# #Cálculo de probabilidade

S = espaço amostral. Todo.   
A = evento. Um grupo do Todo.

p(A) = n(A) / n(S)

p(A) = Probabilidade de ocorrer o evento A

n(A) = Número de elementos de um grupo do todo A

n(S) = Número de elementos do espaço amostral S

**Exemplo**:

No lançamento de um dado, determine a probabilidade de ocorrer:

a) Um número ímpar

Resolução

A = {1,3,5}

S = {1,2,3,4,5,6}

p(A) = (3 / 6) \* 100 = 50%

=====================

Soma de probabilidade (um número par ou um número múltiplo de 3)

P(AUB) = P(A) + P(B) - P(A ( B)

Lançado um dado onde S = {1,2,3,4,5,6}

# #Design Patterns para usar

template\_method , strtegy, decorator

## Reactor Design Pattern

The reactor pattern is used for maintaining non-blocking I/O operations in Node.js. It attaches a callback function (a handler) to each I/O operation. The handler is then submitted to a demultiplexer at the time of the request creation.

Demultiplexer collects every I/O request made in the application and queues them as events in a queue. This queue is what we call the event queue. After queuing the event, the demultiplexer returns the control of the application thread.

Meanwhile, the event loop iterates over each event in the event queue and invokes the attached callback to handle the event response.

This is the reactor pattern used by Node.js.

# #Possíveis traits

#**propício a lesões** => autoexplicativo  
#**firula** => aumenta probabilidade de sofrer faltas próximo a área e penalty.  
#**chega firme** => aumenta probabilidade de fazer faltas e penaltis, além de tomar cartão amarelo e vermelho.  
#**liderança** => o jogador com esse trait, faz com que a equipe ganhe 0.1 de probabilidade de acordo com a posição que ele jogar (defesa, meio-campo, ataque).

# #Endpoints

<http://localhost:3100/match/1>