



**INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA**
RIO GRANDE DO SUL
Câmpus Feliz

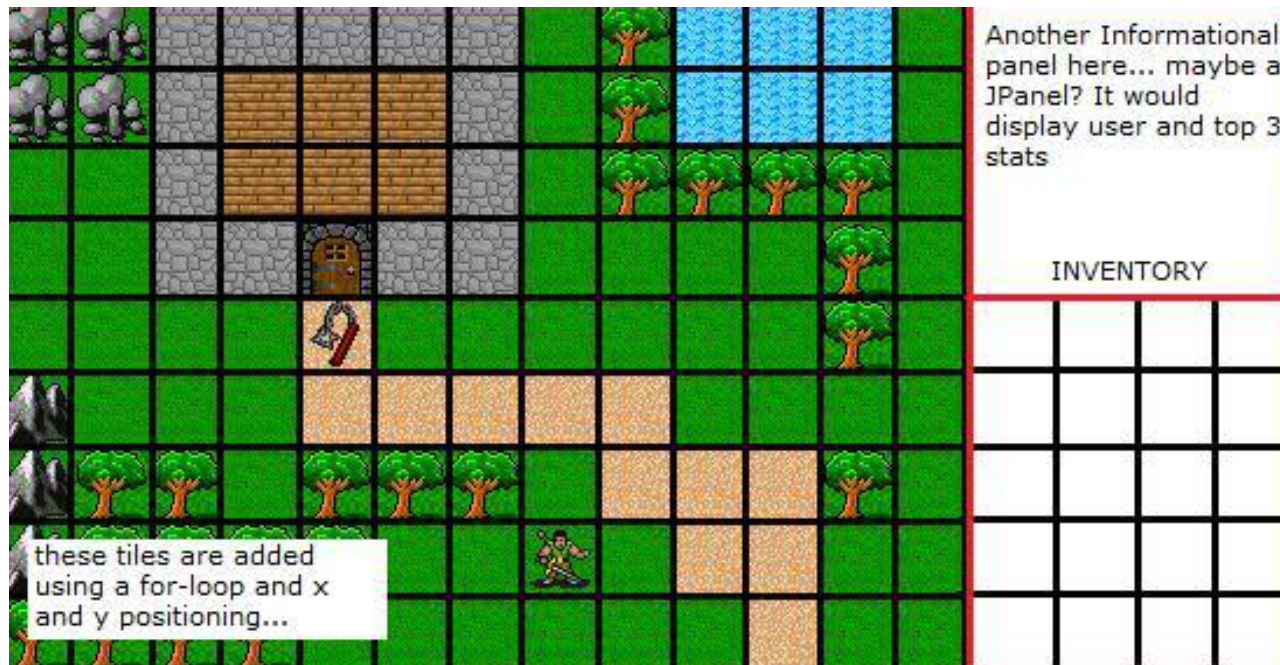
Interfaces

Prof. Moser Fagundes

Programação II

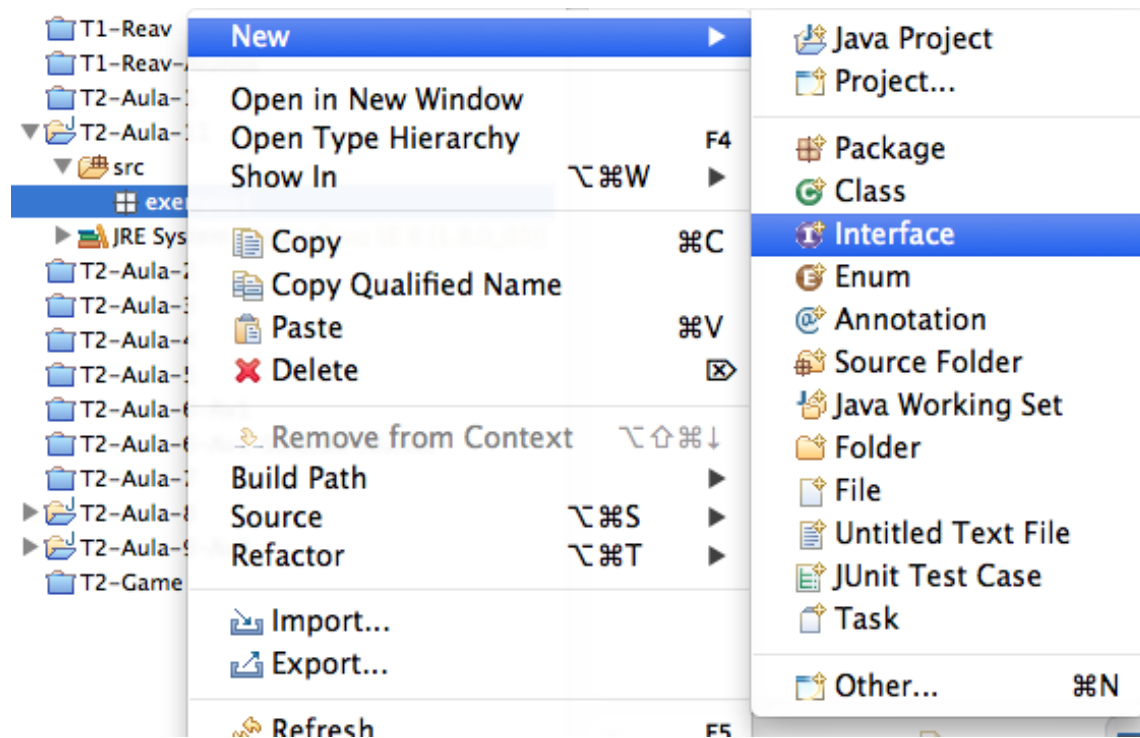
Interfaces

As **interfaces** que vamos estudar hoje não são interfaces gráficas do usuário, mas sim recursos do Java para obrigar que classes tenham um determinado comportamento.



Interfaces

- Uma **interface** em Java é uma especie de classe, porém com algumas diferenças.
- Para criar uma interface usamos a palavra-chave **interface** no lugar da palavra-chave **class**.



Interfaces

- Uma **interface** em Java é uma espécie de classe, porém com algumas diferenças.
- Para criar uma interface usamos a palavra-chave **interface** no lugar da palavra-chave **class**.

```
package exemplo1;
```

```
public interface IPlayer {
```

```
    public abstract void iniciar();  
    public abstract void mover(int x, int y, int velocidade);  
    public abstract void apontar(int x, int y);  
    public abstract void atirar();  
    public abstract void sair();
```

```
}
```

Interfaces

Uma interface possui apenas **assinaturas de métodos**, públicos e abstratos (sem corpo). **Não** pode possuir métodos concretos (com corpo), nem métodos estáticos.

Os prefixos **abstract** e **public** podem ser omitidos.

```
package exemplo1;
```

```
public interface IPlayer {
```

```
    public abstract void iniciar();
```

```
    public abstract void mover(int x, int y, int velocidade);
```

```
    public abstract void apontar(int x, int y);
```

```
    public abstract void atirar();
```

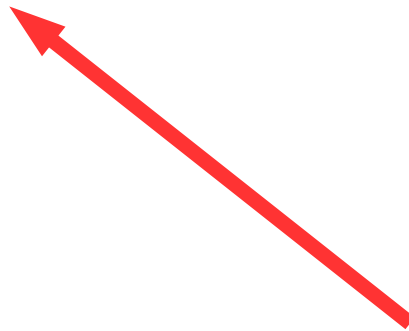
```
    public abstract void sair();
```

```
}
```

Interfaces

Uma interface não é instanciável, ou seja, não podemos criar objetos de uma interface usando operador *new*.

```
IPlayer exemplo1 = new IPlayer();
```



ERRO

Interfaces

Uma interface não pode conter variáveis de instância.

```
package exemplo1;
```

```
public interface IPlayer {
```

```
    private int x;
```

```
    private int y;
```

```
    public abstract void iniciar();
```

```
    public abstract void mover(int x, int y, int velocidade);
```

```
    public abstract void apontar(int x, int y);
```

```
    public abstract void atirar();
```

```
    public abstract void sair();
```

```
}
```



ERRO

Interfaces

Uma interface **pode** conter constantes.

```
package exemplo1;
```

```
public interface IPlayer {
```

```
    public static final int POS_INICIAL_X = 200;
```

```
    public static final int POS_INICIAL_Y = 300;
```

```
    public abstract void iniciar();
```

```
    public abstract void mover(int x, int y, int velocidade);
```

```
    public abstract void apontar(int x, int y);
```

```
    public abstract void atirar();
```

```
    public abstract void sair();
```

```
}
```

OK



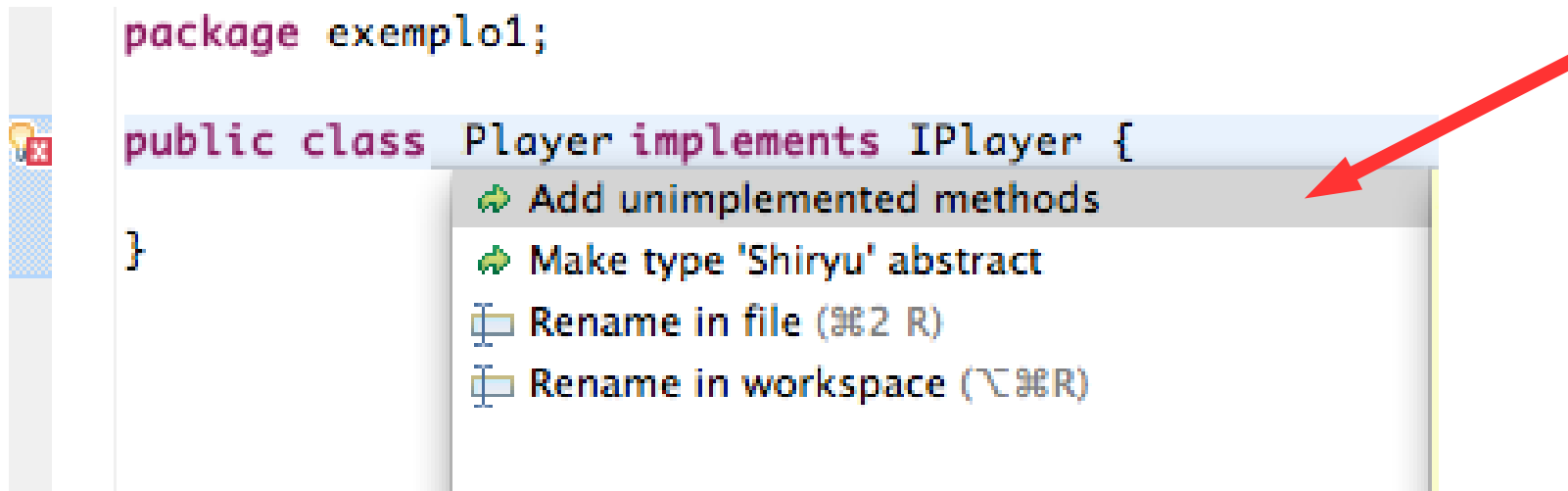
Usando Interfaces

Para usar uma interface, usamos a palavra-chave **implements** na declaração da classe que vai implementar a interface.

```
package exemplo1;

public class Player implements IPlayer {
}


```



The screenshot shows an IDE window with a code editor. The code defines a package `exemplo1` and a public class `Player` that implements the `IPlayer` interface. The line `public class Player implements IPlayer {` is highlighted. A context menu is open over this line, showing options: `Add unimplemented methods`, `Make type 'Shiryu' abstract`, `Rename in file (⌘2 R)`, and `Rename in workspace (⌘⌘R)`. A red arrow points to the `implements` keyword in the code.

Usando Interfaces

Nas classes que implementam a interface, devemos criar o corpo dos métodos que especificamos na interface.

```
package exemplo1;
```

```
public class Player implements IPlayer {
```

```
    public void iniciar() { }
```

```
    public void mover(int x, int y, int velocidade) { }
```

```
    public void apontar(int x, int y) { }
```

```
    public void atirar() { }
```

```
    public void sair() { }
```

```
}
```

Classe Abstrata X Interface

- Uma **classe abstrata** pode conter métodos completos e abstratos, e uma **interface** pode conter apenas assinaturas de métodos.
- Uma **classe abstrata** pode conter atributos e construtores. Uma **interface** não pode conter atributos nem construtores, somente constantes.
- Uma **classe abstrata** não suporta herança múltipla, porém uma classe pode implementar várias **interfaces**.

Pergunta

Em que situações seria útil usar uma interface?