

# **Unidade V:**

## **Estruturas de Dados Básicas**

### **com Alocação Flexível - Coleta de Lixo**

**Prof. Max do Val Machado**



**PUC Minas**

Instituto de Ciências Exatas e Informática  
Curso de Ciência da Computação

# Introdução

- A JVM realiza a coleta lixo automática, reivindicando a memória ocupada por objetos que não são mais acessíveis
- Quando não tivermos mais referências para um objeto, esse fica apto a ser coletado
- Assim, os vazamentos de memória (comuns em linguagens como C/C++) são menos comuns em Java
- O programador não tem controle sobre quando a JVM faz a coleta de lixo

## Método *finalize*

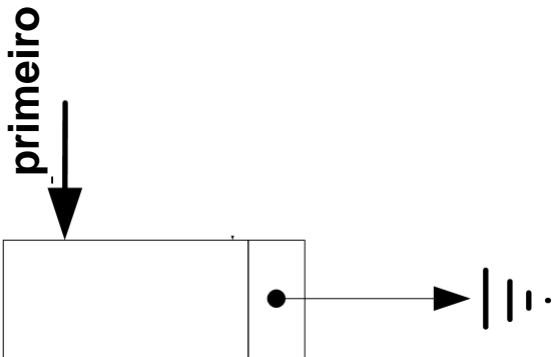
- Todo objeto possui este método, pois ele pertence à classe *Object* e, em Java, todas as classes herdam da *Object*
- Executado pelo coletor de lixo pouco antes da liberação de um objeto
- Raramente utilizado, pois pode prejudicar o desempenho e, como sua execução depende do coletor de lixo, ele também pode não ser executado antes do término do programa
- Não tem parâmetros, retorna *void* e sua visibilidade é *protected*

# Classe Coleta de Lixo

```
class ColetaLixo {  
    private Celula primeiro;  
    public ColetaLixo () {  
        primeiro = new Celula();  
    }  
    public void inserir (int x) {  
        primeiro.prox = new Celula(x);  
        primeiro = primeiro.prox;  
    }  
    public static void main (String [] args) {  
        ColetaLixo coletaLixo = new ColetaLixo();  
        while (true){  
            coletaLixo.inserir(0);  
        }  
    }  
}
```

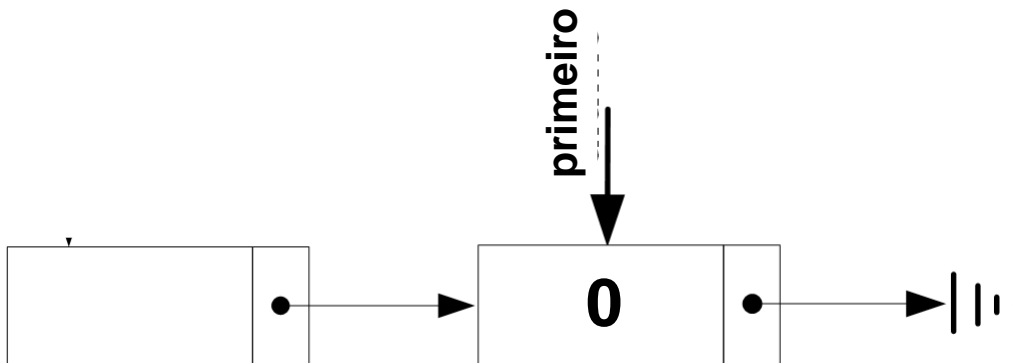
# Classe Coleta de Lixo

```
class ColetaLixo {  
    private Celula primeiro;  
    public ColetaLixo () {  
        primeiro = new Celula();  
    }  
    public void inserir (int x) {  
        primeiro.prox = new Celula(x);  
        primeiro = primeiro.prox;  
    }  
    public static void main (String [] args) {  
        ColetaLixo coletaLixo = new ColetaLixo();  
        while (true) coletaLixo.inserir(0);  
    }  
}
```



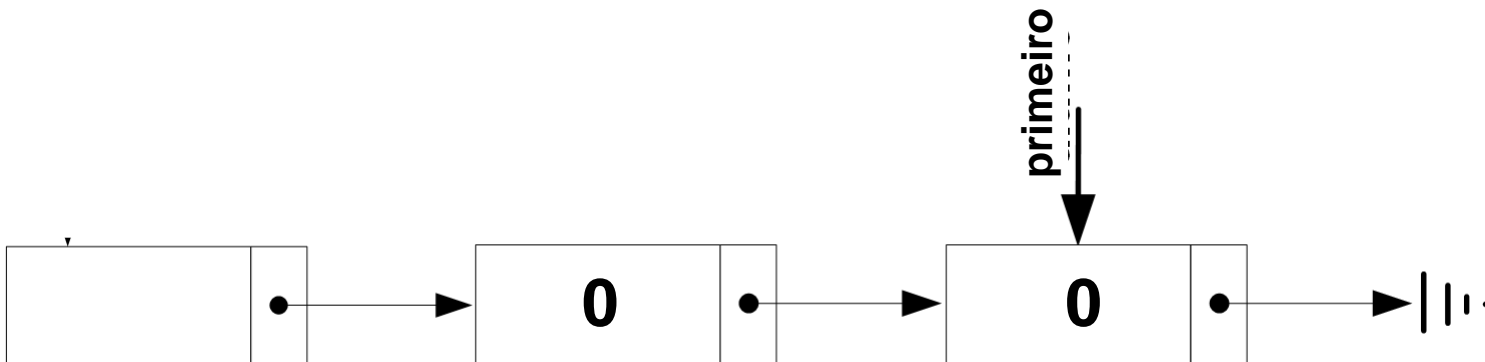
# Classe Coleta de Lixo

```
class ColetaLixo {  
    private Celula primeiro;  
    public ColetaLixo () {  
        primeiro = new Celula();  
    }  
    public void inserir (int x) {  
        primeiro.prox = new Celula(x);  
        primeiro = primeiro.prox;  
    }  
    public static void main (String [] args) {  
        ColetaLixo coletaLixo = new ColetaLixo();  
        while (true) coletaLixo.inserir(0);  
    }  
}
```



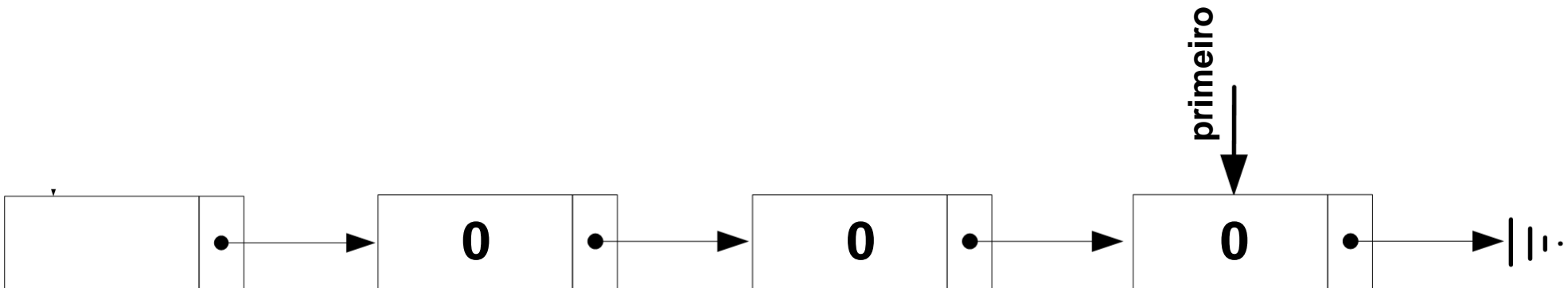
# Classe Coleta de Lixo

```
class ColetaLixo {  
    private Celula primeiro;  
    public ColetaLixo () {  
        primeiro = new Celula();  
    }  
    public void inserir (int x) {  
        primeiro.prox = new Celula(x);  
        primeiro = primeiro.prox;  
    }  
    public static void main (String [] args) {  
        ColetaLixo coletaLixo = new ColetaLixo();  
        while (true) coletaLixo.inserir(0);  
    }  
}
```



# Classe Coleta de Lixo

```
class ColetaLixo {  
    private Celula primeiro;  
    public ColetaLixo () {  
        primeiro = new Celula();  
    }  
    public void inserir (int x) {  
        primeiro.prox = new Celula(x);  
        primeiro = primeiro.prox;  
    }  
    public static void main (String [] args) {  
        ColetaLixo coletaLixo = new ColetaLixo();  
        while (true) coletaLixo.inserir(0);  
    }  
}
```



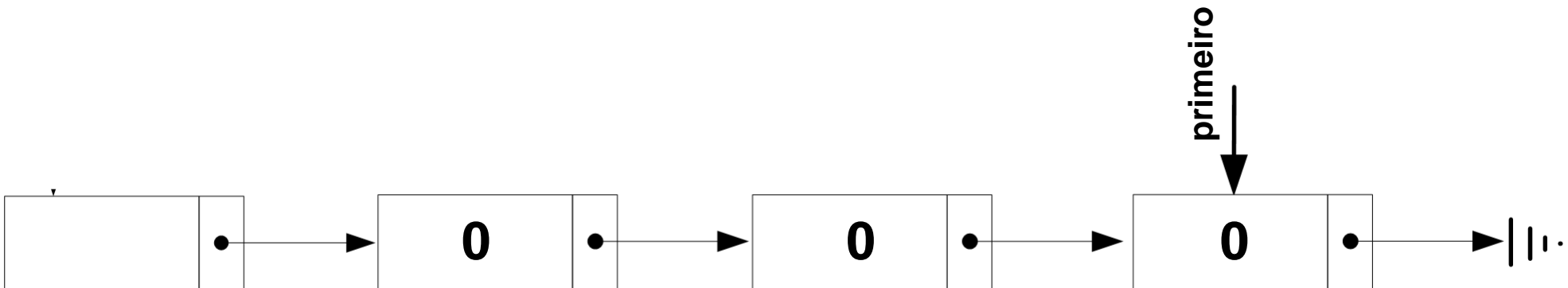


# Classe Coleta de Lixo

```
class ColetaLixo {  
    private Celula primeiro;  
    public ColetaLixo () {  
        primeiro = new Celula();  
    }  
}
```

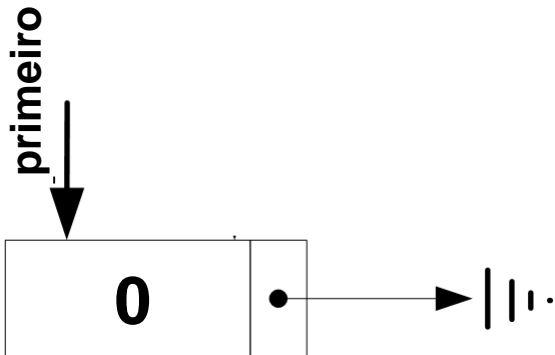
Após a coleta de lixo do Java ...

```
public static void main (String [] args) {  
    ColetaLixo coletaLixo = new ColetaLixo();  
    while (true) coletaLixo.inserir(0);  
}
```



# Classe Coleta de Lixo

```
class ColetaLixo {  
    private Celula primeiro;  
    public ColetaLixo () {  
        primeiro = new Celula();  
    }  
    public void inserir (int x) {  
        primeiro.prox = new Celula(x);  
        primeiro = primeiro.prox;  
    }  
    public static void main (String [] args) {  
        ColetaLixo coletaLixo = new ColetaLixo();  
        while (true) coletaLixo.inserir(0);  
    }  
}
```



## Exercício Resolvido (1)

- Compile e execute a classe `ColetaLixo.java`
- Abra outro terminal e execute o comando *top*
- Observe que seu programa consome praticamente toda a “CPU” e a mesma quantidade de memória apesar das infinitas alocações sucessivas
- Observe que seu programa não trava sua máquina dado que, nesse caso, o coletor de lixo atua proporcionalmente às alocações

## Exercício Resolvido (2)

- Compile o programa `coletalixo.c` (digite `gcc coletalixo.c -o coletalixo`)
- Execute o programa `coletalixo` (digite `./coletalixo`)
- Abra outro terminal e execute o comando `top`
- Observe que seu programa consome praticamente toda a “CPU” e a mesma quantidade de memória dado que ele efetua infinitas alocações e desalocações sucessivas

## Exercício Resolvido (3)

- Comente o programa `coletalixo.c` conforme mostrado abaixo

```
#include <stdlib.h>
```

```
#define true 1
```

```
typedef struct Celula {
```

```
    int elemento;
```

```
    struct Celula *prox;
```

```
} Celula;
```

```
Celula *novaCelula(int elemento) {  
    Celula *nova = (Celula*) malloc(sizeof(Celula));  
    nova->elemento = elemento;  
    nova->prox = NULL;  
    return nova;  
}
```

```
Celula *primeiro;
```

```
void start () {  
    primeiro = novaCelula(-1);  
}
```

```
void inserir(int x) {  
    primeiro->prox = novaCelula(x);  
    //Celula *tmp = primeiro;  
    primeiro = primeiro->prox;  
    //free(tmp);  
}
```

```
int main(int argc, char** argv) {  
    start();  
    while (true) inserir(0);  
    return 0;  
}
```

## Exercício Resolvido (3)

- Compile o programa `coetalixo.c` alterado
- Abra outro terminal e execute o comando `top`
- Execute o programa alterado, retorne **I.M.E.D.I.A.T.A.M.E.N.T.E** para a outra janela e veja que a memória consumida aumenta avassaladoramente
- Retorne **I.M.E.D.I.A.T.A.M.E.N.T.E** para a primeira janela e digite CTRL+C
  - **Cuidado sua máquina pode travar dado o elevado consumo de memória**