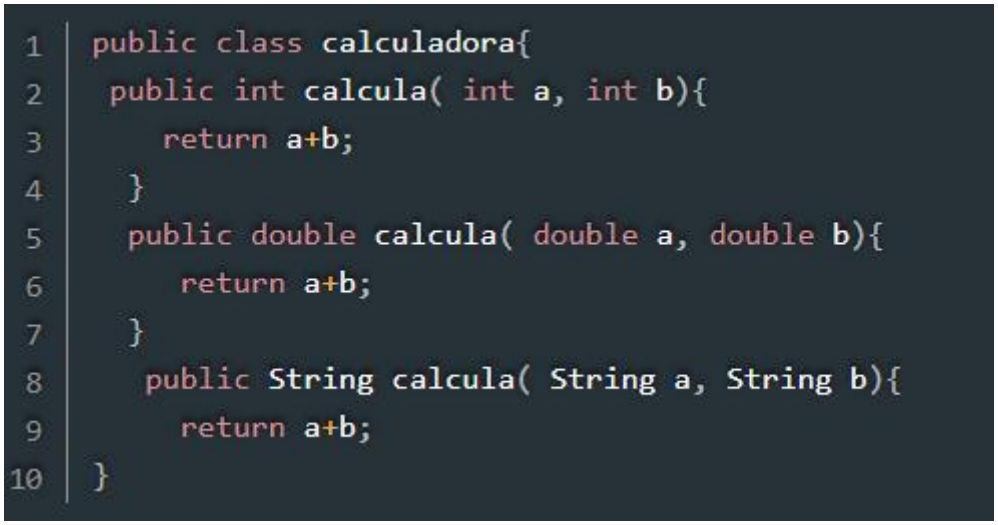
Dia 11 – Sobrecarga

Sobrecarga

O java nos permite trabalhar tanto com métodos sobrecarregados como com tipos genéricos, vamos ver qual a utilidade da Sobrecarga sempre mostrando primeiro o problema para depois apontar a solução.

Para entender melhor a sobrecarga, vamos pensar que estamos implementando uma calculadora simples que some apenas dois valores do mesmo tipo por vez. Nela teremos o método calcula que será sobrecarregado com variações de tipos de soma, como mostra a imagem abaixo.



```
1  public class calculadora{
2   public int calcula( int a, int b){
3      return a+b;
4    }
5    public double calcula( double a, double b){
6       return a+b;
7    }
8     public String calcula( String a, String b){
9       return a+b;
10 }
```

Imagem 1: Exemplo de métodos sobrecarregados

A classe calculadora possui três métodos que somam dois valores do mesmo tipo, porém, eles possuem o mesmo nome, então, como vamos saber se o programa principal vai chamar o método correto ao convocarmos o calcula ()? O programa, ao receber o calcula () com os parâmetros passados, verificará na classe calculadora no tempo de execução qual dos seguintes métodos está implementado para receber o parâmetro e convocará o mesmo. Na segunda imagem vemos um exemplo completo e sua implementação encontra-se na terceira imagem.

Dia 11 – Sobrecarga

```java
public class calculadora{
 public int calcula(int a,int b){
       return a+b;
 }
  public double calcula(double a,double b){
        return a+b;
   }
  public String calcula(String a,String b){
        return a+b;
  }
   public static void main(String args[]){
     calculadora calc= new calculadora();
     System.out.println(calc.calcula(1,1));
    System.out.println(calc.calcula(2.0,6.1));
    System.out.println(calc.calcula("vi","ram?"));
  }
}
```

Imagem 2. Exemplo da passagem de parâmetros para os métodos no main

Podemos ver que o programa principal chama o método sobrecarregado corretamente, então vemos a importância de manter diferentes lista de parâmetros para realizar a sobrecarga do método corretamente, pois caso contrário, o programa principal não conseguirá distingui-los e selecioná-los. Os métodos chamados são vinculados por pós-conexão, portanto, se o programa encontrar dois métodos com os mesmos parâmetros, não saberá qual método será selecionado para a chamada, e ocorrerá um erro no programa.

A sobrecarga é amplamente usada em construtores porque a sobrecarga consiste em linhas de código que sempre são executadas quando a classe é instanciada. Quando criamos um objeto a partir dele, ele deve ser instanciado. Normalmente, o programa criará um construtor não implementado para cada classe criada.

Neste caso, será o construtor padrão, mas podemos criar quantos construtores precisarmos. Podemos entender o construtor como a base inicial quando o objeto é criado. Como base, os construtores devem ter o mesmo nome da classe em que estão. Na classe da calculadora, temos um construtor padrão que não está implementado, mas não podemos vê-lo.

Vamos implementar a nossa classe calculadora com atributos e a sobrecarga de construtores conforme a imagem 3.

Dia 11 – Sobrecarga

```java
public class calculadora{

  private String modelo;
  private String marca;
  private String uso;

  //Sobrecarga de construtores.
  public calculadora(){
    // esse é o construtor padrão que o programa cria para todas as classes.
  }
  public calculadora(String marca,String modelo){
          this.marca=marca;
          this.modelo=modelo;
  }

  public calculadora(String marca,String modelo,String uso){
          this.marca=marca;
          this.modelo=modelo;
          this.uso=uso;
  }
  public int calcula(int a,int b){
       return a+b;
  }
  public double calcula(double a,double b){
        return a+b;
  }
  public String calcula(String a,String b){
         return a+b;
  }
  public static void main(String args[]){
   calculadora calc= new calculadora("optpex","N110","Empresarial");
   calculadora cald= new calculadora("Zion","Neo1");
   System.out.println(calc.calcula(500,1000));
   System.out.println(calc.calcula(99.8,100.1));
   System.out.println(calc.calcula("Sobrecarga de "," construtores"));
   System.out.println("calculadora 1 Marca: "+calc.marca+" Modelo:
   "+calc.modelo+" Uso: "+calc.uso);
   System.out.println("calculadora 2 Marca: "+cald.marca+" Modelo:
   "+cald.modelo);
  }
}
```

Imagem 3. Exemplo de sobrecarga de construtores

A sobrecarga de construtores tem muito em comum com a sobrecarga de métodos; podemos dizer que o conceito de sobrecarga é sempre o mesmo. Utilizamos o comando this para referenciar o objeto no qual estamos, por exemplo, no caso da primeira calculadora (a calc) passamos como parâmetro a marca “optplex”, ou seja, com esse comando o programa vai entender que estamos falando especificadamente da calc.

Portanto entendemos que a sobrecarga é conceito poderoso do polimorfismo, e ela permite ao programador mais facilidade na criação de variações de códigos já criados, poupando-o assim de inventar nomes para cada operação que compõem um mesmo escopo.