



Tecnologia em Análise e Desenvolvimento de Sistema

Programação Orientada a Objetos – POOS3

Laboratório 1





Objetivo

- Trabalhar os conceitos de herança em sistemas orientados a objeto.

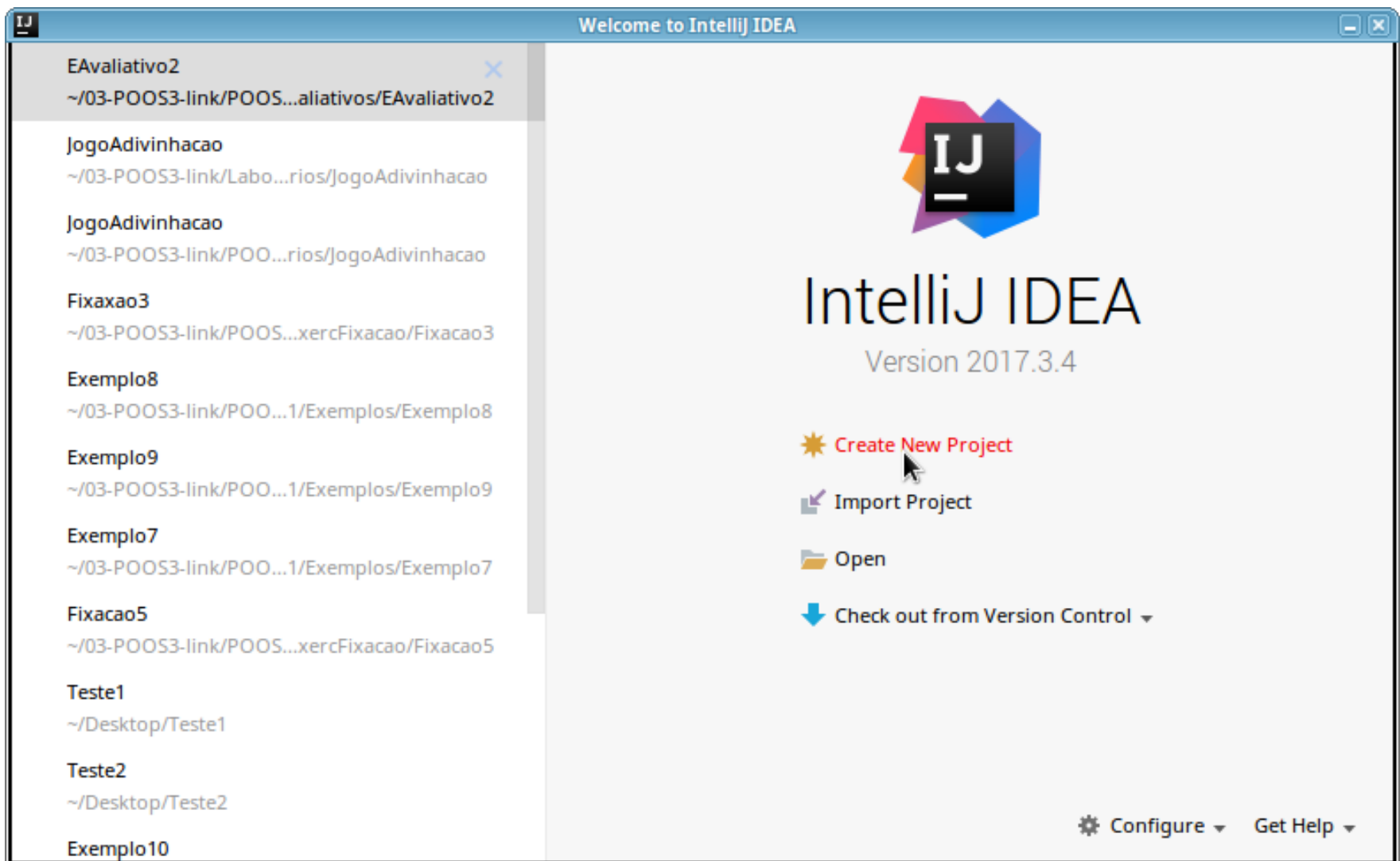


Atividade

- Terceiros desenvolveram um jogo de dados. O jogo consiste em lançar dois dados de seis faces.
- Nosso cliente solicita algumas melhorias no jogo, contudo, não temos acesso ao código fonte implementado, apenas ao arquivo .jar da aplicação. Também temos acesso a documentação do software.
- São as solicitações:
 - Permitir a construção de Dados com qualquer número de faces;
 - Armazenar todos os resultados do lançamento de cada dado;
 - Fazer com que o dado indique se ele está ou não viciado (um número é sorteado mais de 35% das jogadas);
 - Elaborar um programa teste que lance cinco dados 10.000 vezes e apresente um relatório sobre o cada dado.



Criar um novo projeto no IntelliJ





ProjLaboratorio1

New Project

Project name: ProjLaboratorio1

Project location: /home/ednilsonrossi/03-POOS3-link/Laboratorios/Laboratorio_1/ProjLaboratorio1

▼ More Settings

Module name: ProjLaboratorio1

Content root: /home/ednilsonrossi/03-POOS3-link/Laboratorios/Laboratorio_1/ProjLaboratorio1

Module file location: /home/ednilsonrossi/03-POOS3-link/Laboratorios/Laboratorio_1/ProjLaboratorio1

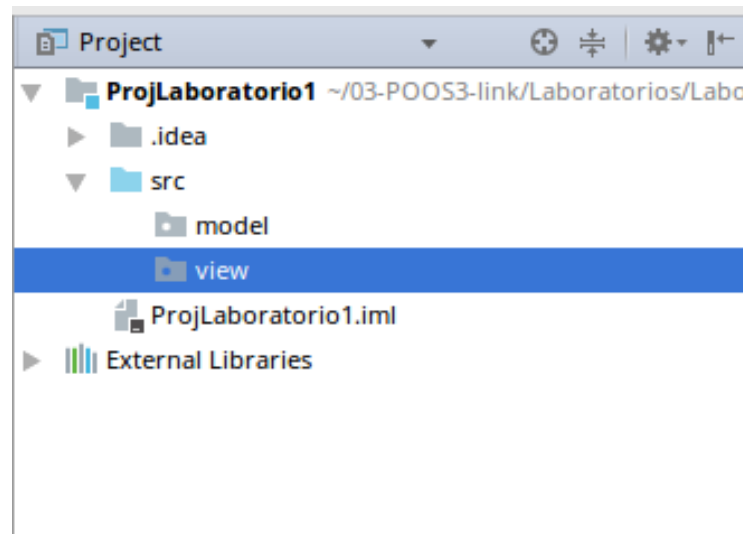
Project format: .idea (directory based)

Previous Finish Cancel Help



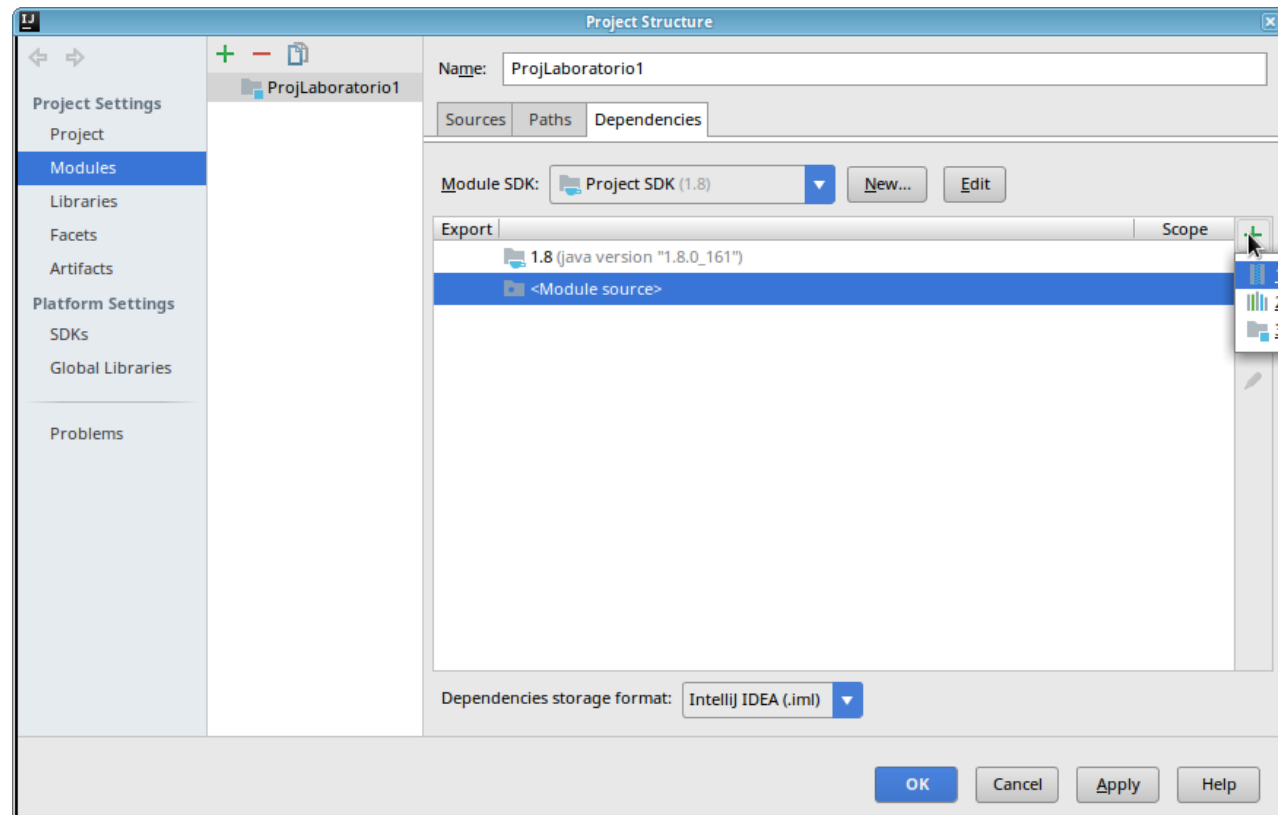
Estrutura de Pacotes

- Crie os pacotes model e view:



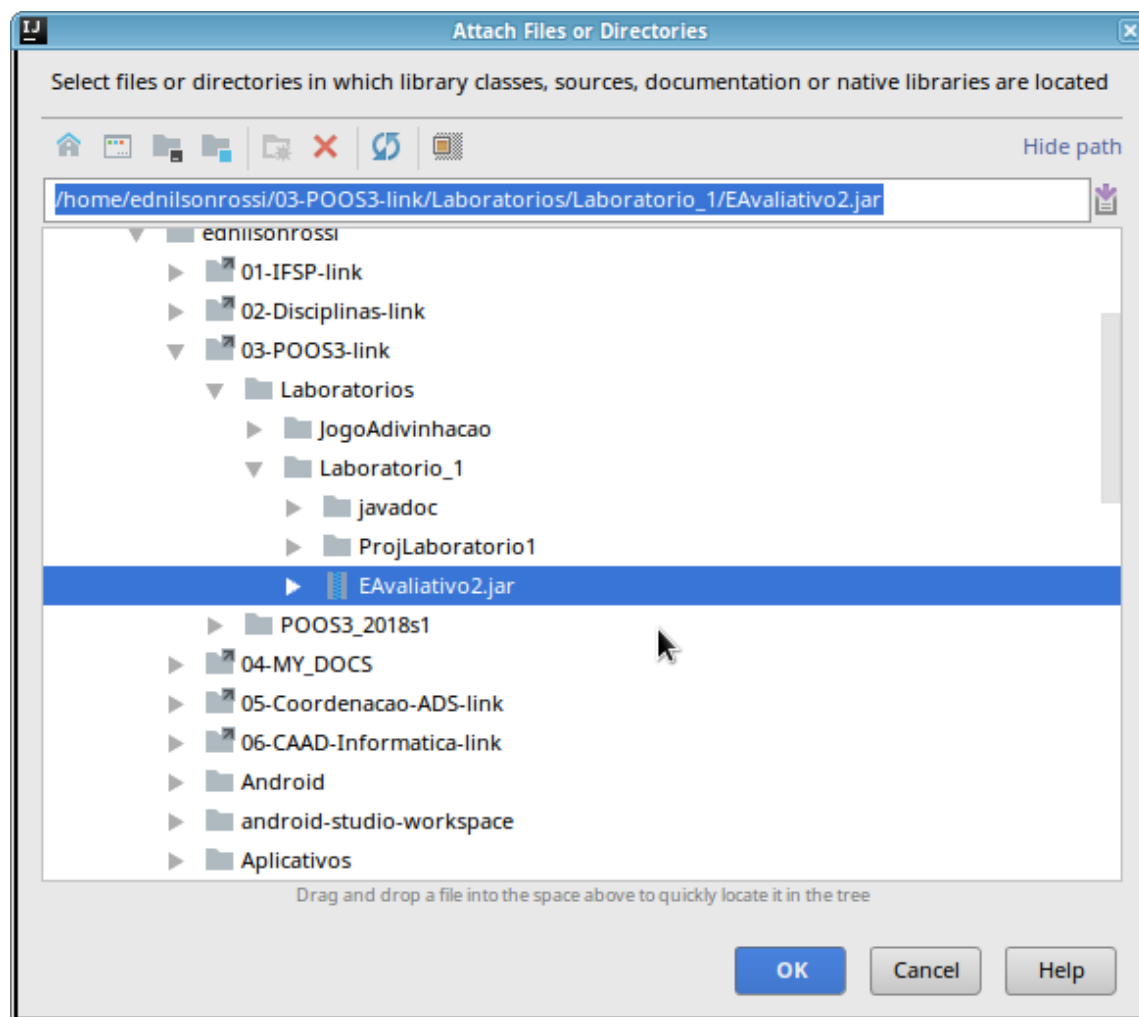
Incluir módulos

- Vamos incluir em nosso projeto o módulo disponibilizado pelo cliente EAvaliativo2.jar.
 - Acesse o menu: File | Project Structure ...
 - Selecione Project Settings | Modules
 - Na aba Dependencies | + JARs ou directories



Incluir módulos

- Selecione o arquivo disponibilizado.
- Clique em OK.

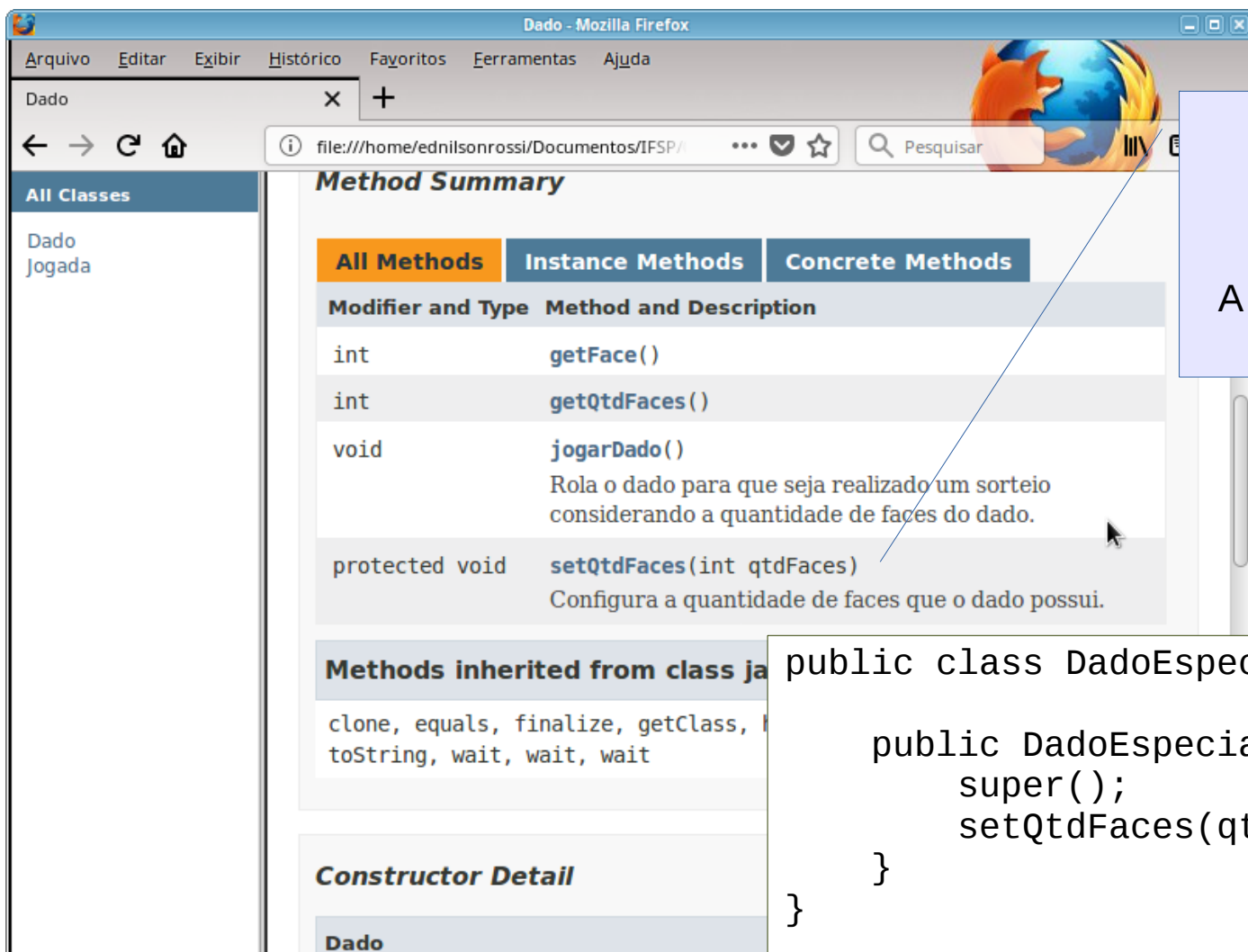




Classe DadoEspecializado

- Crie no pacote “model” a classe DadoEspecializado. Essa classe será a responsável por nossa melhoria no Dado, por isso, ela estende as características e funcionalidades de Dado.

Como criar dados com várias faces?



Method Summary

Modifier and Type	Method and Description
int	<code>getFace()</code>
int	<code>getQtdFaces()</code>
void	<code>jogarDado()</code> Rola o dado para que seja realizado um sorteio considerando a quantidade de faces do dado.
protected void	<code>setQtdFaces(int qtdFaces)</code> Configura a quantidade de faces que o dado possui.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, toString, wait, wait, wait

Constructor Detail

Dado

Documentação da classe Dado indica que existe um método `setQtdFaces()`.

A princípio, basta construir o objeto configurando essa propriedade.

```
public class DadoEspecializado extends Dado {
    public DadoEspecializado(int qtdFaces) {
        super();
        setQtdFaces(qtdFaces);
    }
}
```



Entendendo

```
public class DadoEspecializado extends Dado {  
    public DadoEspecializado(int qtdFaces) {  
        super();  
        setQtdFaces(qtdFaces);  
    }  
}
```

O **super()** “chama” o construtor da classe pai. Como DadoEspecializado é um Dado devemos construir o DadoEspecializado a partir do Dado.

O método `setQtdFaces()` de Dado configura a quantidade de faces do Dado e consequentemente do DadoEspecializado. Só precisamos chamar o método.

Criou-se uma nova classe (DadoEspecializado) que amplia, estende as características e funcionalidades da classe Dado.

A DadoEspecializado **herda** as características e funcionalidades da classe Dado.

A **Herança** é fundamental na programação orientada a objetos. Por meio dela podemos reutilizar componentes já implementados.

Definiu-se um novo construtor, esse recebe como argumento a quantidade de faces que o dado possui.

Existe uma limitação imposta pela classe Dado que “travou” o mínimo de 6 faces para um dado. Como nosso DadoEspecializado é um Dado ele segue a mesma prerrogativa.



Armazenar o resultado de cada jogada do dado

```
public class DadoEspecializado extends Dado {  
    private int sorteios[];  
  
    public DadoEspecializado(int qtdFaces) {  
        super();  
        setQtdFaces(qtdFaces);  
        sorteios = new int[getQtdFaces()];  
        zerarSorteios();  
    }  
  
    private void zerarSorteios(){  
        for(int i = 0; i < sorteios.length; i++)  
            sorteios[i] = 0;  
    }  
}
```

Faremos o armazenamento em um array do tipo inteiro.

Observe que DadoEspecializado possui os atributos de Dado acrescido do array.

Ao construir um DadoEspecializado define-se o tamanho do vetor. Por exemplo um dado com 20 faces tem um array de 20 posições (0 a 19).

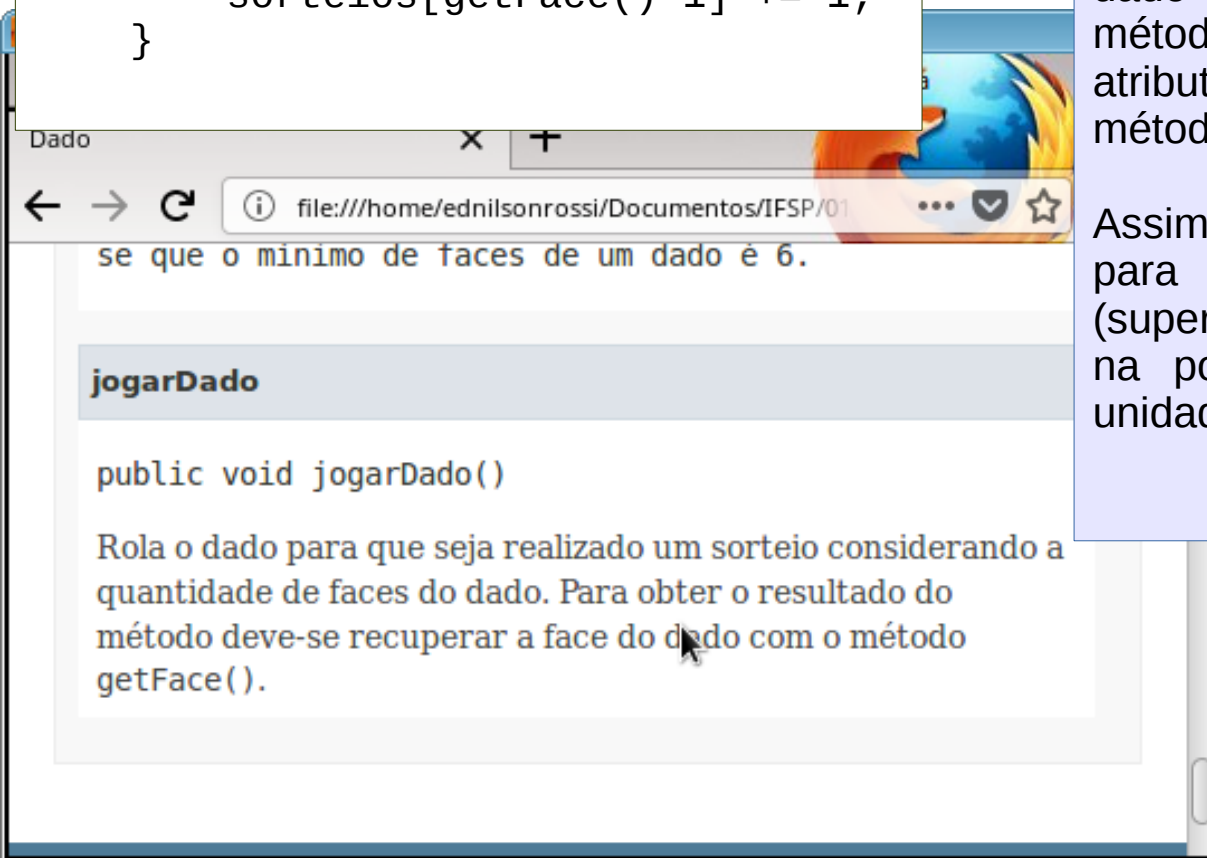
Como será realizado o incremento a cada jogada do dado inicia-se o array com zero em todas as posições.

Armazenar o resultado de cada jogada

```
@Override
public void jogarDado() {
    super.jogarDado();
    sorteios[getFace()-1] += 1;
}
```

A classe Dado possui um método jogarDado() que, conforme documentação, sorteia a face do dado que será exibida. Observa-se que esse método armazena o resultado em algum atributo, pois o resultado é obtido a partir do método getFace().

Assim, reescreveu-se o método jogarDado() para que a cada jogarDado() do pai (super.jogarDado()) seja incrementado no array, na posição referente a face sorteada, uma unidade, indicando um sorteio.





Auditar o Dado

```
private void auditarResultados(){
    viciado = false;
    double porcentagem=0;
    for(int i=0; i < sorteios.length; i++){
        porcentagem = (sorteios[i] * 100)/qtdadeLances;
        if(porcentagem >= 35.0)
            viciado = true;
    }
}
```

Definiu-se a propriedade **viciado** como um boolean.

A cada jogarDado() executado é realizada a auditoria que verifica se houve algum dado que sorteou uma face mais de 35% dos lances, assim, observa-se outro novo atributo **qtdadeLances**.

```
package model;

public class DadoEspecializado extends Dado {

    private int sorteios[];
    private int qtdadeLances;
    private boolean viciado;

    public DadoEspecializado(int qtdFaces) {
        super();
        setQtdFaces(qtdFaces);
        sorteios = new int[getQtdFaces()];
        zerarSorteios();
        qtdadeLances = 0;
        viciado = false;
    }

    private void zerarSorteios(){
        for(int i = 0; i < sorteios.length; i++)
            sorteios[i] = 0;
    }

    @Override
    public void jogarDado() {
        super.jogarDado();
        sorteios[getFace()-1] += 1;
        qtdadeLances += 1;
        auditarResultados();
    }

    private void auditarResultados(){
        viciado = false;
        double porcentagem=0;
        for(int i=0; i < sorteios.length; i++){
            porcentagem = (sorteios[i] * 100)/qtdadeLances;
            if(porcentagem >= 35.0)
                viciado = true;
        }
    }
}
```



```
public int getQtdadeSorteios(int face){  
    int retorno = -1;  
    if(face > 0 && face <= sorteios.length){  
        retorno = sorteios[face-1];  
    }  
    return retorno;  
}  
  
public int getQtdadeLances() {  
    return qtdadeLances;  
}  
  
public boolean isViciado() {  
    return viciado;  
}  
}
```




```
public class Main {

    private static final int JOGADAS = 100;
    private static final int QTD_DADOS = 5;

    public static void main(String[] args) {
        int contador;
        DadoEspecializado dados[] = new DadoEspecializado[QTD_DADOS];

        for(int i=0; i<QTD_DADOS; i++){
            dados[i] = new DadoEspecializado(6*(i+1));
        }

        for(contador=0; contador < JOGADAS; contador++){
            for (int i=0; i < QTD_DADOS; i++){
                dados[i].jogarDado();
            }
        }

        System.out.println("RELATÓRIO DE JOGADAS DOS DADOS");
        for(int i=0; i < QTD_DADOS; i++){
            System.out.println("\n\nDado " + (i+1));
            for(int j=0; j<dados[i].getQtdFaces(); j++){
                System.out.println("Face " + (j+1) + ": " + dados[i].getQtdadeSorteios(j+1));
            }
            System.out.println("Total de lances: " + dados[i].getQtdadeLances());
            if(dados[i].isViciado()){
                System.out.println("Dado é viciado");
            }else{
                System.out.println("Dado não é viciado");
            }
        }
    }
}
```