Introdução à Programação

Construções Básicas de Java

Programa em Java

```
import br.ufpe.cin.miniJava.gui.Button;
import br.ufpe.cin.miniJava.gui.Window;

public class MinhaPrimeiraClasse {

    public static void main(String[] args) {
        Window ianela ;
        Button botao  ;
        String textoBotao = “Fechar”;
        janela = new Window ("Minha Janela");
        botao = new Button();
        janela.include(botao,50,90);
        botao.setText(textoBotao);
        janela.setVisible ( true );
    }
}
```

Palavras Reservadas

E ainda chamada às APIs ...

Identificadores criados pelo programador

Tópicos da Aula

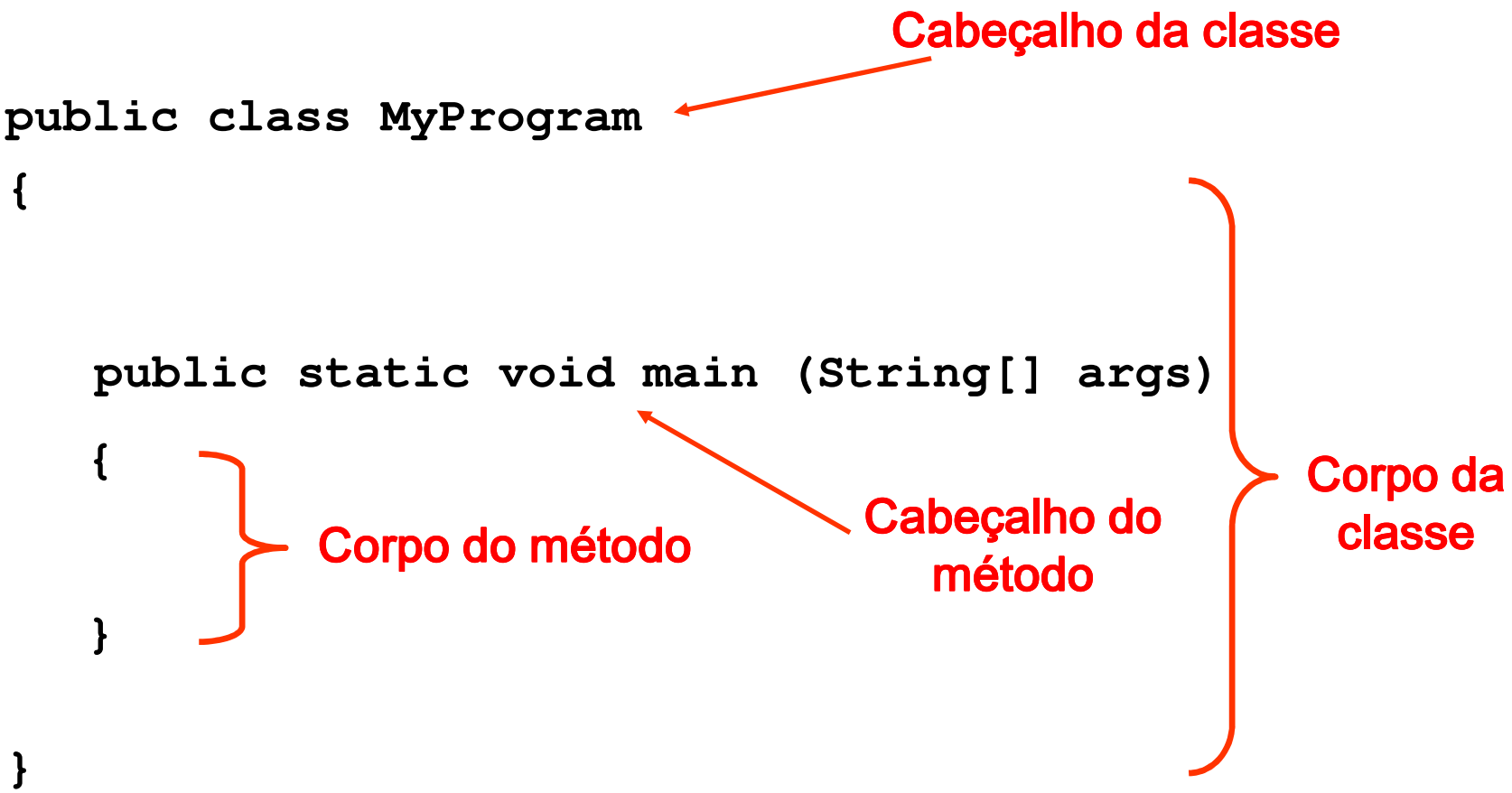
- Hoje aprenderemos a escrever um programa em Java, para isto veremos
  - Estrutura de um programa
  - Estruturas básicas de uma linguagem de programação
    - Identificadores,Variáveis,Tipos, Atribuição e Expressões
  - Criação de objetos
  - Chamada de métodos
  - Classe String
  - Importação de classes
  - Erros (estáticos) possíveis
- Depois escreveremos um programa em um ambiente de programação
  - Utilizando a API gráfica
  - Executando um programa

Estrutura de um Programa

- Como Java é uma linguagem OO, a sua construção fundamental é a **classe**:
  - Um programa é constituído de uma ou mais classes
  - Uma classe contém um ou mais atributos e métodos
  - Um método é composto de instruções

- Uma aplicação (programa) Java deve ter pelo menos uma classe e esta classe deve ter um método chamado ***main***
  - Nem toda classe deve ter o método *main*
  - O programa inicia sua execução neste método

Estrutura de um Programa



Identificadores

- Identificadores são palavras que o programador utiliza em programas
- Em Java, um identificador é formado por um conjunto de letras, dígitos, o caractere *underscore* "_", e o caractere dollar "$"
  - Identificadores não podem começar com um dígito
    - Exemplo de identificadores **válidos**: _a, a3_,bom_dia
    - Exemplo de identificadores **inválidos**: 2a, a-b, a_ b
  - Java é "case-sensitive"
    - Os identificadores ***casa*** e ***CASA*** são diferentes

Identificadores

- Identificadores podem ser:
  - Nomes que o programador escolheu
    - Exemplo: nome de uma classe
  - Nomes que terceiros escolheram
    - Exemplo: nome de um método de uma API
  - Palavras reservadas da linguagem
    - Não podem ser usadas de outra forma
    - Exemplo: *class, static, public*, etc

Identificadores em Java

```
import br.ufpe.cin.miniJava.gui.Button;
import br.ufpe.cin.miniJava.gui.Window;

public class MinhaPrimeiraClasse {

    public static void main(String[] args) {
        Window ianela ;
        Button botao  ;
        String textoBotao = “Fechar”;
        janela = new Window ("Minha Janela");
        botao = new Button();
        janela.include(botao,50,90);
        botao.setText(textoBotao);
        janela.setVisible ( true );
    }
}
```

Palavras Reservadas

Chamada às APIs

...

Identificadores criados pelo programador

Variáveis

- Programas manipulam dados e esses dados são armazenados em ***variáveis***
- Uma variável é uma posição na memória referenciada por um identificador (nome)
- Uma variável deve ser ***declarada*** informando o tipo de dado que ela armazenará e o nome dela


```
int total;

int count, temp, result;
```

Muitas variáveis podem ser criadas em uma declaração

Variáveis

- Uma ***declaração*** de variável instrui o compilador/interpretador:
  - a reservar um espaço de memória suficiente para armazenar o tipo de dado declarado
  - o nome ao qual iremos referenciar esta posição de memória
- Só após a declaração da variável, é que esta pode ser referenciada
- Quando uma variável é referenciada no programa, o valor armazenado nela é utilizado

Atribuição

- Um ***comando*** de ***atribuição*** modifica o valor armazenado na variável
- O operador de atribuição é o sinal de **=**

```
total = 55;
```

Variável *total* armazena valor 55

```
total = 65;
```

Valor 65 sobrescreve o valor armazenado antes

Só se pode atribuir a uma variável valores compatíveis com o tipo declarado da variável

Inicialização de Variáveis

- Uma variável pode ser inicializada com o comando de atribuição na hora de sua declaração

Declara variável *total* do tipo int

```
int total = 55;
```

Inicializa *total* com valor 55

- Caso variáveis não sejam inicializadas, dependendo de onde a variável for declarada **(não vale para variáveis declaradas em métodos)** , Java atribui valores padrões que dependem do tipo da variável
  - "0" para tipos numéricos
  - "false" para *boolean*
  - null para referências a objetos

Veremos em breve!

Constantes

- Uma ***constante*** é um identificador semelhante a uma variável, exceto pelo fato de só poder armazenar o mesmo valor durante toda sua existência
- Uma constante é declarada usando a palavra reservada ***final***
- Deve-se inicializar a constante no ato da sua declaração

```
final float PI = 3.1416;
PI = 3.141618;
```

Esse comando gera um erro de compilação

Não se pode mudar o valor de uma constante

Constantes

- São úteis para dar um significado mais compreensível a determinados valores
  - Exemplo : LIMITE_VAGAS é mais compreensível que o valor 2546
- Facilitam a manutenção do programa
  - Caso uma alteração no programa seja necessária que acarrete uma mudança no valor da constante e esta constante seja referenciada em vários lugares do programa, só precisamos alterar o programa em um lugar
- Explicitam formalmente que um determinado valor não pode ser alterado
  - Evitam erros de outros programadores

Tipos em Java podem ser...

- Primitivos
  - 4 representando números inteiros
    - byte, short, int, long
  - 2 representando números reais
    - float, double
  - 1 representando caracteres
    - char
  - 1 representando valores *booleanos* (false, true)
    - boolean
- Referências
  - Referenciam objetos das classes existentes

Tipos Primitivos Numéricos

- A diferença entre os vários tipos numéricos é o tamanho que cada um ocupa em memória
- A diferença de tamanho reflete nos valores que um tipo pode ter

Tipo	Tamanho	Valor mínimo	Valor máximo
`byte`	8 bits	-128	127
`short`	16 bits	-32768	32767
`int`	32 bits	-2147483648	2147483647
`long`	64 bits	$< -9 \times 10^{18}$	$> 9 \times 10^{18}$
`float`	32 bits	$+/- 3.4 \times 10^{38}$ com 7 dígitos significativos	
`double`	64 bits	$+/- 1.7 \times 10^{308}$ com 15 dígitos significativos	

Tipo Caractere

- Uma variável do tipo *char* armazena um único caractere
- Um caractere é delimitado por aspas simples
  - Exemplos de atribuição
    - char a = ‘A’;
    - char b = ‘g’;
    - char c =‘\n’;
    - char d = ‘,’;
- Uma variável do tipo *char* ocupa 16 bits

Tabela de Caracteres

- Uma tabela de caracteres ou *character set* é uma lista ordenada de caracteres, onde cada caractere corresponde a um único número
- Uma variável *char* em Java pode armazenar qualquer caractere pertencente a tabela *Unicode*
- *Unicode* usa 16 bits para representar cada caractere, permitindo assim 65536 caracteres
- Esta tabela inclui caracteres que fazem parte de muitas línguas espalhadas pelo mundo
- A tabela *ASCII* é uma tabela mais antiga e muito popular, e é um subconjunto da *Unicode*

Expressões

- Uma **expressão** é uma combinação de um ou mais operadores e operandos que geralmente realiza um cálculo
- A **avaliação** ou cálculo da expressão se faz obedecendo regras de associação e precedência estabelecidas na linguagem

```
int total = 3 + 4/2 ;
```

Expressão é avaliada e o resultado é atribuído a *total* que armazena agora o valor 5

Divisão (/) tem precedência sobre soma (+)

- O valor calculado pode não ser necessariamente um número
  - Pode ser um booleano, caractere, Objeto

Expressões

- Operandos podem ser:
  - Valores (literais), variáveis, constantes ou outra fonte de dado
- Operadores podem ser:
  - Aritméticos
    - +, -, *, /,% (Resto da divisão)
  - Relacionais
    - >,<,>=,<=,==,!=
  - Lógicos
    - &&, ||, !

Criação de Objetos

- Em linguagens OO, escrevemos programas utilizando objetos
- Mas, primeiro, temos que **criá-los**...
- Um objeto em Java é criado através do operador ***new***

```
new Button();
```

Palavra reservada ordenando a criação de um objeto

Nome da classe do objeto a ser criado

Avaliação do operador `new`

Para avaliar uma expressão do tipo

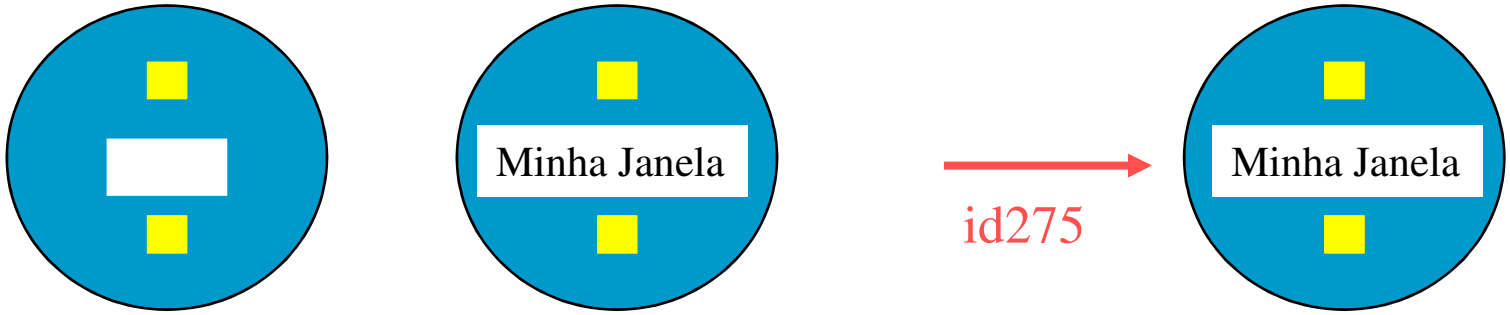
```
new NomeDaClasse();
```

o computador...

- Cria um objeto da classe **`NomeDaClasse`** e armazena na sua memória
- Inicializa os atributos deste objeto usando o construtor desta classe (em Java, o construtor deve ter o mesmo nome da classe)
- Devolve como resultado da avaliação uma referência (endereço da memória) para o objeto criado

Criando um objeto...

```
new Window("Minha Janela");
```



1. Cria o objeto na memória

2. Inicializa o objeto

3. Associa uma referência ao objeto

4. Devolve a referência como resultado da avaliação

Construtores e `new`

- Podemos ter mais de um construtor para um determinado objeto
- A escolha do construtor para inicializar os atributos é determinada pela lista de **argumentos**, entre parênteses

```
new NomeDaClasse(argumentos)
```

O número, ordem e tipos dos argumentos determina o construtor

Parâmetros e argumentos

- Os **parâmetros** são nomes que aparecem na **declaração** do construtor ou método

```
Window(String title)
```

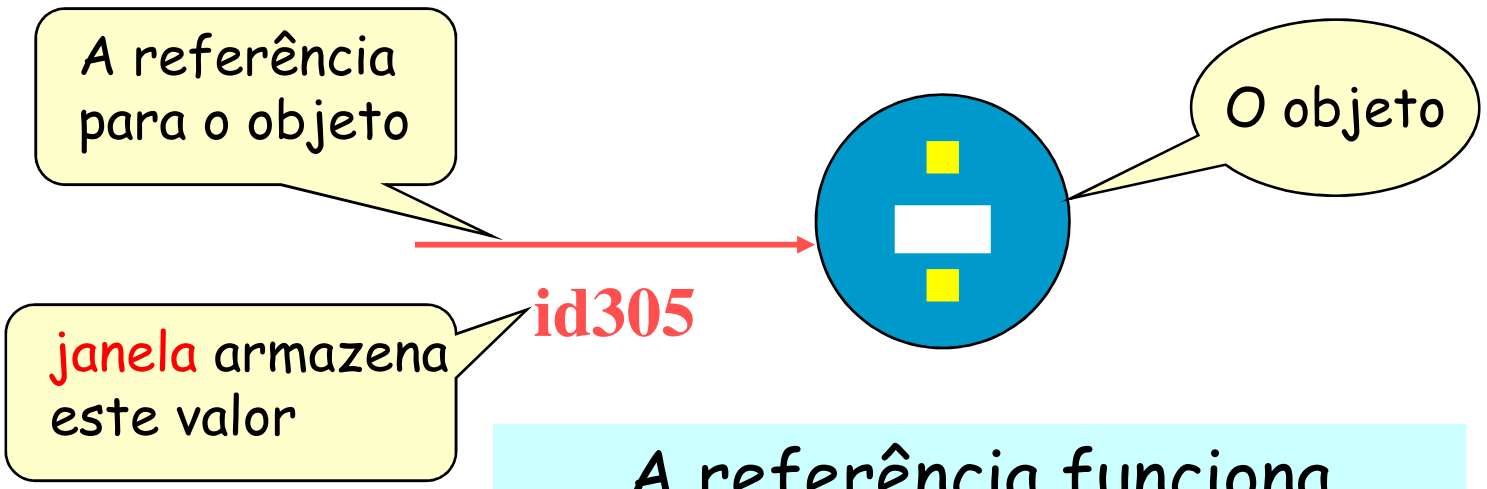
- Os **argumentos** são expressões que aparecem na expressão de criação, na **invocação** do construtor ou método

```
new Window("Minha Janela");
```

Referências para objetos

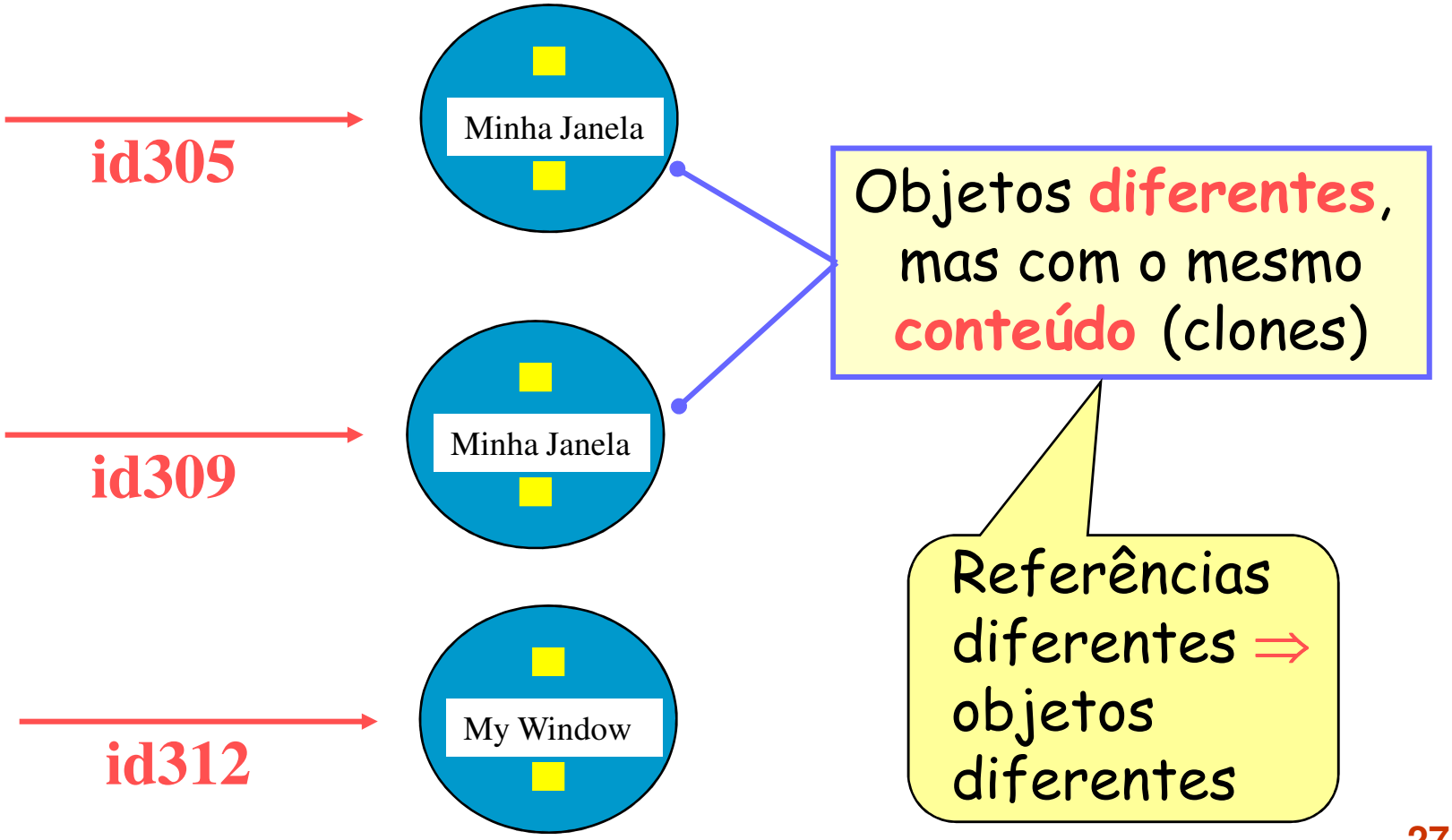
- Todo acesso e manipulação é feito indiretamente, através de uma **referência** para o objeto
- Quando declaramos uma variável do tipo de um determinado objeto, ela não armazena o objeto e sim uma referência para um objeto daquele tipo

```
Window janela = new Window("Janela");
```



A referência funciona como a identidade do objeto

Referências e identidade de um objeto



Considerações sobre Criação de Objetos

- Os valores dos atributos (estado) de um objeto podem mudar durante a execução de um programa, mas isso não afeta a identidade do objeto
- Uma variável que guarda uma referência para um objeto pode ser inicializada com o valor ***null***
  - Indica que a variável não referencia nenhum objeto
  - O valor ***null*** pode ser explicitamente atribuído a uma variável
- O ato de criar um objeto é chamado de **instanciar** um objeto de uma determinada classe

Os elementos de um tipo primitivo são **valores primitivos**, enquanto os elementos de um tipo referência são **referências** para objetos!

Considerações sobre Criação de Objetos

Por que um objeto não é criado no ato de declaração da variável que armazena a referência para o objeto?????

- Pode se ter um processamento dispendioso, antes mesmo de se começar qualquer processamento útil
- Em alguns casos, duas referências devem estar ligadas ao mesmo objeto. Não faria sentido, portanto, se criar dois objetos idênticos

Comunicação entre objetos

- Os objetos se comunicam para realizar serviços
- A comunicação é feita através da troca de mensagens ou chamada de métodos
- Cada mensagem é uma requisição para que um objeto execute um método específico

Executando métodos

O operador ponto (.) deve ser utilizado para invocar o método

```
janela.include(botao,50,90);
```

Variável contendo uma referência para o objeto no qual o método será executado

Nome do método a ser executado

Argumentos do método

A ordem é importante!

Efeito da execução de um método

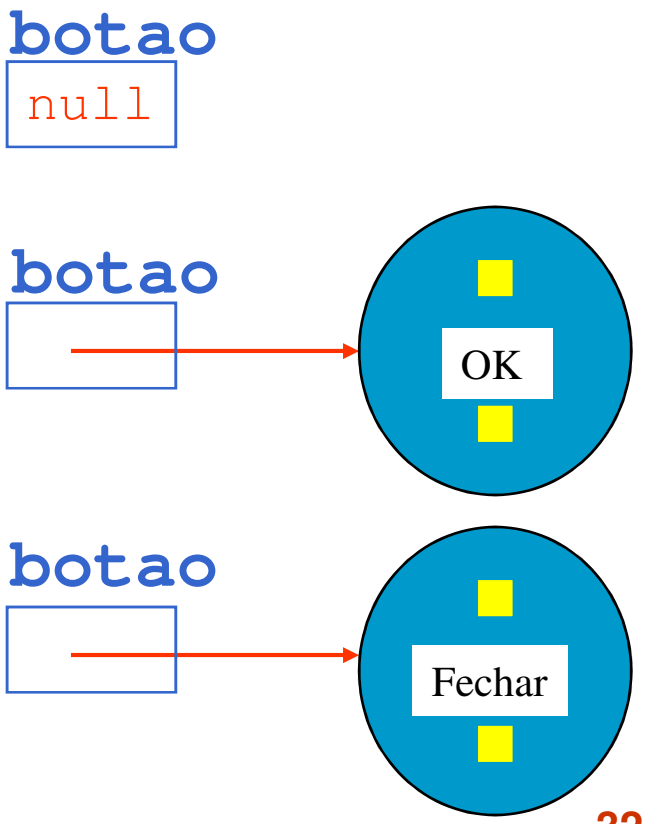
Programa

```
Button botao = null;

botao = new Button();

botao.setText("Fechar");
```

Efeito da execução do programa



Classe String

- Java vem com uma API disponibilizando milhares de classes para o programador
- Uma das classes que é muito utilizada é a ***String***
- Como utilizamos String freqüentemente, para criar um objeto do tipo String não precisamos utilizar o operador *new*
  - Um objeto String pode ser criado de 2 formas:

```
String texto = new String("Fechar");
```

ou

Esta sintaxe só vale para String

```
String texto = "Fechar";
```

- Um valor do tipo String deve ser colocado entre aspas duplas (" ")

Métodos de String

- Uma vez que um objeto String é criado, seu valor e tamanho não podem ser modificados
  - Por isso que se diz que uma String é imutável
- Entretanto, os vários métodos oferecidos por String retornam novos objetos String que são versões modificadas do objeto String original
- Os métodos da classe String podem ser visualizados aqui

Índices de String

- Às vezes, precisamos examinar um determinado caractere de uma String para realizar algum processamento
- Isto pode ser feito especificando o índice numérico do caractere na String
- O caractere mais à esquerda de uma String tem índice 0 em cada String e o caractere mais à direita tem índice igual ao tamanho da String - 1

Importação de Classes

- Para aumentar produtividade, fazemos constante reuso de classes já existentes
- Para reutilizar classes feitas por terceiros ou que estão em pacotes diferentes, devemos importar estas classes
- Podemos fazer isto escrevendo antes da definição da classe uma declaração de importação:

```
import br.ufpe.cin.miniJava.gui.Button;
```

Palavra reservada para importar classes

Nome do pacote

Nome da classe

Importação de Classes

- Utilizando a palavra reservada ***import***, podemos referenciar a classe no programa sem informar o pacote onde a classe está

```
import br.ufpe.cin.miniJava.gui.Button;
public class MinhaPrimeira Classe{
...
Button botao;
```

- Sem utilizar ***import***, ainda podemos utilizar a classe, desde que coloquemos o nome completo da classe (incluindo o pacote)

```
 public class MinhaPrimeira Classe{
...
br.ufpe.cin.miniJava.gui.Button botao;
```

Erros de Programação

- Um programa pode ter 3 tipos de erros
  - Erros estáticos
    - Erros verificados em tempo de compilação
  - Erros dinâmicos
    - Erros que acontecem durante a execução do programa que causam o fim de execução de um programa de forma anormal
  - Erros de lógica
    - O programa executa sem problemas, mas produz resultados errados

Hoje, estudaremos erros de programação estáticos

Erros de Programação Estáticos

- Um programa pode dar erro de compilação quando:
  - Há erros de sintaxe
    - Uso de identificadores inválidos
    - Emprego errado de palavras reservadas e símbolos da linguagem
  - Classes não foram importadas
    - O compilador não sabe o que o identificador da classe significa
  - Há erros de tipo
    - Atribuição de um valor de tipo incompatível ao tipo da variável declarada
    - Operação inválida para um determinado tipo
    - Java é uma linguagem **fortemente tipada**

Ambiente de Programação

- Para acelerar o desenvolvimento de programas, é comum utilizar ambientes de programação ou ***IDE***s (Integrated Development Environment)
  - Integra várias ferramentas em um único ambiente
    - Editores de texto
    - Compiladores
    - APIs
    - E muito mais ...
  - **Eclipse, Net Beans, Visual J++, etc**

Escrevendo o Primeiro Programa

- Podemos, então escrever o primeiro programa utilizando as classes da API gráfica miniJava
- Ao fim do programa, podemos ter este resultado:

Resumindo...

- Conceitos Básicos de Java
  - Estrutura de um programa
  - Estruturas básicas de uma linguagem de programação
  - Criação de objetos
  - Chamada de métodos
  - Classe String
  - Importação de classes
  - Erros (estáticos) possíveis
- Escrevendo o primeiro programa
  - Utilizando a API gráfica
  - Executando um programa