ToString

ao utilizar a classe de exemplo a baixo é possivel notar que ela irá gerar um objeto do tipo Pessoa com o nome "João" e idade 20, porém ao tentar imprimir o objeto, o resultado será o endereço de memória onde o objeto está armazenado.

```
public class Pessoa {
  private String nome;
  private int idade;

  public Pessoa(String nome, int idade) {
    this.nome = nome;
    this.idade = idade;
  }
}

public class Teste {
  public static void main(String[] args) {
    Pessoa p = new Pessoa("João", 20);
    System.out.println(p); // Pessoa@15db9742
  }
}
```

Para resolver esse problema, podemos sobrescrever o método toString() da classe Object, que é a classe mãe de todas as classes Java. O método toString() retorna uma String que representa o objeto. Por padrão, o método toString() retorna o nome da classe, seguido do caractere @ e do código hash do objeto em hexadecimal.

```
public class Pessoa {
  private String nome;
  private int idade;

  public Pessoa(String nome, int idade) {
    this.nome = nome;
    this.idade = idade;
  }

  @Override
  public String toString() {
    return "Nome: " + this.nome + ", idade: " + this.idade;
  }
}

public class Teste {
  public static void main(String[] args) {
    Pessoa p = new Pessoa("João", 20);
    System.out.println(p); // Nome: João, idade: 20
  }
}
```

Além disso o @override é uma anotação que indica que o método toString() está sendo sobrescrito. Se o método toString() não existir na classe Object, o compilador irá

gerar um erro.

> *Obs: existem outros métodos como o*

- equals()
- hashCode()
- clone()
- finalize()
- wait()
- notify()
- notifyAll()
- getClass()
- entre outros...