

Análise e Projeto de Desenvolvimento de Software EAJ6009

Prof. Dr. Josenalde Barbosa de Oliveira

josenalde.oliveira@ufrn.br

Aulas: 35M45

<https://github.com/josenalde/apds>

Itens associados ao projeto:

- Cardinalidade das Associações
- Definição dos tipos dos atributos
- Definição dos Objetos
- Inserção dos Métodos

Jogo de Dados

Visão Geral do Sistema

O sistema consiste de 2 dados e uma quantidade **X** de jogadores informada ao iniciar o jogo. Cada jogador escolhe um valor para apostar, após todos os jogadores informarem sua aposta os dados são lançados. O sistema apresenta o resultado: Se a soma do valor das faces dos dados for igual ao valor de uma das apostas, o sistema informa qual o jogador vencedor, caso nenhum jogador acerte o valor, é informado que o computador venceu.

Jogo de Dados

- Funcionais

- ① Inserir Jogadores
- ② Escolher Valor para Apostar
- ③ Lançar Dados
- ④ Apresentar Resultado
- ⑤ Informar Jogador Vencedor

- Não Funcionais

- ① O máximo de jogadores é 11
- ② O valor escolhido deve estar entre 2 e 12 (Resultados possíveis)
- ③ Um jogador não pode escolher um valor já escolhido por outro

Jogo de Dados

- Caso de Uso: Inserir Jogadores

- ① Sistema solicita a quantidade de jogadores
- ② Informar a quantidade de jogadores
- ③ Sistema solicita um nome para cada jogador
- ④ Cada jogador informa um nome

- Caso de Uso: Escolher valor para apostar

- ① Sistema solicita o valor para aposta de cada jogador
- ② Cada Jogador escolhe um valor entre 2 e 12 para apostar

- Caso de Uso: Lançar dados e Apresentar resultado

- ① Jogador solicita o lançamento dos dados
- ② Sistema lança os dados
- ③ Sistema apresenta o resultado

Jogo de Dados

- Caso de Uso: Informar Vencedor

- ① Sistema verifica se o resultado é igual a um dos valores apostados
- ② Caso seja igual, sistema informa o nome do jogador que venceu
- ③ Caso seja diferente, sistema informa que o computador venceu

Modelagem Conceitual: Classes Candidatas

- Caso de Uso: Inserir Jogadores

- ① Sistema solicita a quantidade de jogadores
- ② Informar a quantidade de jogadores
- ③ Sistema solicita um nome para cada jogador
- ④ Cada jogador informa um nome

- Caso de Uso: Escolher valor para apostar

- ① Sistema solicita o valor para aposta de cada jogador
- ② Cada Jogador escolhe um valor entre 2 e 12 para apostar

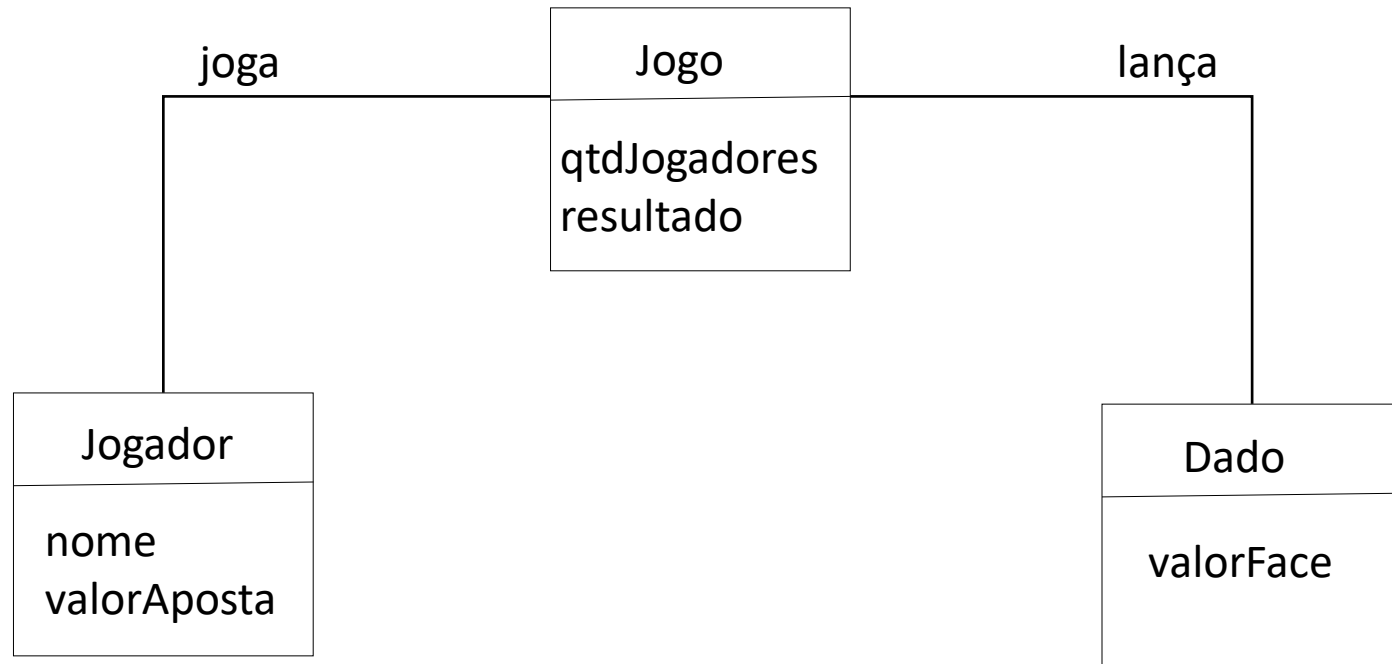
- Caso de Uso: Lançar dados e Apresentar resultado

- ① Jogador solicita o lançamento dos dados
- ② Sistema lança os dados
- ③ Sistema apresenta o valor da face de cada dado

- Caso de Uso: Informar Vencedor

- ① Sistema verifica se o resultado é igual a um dos valores apostados
- ② Caso seja igual, sistema informa o nome do jogador que venceu
- ③ Caso seja diferente, sistema informa que o computador venceu

Modelagem Conceitual: Classes, Atributos e Associações



No projeto

- Acrescentaremos no diagrama de classes:
 - A cardinalidade das associações
 - Os tipos dos atributos
 - Os objetos
 - Os métodos

Cardinalidade (Multiplicidade) das Associações

Quantidade de
elementos de um
conjunto

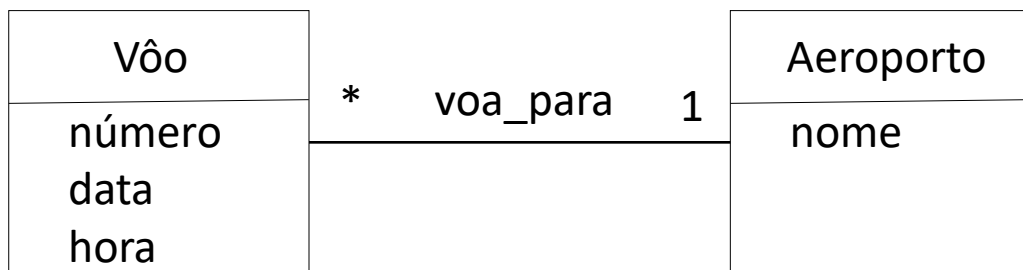
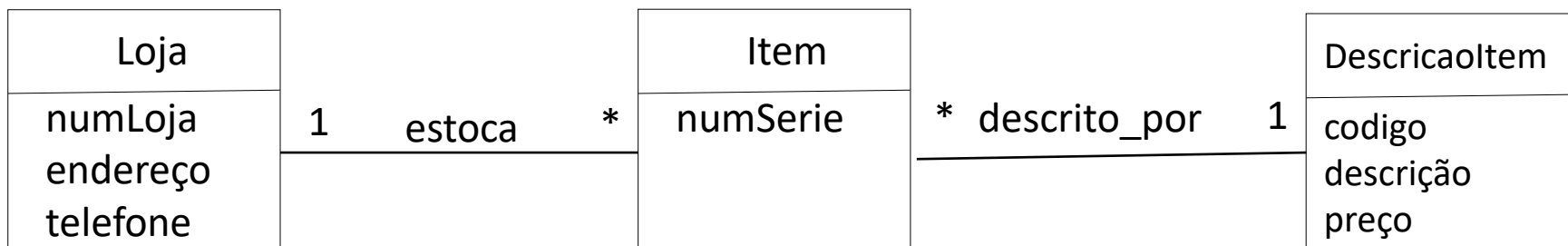


Cardinalidade das Associações

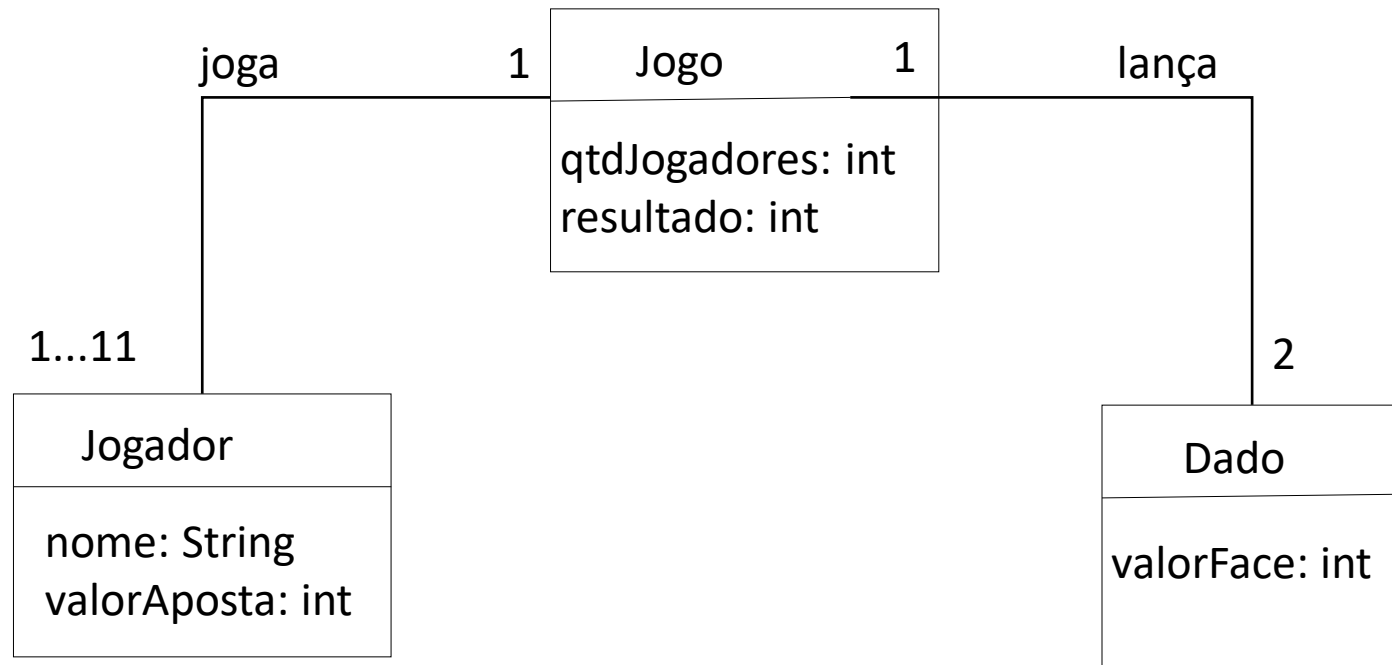
- Informa uma restrição que deve ser considerada na implementação
- Quantas instâncias de uma classe A podem estar associadas a uma instância da classe B

| Símbolo | Significado |
|---------|-----------------------|
| * | Zero ou mais (Muitos) |
| 1..* | Um ou mais |
| 1..40 | Um a 40 |
| 5 | Exatamente 5 |
| 3,5,8 | Exatamente 3, 5 ou 8 |

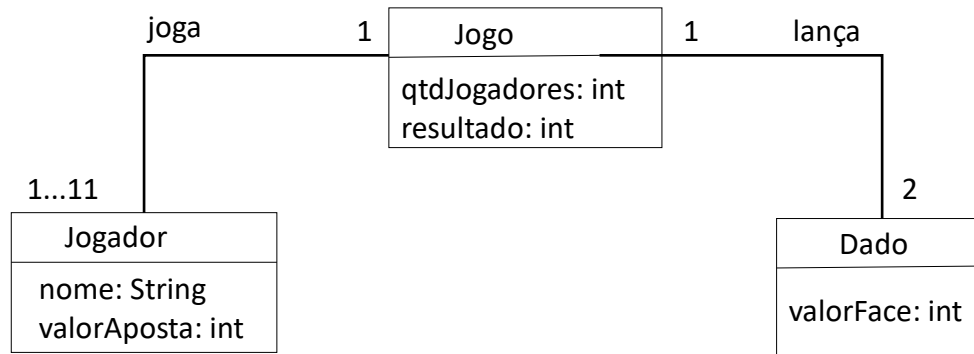
Exemplos de Cardinalidade das Associações



Definir tipo dos atributos



- O projeto começa a modelar detalhes da **implementação**, então surgem algumas perguntas:
 - COMO IMPLEMENTAR AS ASSOCIAÇÕES
 - COMO ASSOCIAR UMA CLASSE A OUTRA CLASSE?
 - COMO AS CLASSES SE COMUNICAM?
- Exemplo:
 - Para que a classe **Jogo** calcule o resultado, ela precisará de informações que estão definidas na classe **Dado** (valorFace) e precisará saber o valor da aposta de cada **Jogador**, bem como de seus nomes
 - Pensando agora na implementação, como isso é feito?



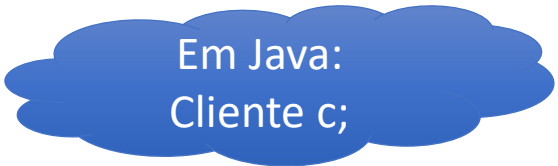
Objetos no Diagrama de Classes

- Para que haja comunicação entre essas classes é necessário definir variáveis que as interliguem
- Como se define uma variável?
 - TIPO NOME;
 - Exemplos em Java
 - **double** preco;
 - **int** quantidade;
 - **String** nome;
- **As variáveis responsáveis por interligar classes, fazendo com que elas se comuniquem, são chamadas de objetos**



Definição de Objetos

- **OBJETOS SÃO AS CLASSES EM EXECUÇÃO**
- Em um sistema, quando precisamos que uma classe A acesse atributos ou métodos de uma classe B, criamos uma variável x cujo tipo é a classe B
- **A variável x é chamada de objeto da classe B**
- **Exemplo:** Para que em uma classe Loja, seja feito o cadastro de um cliente (que é definido por uma classe Cliente), precisamos criar em Loja uma variável cujo tipo é Cliente, essa variável é chamada de objeto da classe cliente



Em Java:
Cliente c;

| Loja |
|------------|
| c: Cliente |

| Cliente |
|---|
| nome: String telefone: int CPF: int |

Diagrama de Objetos

Mostra as classes
em execução

| |
|--------------------------------|
| jogador[0]:Jogador |
| nome = José valorAposta = 1 |

| |
|----------------------------------|
| jogador[1]:Jogador |
| nome = Maria valorAposta = 12 |

| |
|-----------------------------------|
| jogador[2]:Jogador |
| nome = Antônio valorAposta = 5 |

| |
|-----------------------------------|
| execucao: Jogo |
| qtdJogadores = 3 Resultado = 8 |

| |
|---------------|
| dado1:Dado |
| valorFace = 6 |

| |
|---------------|
| dado2:Dado |
| valorFace = 2 |

Classes x Objetos

- **UMA CLASSE DEFINE UM ELEMENTO (CONCEITO) QUE FAZ PARTE DO PROBLEMA (SISTEMA)**
- Exemplo de sistema acadêmico: Aluno, Professor, Disciplina, Turma, Curso (diagrama de classes)
- **UM OBJETO É A EXECUÇÃO DE UMA CLASSE**
- Exemplo: dado1, dado2 (veja diagrama de objetos)
- Comparando com um banco de dados:
 - Classes seriam as tabelas do banco
 - Objetos seriam as instâncias das tabelas
 - No BD cada instância tem um numero (id) para identificá-la, o objeto tem um nome (nome da variável)

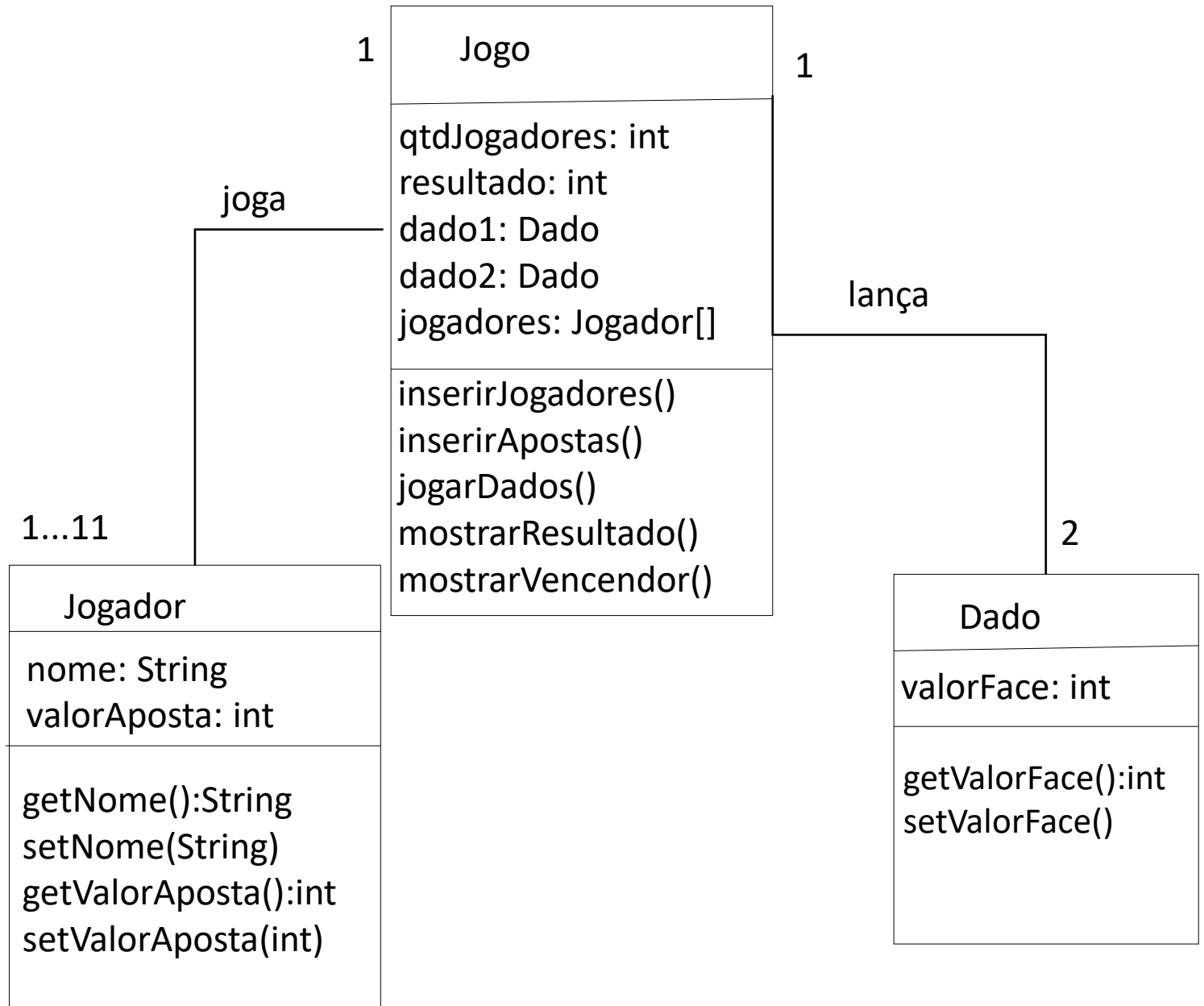
O Que são métodos?

- **Métodos são operações realizadas por uma classe**
- Em programação orientada a objetos os métodos devem ser bem específicos
 - Ou seja, devem resolver apenas a função a que se propõem
 - Isso facilita a manutenção das classes (e consequentemente do sistema)
- No Diagrama de Classes, o terceiro compartimento da caixa de classe mostra a assinatura dos métodos

nomeDoMetodo (lista-de-parâmetros): tipo-de-retorno

Entradas do
Método

Saídas do
Método



Criando Objetos em Java

- **NomeDaClasse** nomeDoObjeto = **new** **NomeDaClasse**();



Operador de Inicialização de Objetos

- Exemplos:
 - Dado dado1 = **new** Dado();
 - **Inicializa um objeto do tipo Dado**
 - Jogador[] jogadores = **new** Jogador[qtdJogadores];
 - **Inicializa um vetor de objetos do tipo Jogador**
 - jogadores[i] = **new** Jogador();
 - **Inicializa cada objeto do vetor de Jogador**

O Que são métodos?

- **Métodos são operações realizadas por uma classe**
- Em programação orientada a objetos os métodos devem ser bem específicos
- Ou seja, devem resolver apenas a função a que se propõem
- Isso facilita a manutenção das classes (e consequentemente do sistema)

Parâmetros e Retorno

- Entrada dos Métodos
 - São valores passados aos métodos, necessários à execução de sua tarefa
- Saída dos Métodos
 - São valores que o método produz como resultado de sua operação
- Operações de Acesso a Atributos (Get e Set)
 - São operações que fazem consulta ou atualização do valor de um atributo
 - Em geral, as classes devem possuir um método GET e um método SET para cada um de seus atributos (globais)
 - GET são métodos de acesso aos valores dos atributos, retornam o valor do atributo (Método de saída)
 - SET são métodos de modificação/atualização do valor do atributo, normalmente recebem o valor para atualização por parâmetro (Método de entrada)

Parâmetros e Retorno

- Entrada dos Métodos
 - São valores passados aos métodos, necessários à execução de sua tarefa
- Saída dos Métodos
 - São valores que o método produz como resultado de sua operação
- Operações de Acesso a Atributos (Get e Set)
 - São operações que fazem consulta ou atualização do valor de um atributo
 - Em geral, as classes devem possuir um método GET e um método SET para cada um de seus atributos (globais)
 - GET são métodos de acesso aos valores dos atributos, retornam o valor do atributo (Método de saída)
 - SET são métodos de modificação/atualização do valor do atributo, normalmente recebem o valor para atualização por parâmetro (Método de entrada)

API JAVA

- Na internet
 - <https://docs.oracle.com/javase/8/docs/api/API>
 - **(Interface de Programação de Aplicativos)**
- Classes Java desenvolvidas para facilitar a construção de aplicativos
- Métodos prontos para ser utilizados em outros programas
- É um conjunto de classes implementadas e também um documento que serve para mostrar ao programador quais são os métodos e os parâmetros necessários para utilização de uma determinada classe

API JAVA

Exemplos de Classes e
Métodos

| | Métodos |
|--------------------|--|
| Classe String | charAt equals equalsIgnoreCase length split toArray |
| Classe Integer | parseInt toString |
| Classe Double | parseDouble toString |
| Classe Math | pow random sqrt |
| Classe JOptionPane | showInputDialog showMessageDialog |

```
package apds.dialog;
import javax.swing.*;
import java.awt.event.*;

/**
 *
 * @author josen
 */
public class Dialog extends WindowAdapter {
    JFrame f;
    Dialog(){
        f=new JFrame();
        f.addWindowListener(this);
        f.setSize(300, 300);
        f.setLayout(null);
        f.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        f.setVisible(true);
    }
    @Override
    public void windowClosing(WindowEvent e) {
        int a=JOptionPane.showConfirmDialog(f,"Are you sure?");
        if(a==JOptionPane.YES_OPTION){
            f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
    }
    public static void main(String[] args) {
        Dialog d = new Dialog();
    }
}
```

Classe Jogador.java

```
package jogoDeDados;
```

```
public class Jogador {
```

```
    String nome;
```

```
    int valorAposta;
```

```
    public String getNome() {  
        return nome;  
    }
```

```
    public void setNome(String nome) {  
        this.nome = nome;  
    }
```

```
    public int getValorAposta() {  
        return valorAposta;  
    }
```

```
    public void setValorAposta(int valorAposta) {  
        this.valorAposta = valorAposta;  
    }
```

```
}
```

Jogador

nome: String
valorAposta: int

getNome():String
setNome(String)
getValorAposta():int
setValorAposta(int)

this.nome serve para
diferenciar a variável
global da variável local

Classe Dado.java

```
package jogoDeDados;  
  
public class Dado {  
    int valorFace;  
  
    public int getValorFace() {  
        return valorFace;  
    }  
  
    public void setValorFace() {  
        valorFace = (int) (1 + Math.random() * 6);  
    }  
}
```

| Dado |
|--------------------------------------|
| valorFace: int |
| getValorFace():int setValorFace() |

Transforma o
resultado double
em inteiro

Classe Jogo.java

```
package jogoDeDados;
```

```
import javax.swing.JOptionPane;
```

```
public class Jogo {
```

```
    int qtdJogadores;
```

```
    int resultado;
```

```
    Dado dado1;
```

```
    Dado dado2;
```

```
    Jogador[] jogadores;
```

```
    public void inserirJogadores() {
```

Inicializa o
vetor de
jogadores

```
        qtdJogadores = Integer.parseInt(JOptionPane  
                                         .showInputDialog("Quantos jogadores irão participar?"));
```

```
        jogadores = new Jogador[qtdJogadores];
```

```
        for (int i = 0; i < qtdJogadores; i++) {
```

```
            jogadores[i] = new Jogador();
```

```
            jogadores[i].setNome(JOptionPane.showInputDialog("Nome do Jogador " + (i + 1)));
```

```
        }
```

```
    }
```

Inicializa cada
objeto do vetor

Jogo

qtdJogadores: int

resultado: int

dado1: Dado

dado2: Dado

jogadores: Jogador[]

inserirJogadores()

inserirApostas()

jogarDados()

mostrarResultado()

mostrarVencedor()

Classe Jogo.java

```
public void inserirApostas() {  
    for (int i = 0; i < qtdJogadores; i++) {  
        jogadores[i].setValorAposta(Integer.parseInt(JOptionPane  
            .showInputDialog(jogadores[i].getNome() + ",qual sua aposta?")));  
    }  
}  
  
public void lancarDados() {  
    dado1 = new Dado();  
    dado2 = new Dado();  
  
    dado1.setValorFace();  
    dado2.setValorFace();  
  
}
```

Classe Jogo.java

```
public void mostrarResultado() {
    resultado = dado1.getValorFace() + dado2.getValorFace();
    JOptionPane.showMessageDialog(null, "Resultado = " + resultado);
}

public void mostrarVencedor() {
    for (int i = 0; i < qtdJogadores; i++) {
        if (jogadores[i].getValorAposta() == resultado) {
            JOptionPane.showMessageDialog(null, jogadores[i].getNome() + "Venceu!");
            break;
        }
        else if (i == qtdJogadores-1)
            JOptionPane.showMessageDialog(null, "O computador venceu!");
    }
}
}
```


Classe Principal.java

```
package jogoDeDados;

public class Principal{

    public static void main(String[] args) {
        Jogo run = new Jogo();
        run.inserirJogadores();
        run.inserirApostas();
        run.lancarDados();
        run.mostrarResultado();
        run.mostrarVencedor();
    }
}
```