  - P (paus)
  - O (ouros)

- Valores: de 2 a 9, D é 10, J, Q, K, A
- Exemplos: *2*C *3*E *4*P *5*O *D*O *J*E *Q*O *K*E *A*P

# Tarefas pendentes

- Decidir quem ganhou no poker
- Jogo:
  - Deve ter 5 cartas
  - Deve comparar de acordo com as regras
- Carta:
  - Deve ter valor
  - Deve ter naipe
  - Deve comparar por valor

# Jogos

- Cada jogo tem 5 cartas


- Exemplo: '2E 5P 7C 8E DO'
- Existem diversos tipos de jogos, e queremos saber compará-los

# Tarefas pendentes

- Decidir quem ganhou no poker
- Jogo:
  - Deve ter 5 cartas
  - Deve comparar de acordo com as regras
- Carta

# Carta mais alta

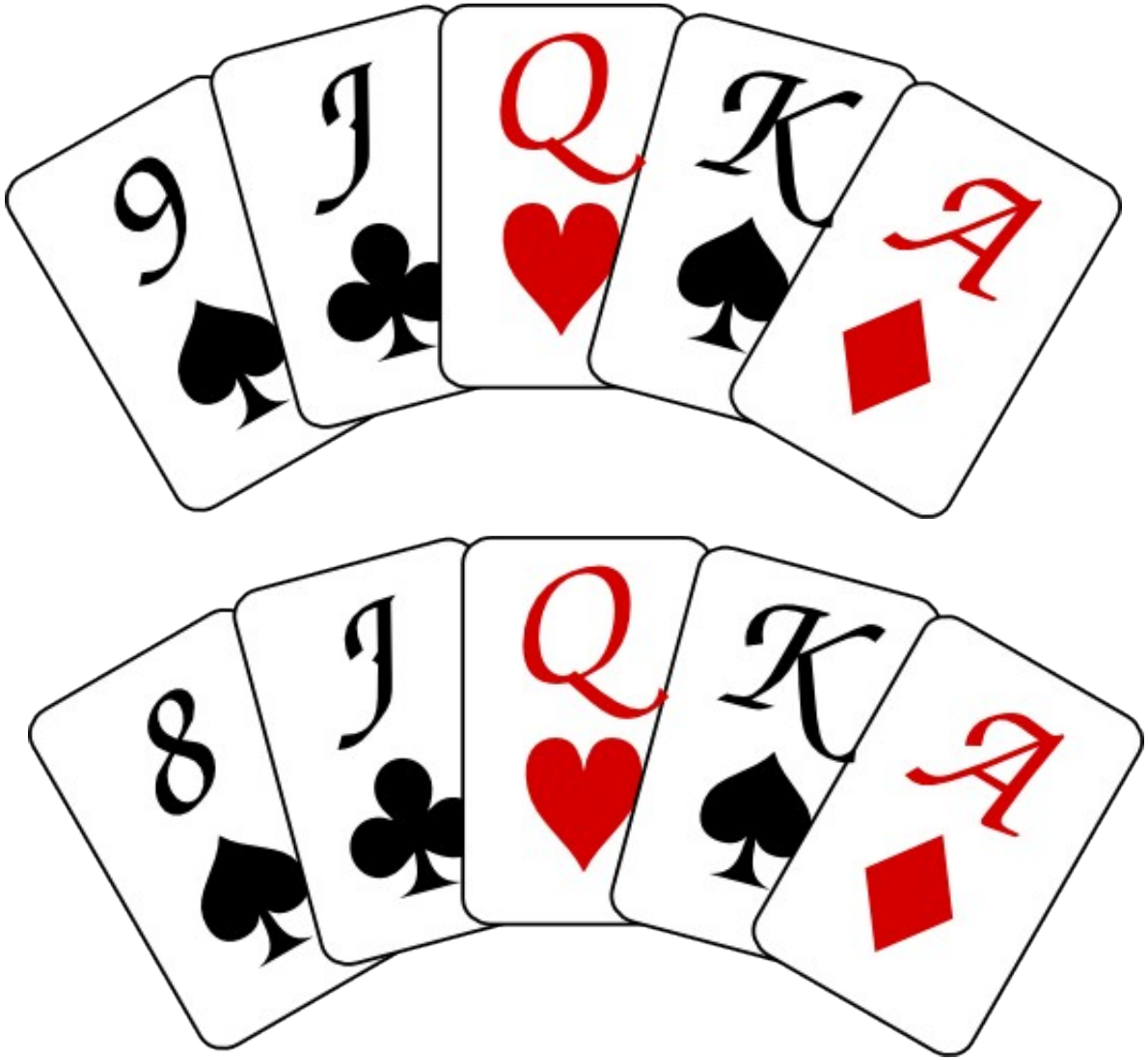

- '2E 5P 7C 8E DO'
- Vale a carta mais alta: 10
- Desempate pelo valor das cartas

# Base 16??

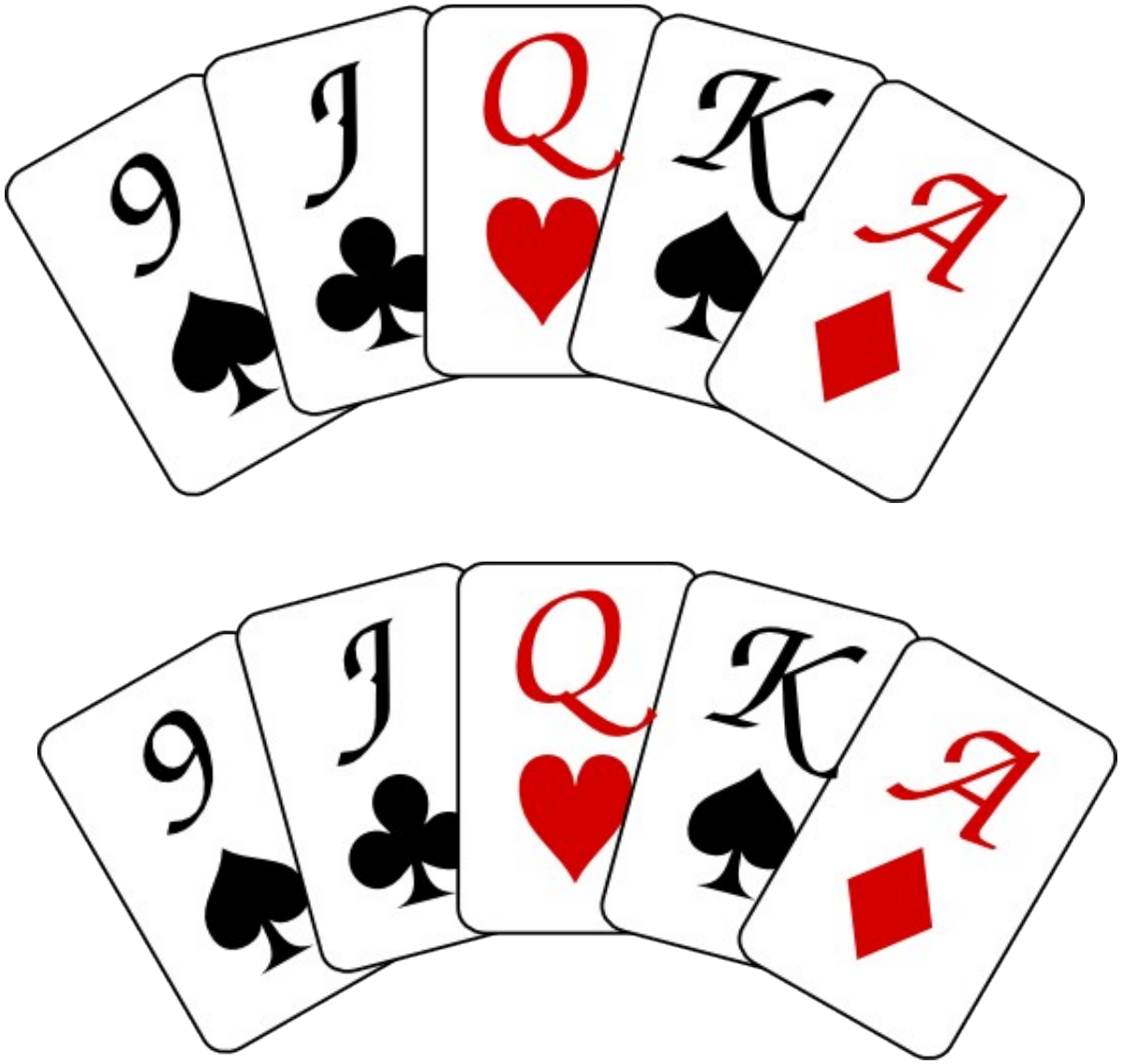
# SIMPLES

# Maior Simples contra Menor Simples

# Simples contra Simples Repetindo Maior

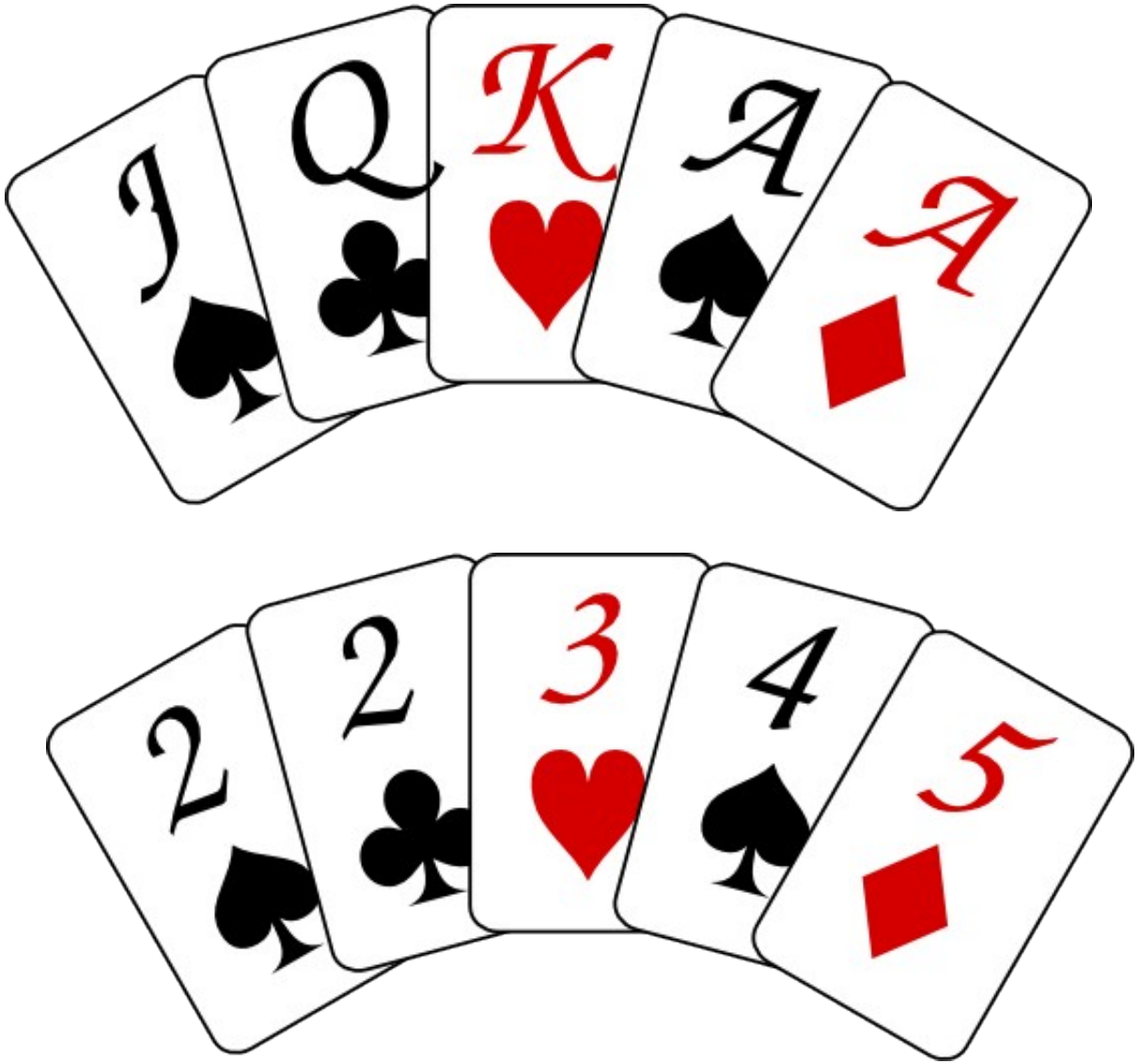


# Simples contra Simples: Empate

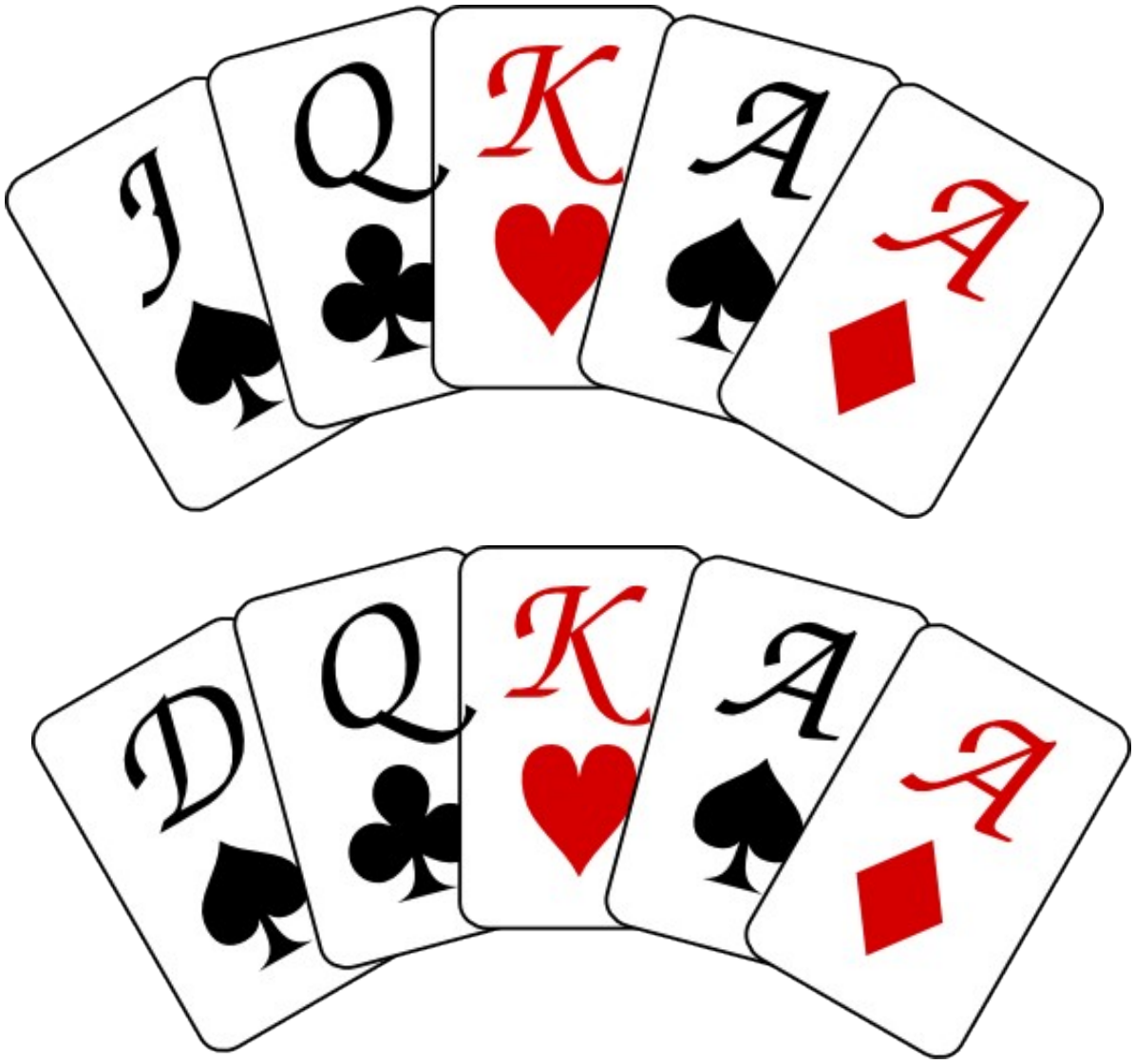


# PARES

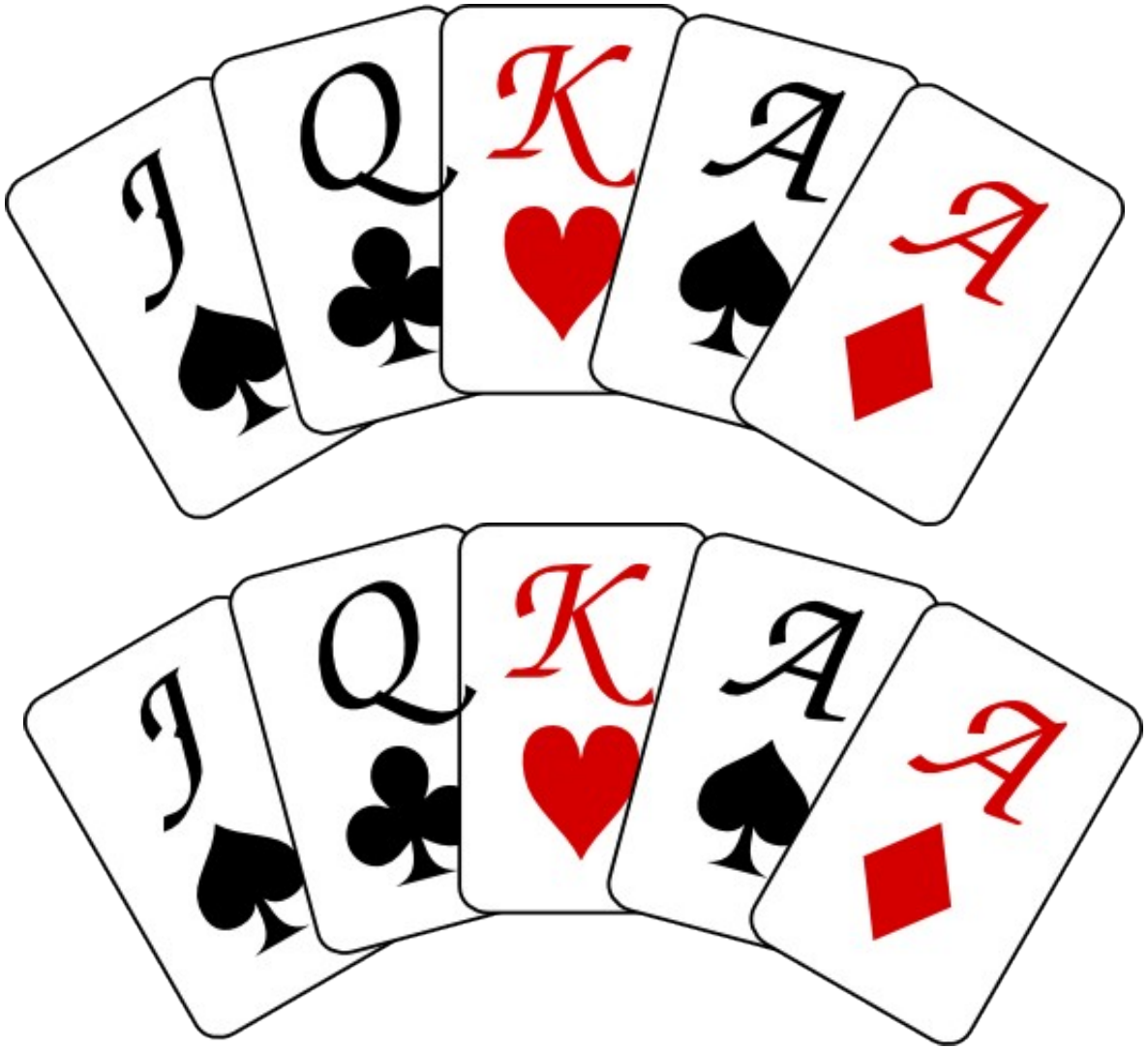
# Maior Par contra Menor Par



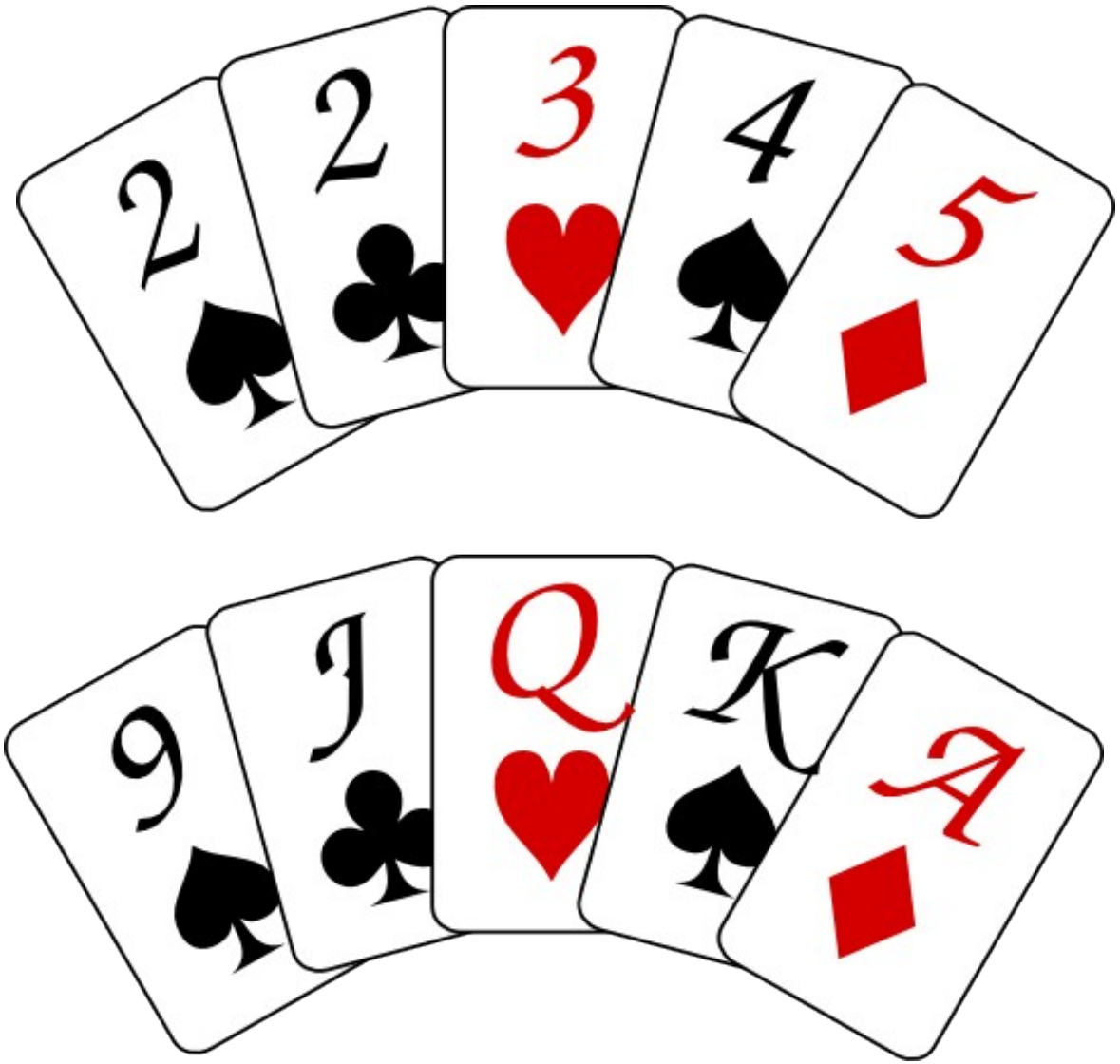
# Par contra Par com Par Igual



# Par contra Par: Empate

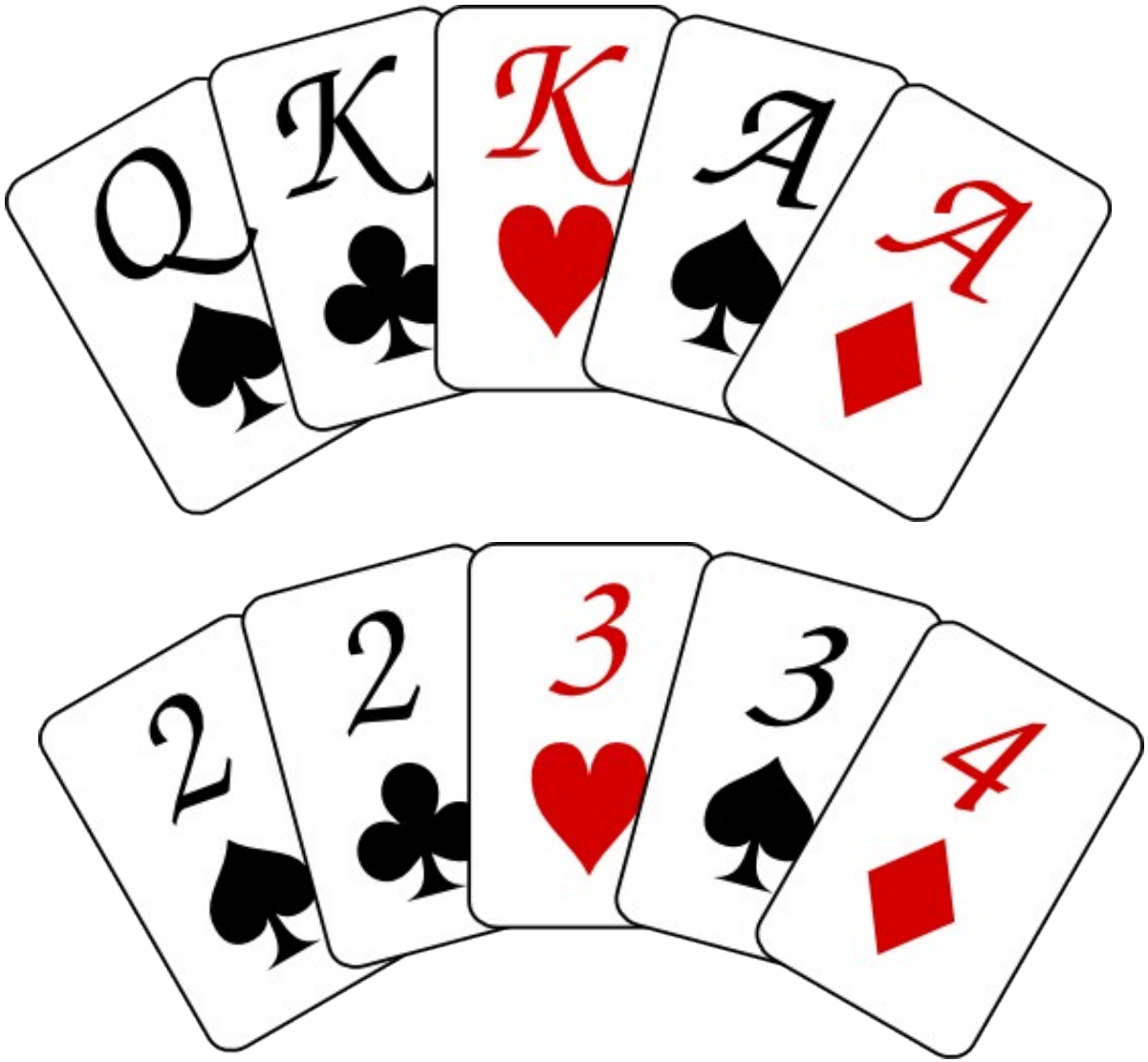


# Menor Par contra Maior Simples

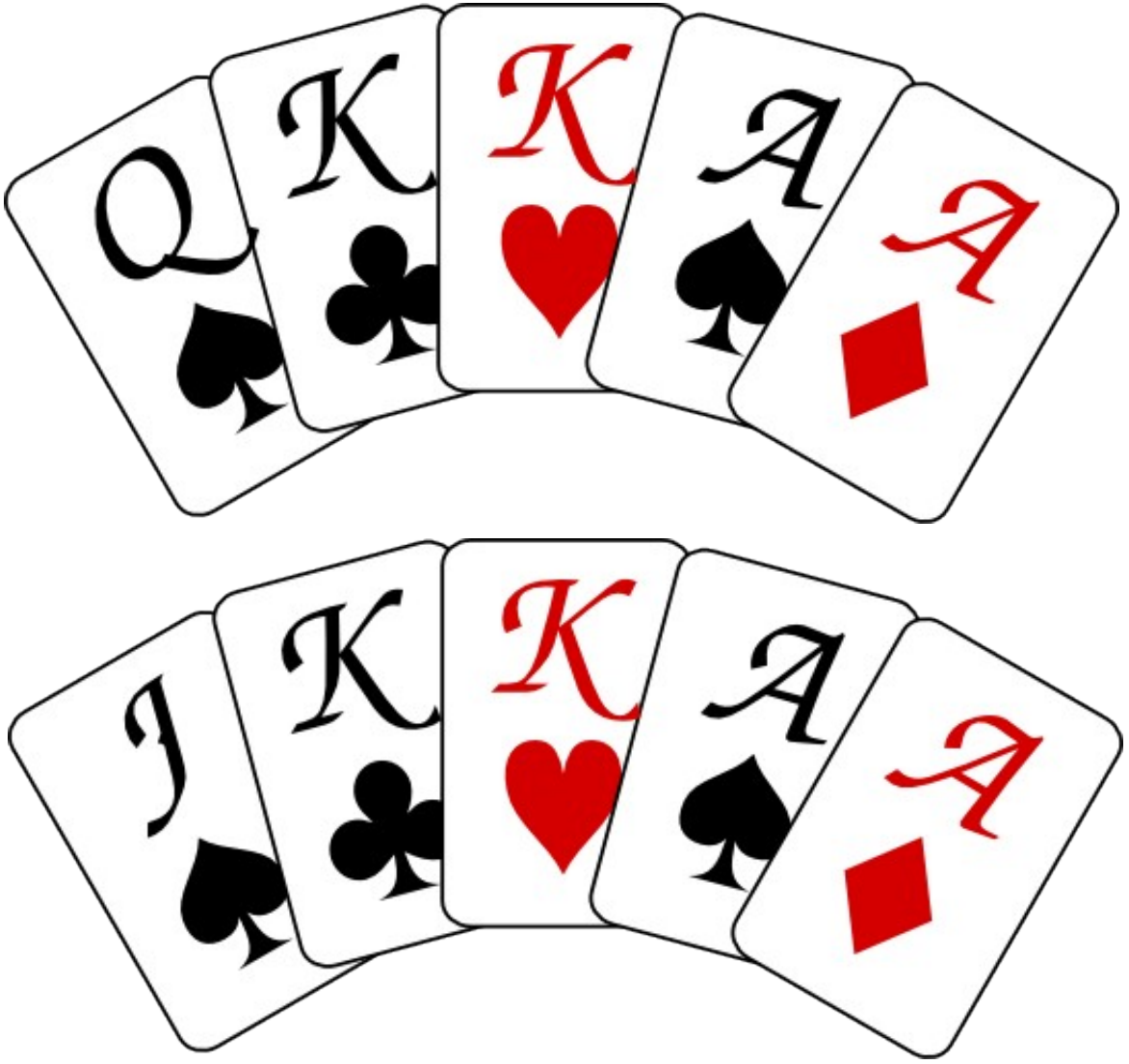


# DOIS PARES

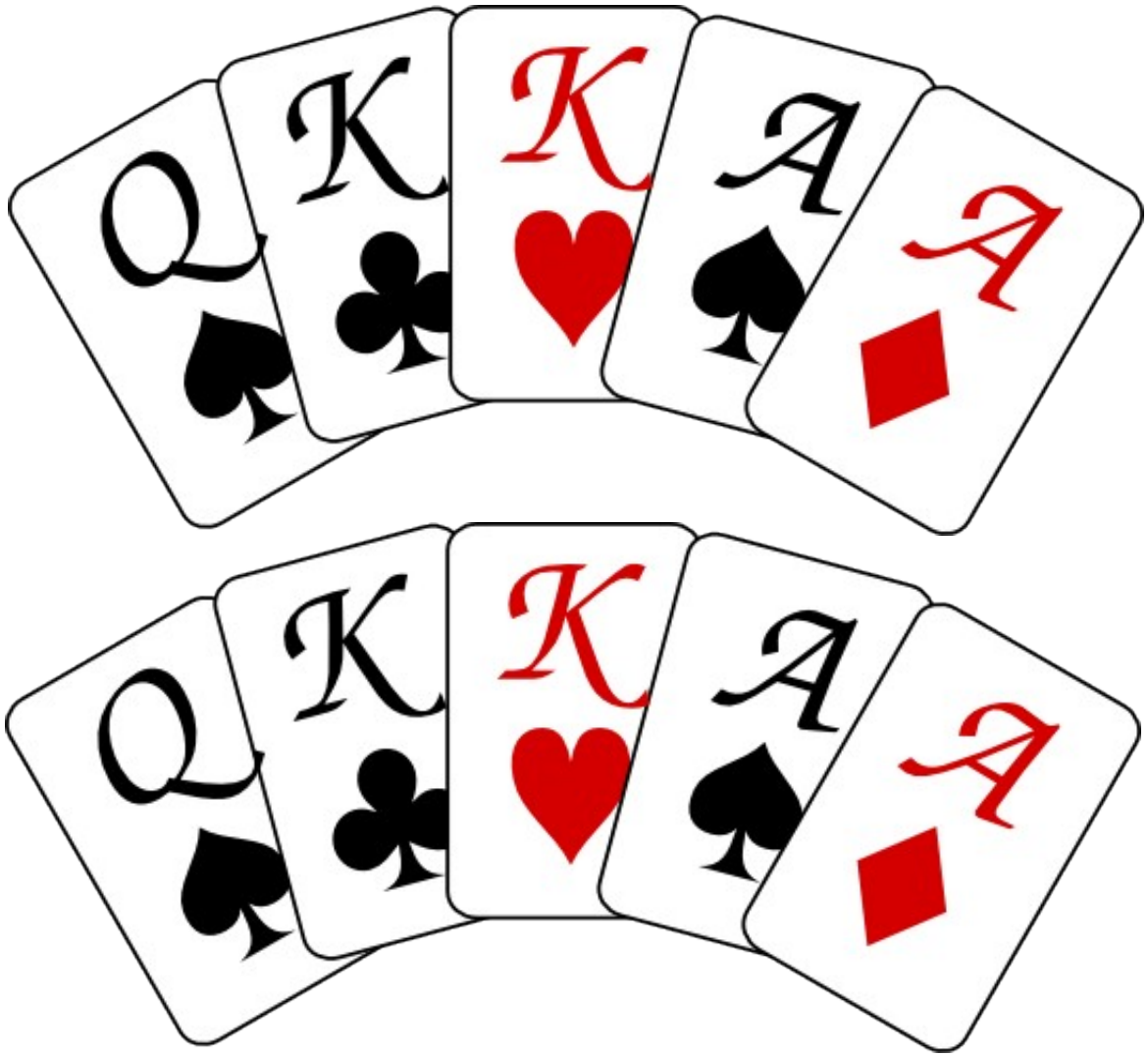
# Maior Dois Pares contra Menor Dois Pares



# Dois Pares contra Dois Pares com Dois Pares Iguais



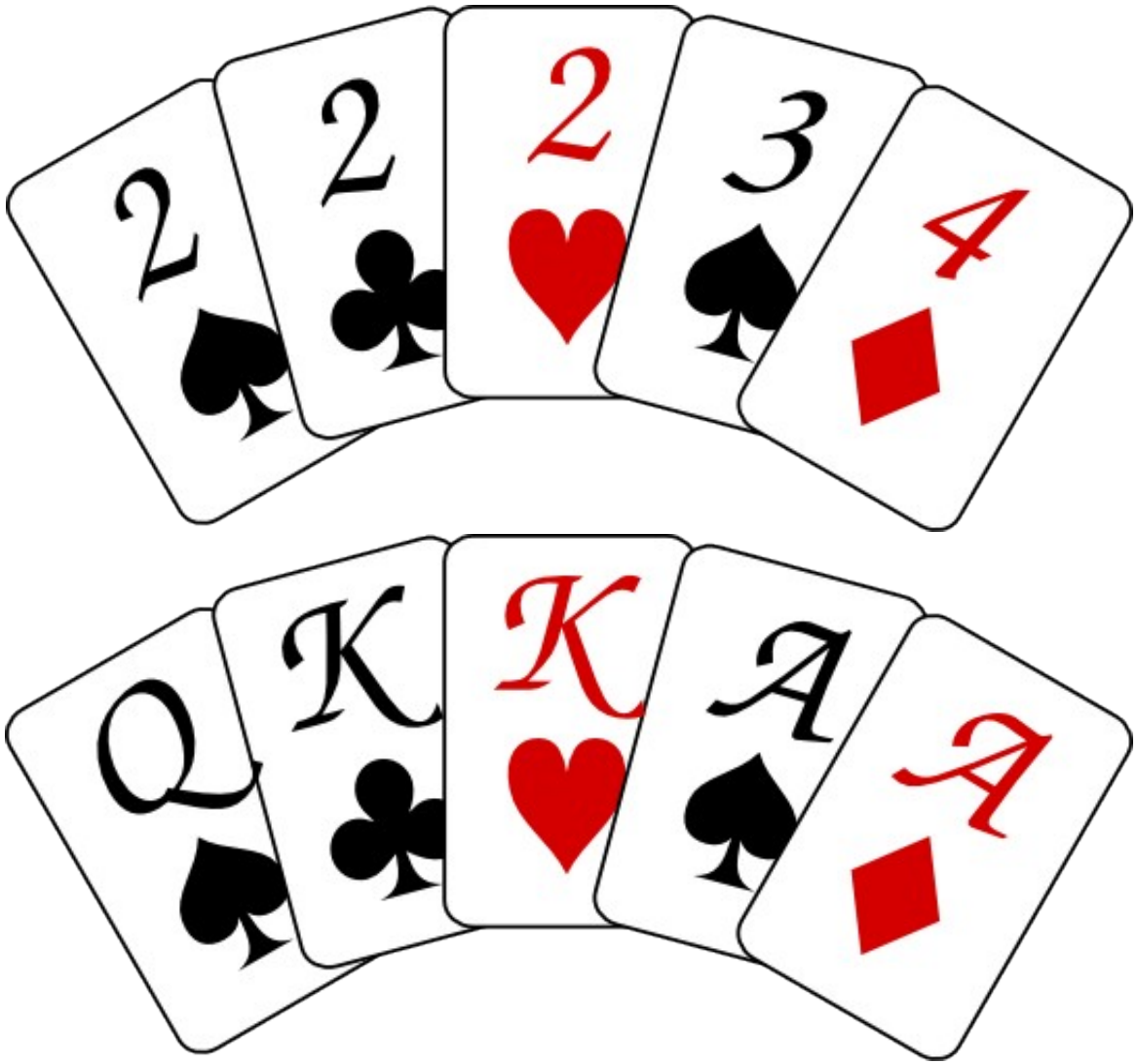
# Dois Pares contra Dois Pares: Empate



# Menor Dois Pares contra Maior Par

# TRINCA

# Menor Trinca contra Maior Dois Pares



# Tarefas pendentes

- Decidir quem ganhou no poker
- Jogo:
  - Deve ter 5 cartas
  - Deve comparar de acordo com as regras
- Carta
- Carta mais alta

# Conclusão

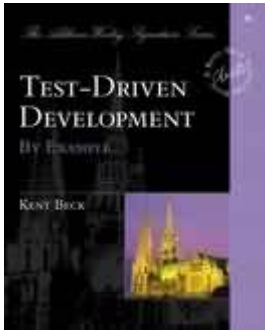
- *Design* evolui junto com o conhecimento
  - *Code for tomorrow, design for today (K. Beck)*
- Desenvolvimento com “Passos Pequenos”
- Expressa a intenção do programador com testes
- Servem como documentação

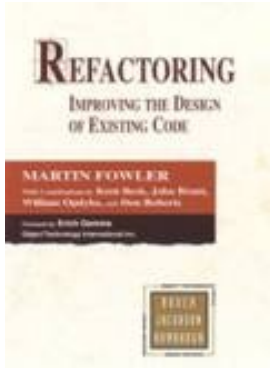
# Conclusão

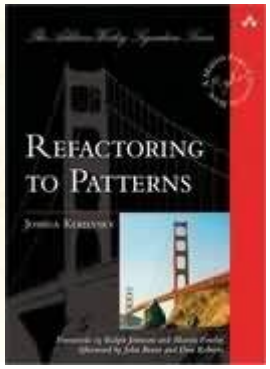
- O Código:
  - Nome dos testes definem o comportamento esperado
  - Fatorados (sem duplicação)
  - Alta cobertura
  - Evita código inútil
  - Alta qualidade do código
    - Refatorações são seguras com testes automatizados

# Referências

- Livros:
  - Kent Beck, “Test-Driven Development: By Example”, Addison-Wesley Professional, 2002
  - David Astels, “Test Driven Development: A Practical Guide”, Prentice Hall PTR, 2003
  - Martin Fowler et al, “Refactoring: Improving the Design of Existing Code”, Addison-Wesley Professional, 1999
  - Joshua Kerievsky, “Refactoring to Patterns”, Addison-Wesley, 2004







# Referências

- Online:
  - www.testdriven.com
  - www.xprogramming.com
  - www.dtsato.com/blog/default
  - www.agilcoop.org.br