## método toString()

O Java usa o método toString() sempre que for necessário converter um objeto em String, ou seja, para obter ume representação textual do objeto. Por exemplo para imprimir um objeto qualquer usando System.out.println ou para mostrar os itens de um JComboBox.

Esse método toString() é definido na classe Object, portanto, é herdado por todos os objetos - todos os objetos são capazes de gerar uma representação textual. Mas o método do Object não conhece as classes derivadas, não *sabe* como o objeto deve ser representado. Por isso usa um padrão: o nome da classe seguido por um '@' e pelo hashCode () em hexadecimal da instância em questão:

```
        public String toString() {
       return getClass().getName() + "@" + Integer.toHexString(hashCode());
 }
```

Exemplo:

```
1.  public class Pessoa {
2.
3.      private final String nome;
4.      private int idade;
5.
6.      public Pessoa(String oNome, int aIdade) {
7.          nome = oNome;
8.          idade = aIdade;
9.      }
10.     public static void main(String[] args) {
11.         Pessoa pessoa = new Pessoa("fulano", 21);
12.         System.out.println(pessoa); // equivale aa System.out.println(pessoa.toString());
13.     }
14. }
```

O código acima imprime algo como "Pessoa@1b533b0" (provavelmente com outro número), que não é o formato desejado, para isso é necessário usar o método **toString().**

Exemplo: para mostrar o nome e a idade (entre parênteses) da pessoa, adicionamos o método toString():

```
1.  public class Pessoa {
2.
3.      private final String nome;
4.      private int idade;
5.
6.      public Pessoa(String oNome, int aIdade) {
7.          nome = oNome;
8.          idade = aIdade;
9.      }
10.     @Override
11.     public String toString() {
12.         return nome + "(" + idade + ")";
13.     }
14.
```

```
15.     public static void main(String[] args) {
16.         Pessoa pessoa = new Pessoa("fulano", 21);
17.         System.out.println(pessoa);
18.     }
19. }
```

**OUTPUT:** fulano(21)

O mesmo problema ocorre com um vetor (Array):

```
1.  ...
2.      public static void main(String[] args) {
3.          int[] A = {1, 2, 3};
4.          System.out.println(A);
5.      }
6.  ...
```

OUTPUT: [I@15497e0 - "[I" representa um array de int's.

Neste caso não podemos usar diretamente o método toString() num Array, mas podemos usar a classe auxiliar Arrays do pacote java.util que contém vários métodos estáticos para trabalhar com um Array. No nosso caso podemos usar o toString(int[]):

```
1.  import java.util.Arrays;
2.  ...
3.      public static void main(String[] args) {
4.          int[] A = {1, 2, 3};
5.          System.out.println(Arrays.toString(A));
6.      }
7.  ...
```

OUTPUT: [1, 2, 3]

**Resumo:**
Sempre que se obtenha algo como "Classe@146c21b" é porque a classe em questão não sobrescreveu o método toString().

**Solução:**
Implementar/sobrescrever o método toString() do Objeto ou usar o Arrays.toString() se for um Array. (Exemplos acima)

## Exercícios:

1. Na classe **ExecutarComida**, do pacote Java1_F8, acrescenta no método *main* um objeto de nome **batata** com os seguintes valores:

Designação: "Batata cozida"

Calorias: 71

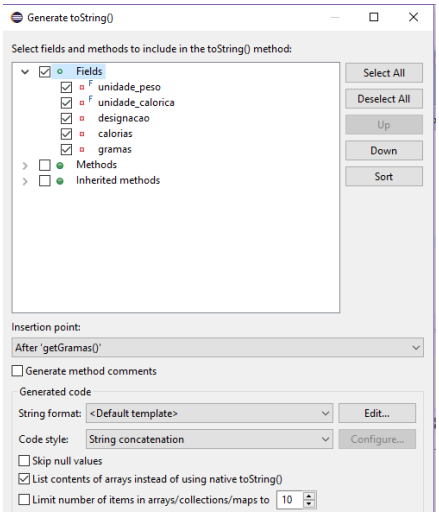
Gramas: 140

2. Lista os dados do objeto **batata** utilizando System.out.println e os métodos getters. De modo a listar a seguinte informação:

   Batata cozida tem 71cal em 140g

3. Lista os dados do objeto **batata** utilizando System.out.println e o método toString. De modo a listar a seguinte informação:

em Source escolhe"*toString…*", seleciona todos os campos e clica em OK.




É gerado o código seguinte:

```
@Override
	public String toString() {
		return "Comida [unidade_peso=" +
unidade_peso + ", unidade_calorica=" +
unidade_calorica + ", designacao="
	+ designacao + ", calorias=" + calorias + ",
gramas=" + gramas + "]";
}
```

4. Na classe **ExecutarComida**, acrescenta a seguinte linha de código no final do método main:

```
System.out.println(batata.toString());
```

5. Executa o programa.

6. Altera os campos do objeto batata, utilizando os Setters criados na ficha anterior, para:

   Calorias: 37,5

   Gramas: 50

7. Lista os novos valores como feito na questão 2 e 3.
8. Executa o programa.
9. Acrescenta mais dois objetos massa e ameixa, inventa valores de calorias para 100g e altera os mesmos para 50g, como feito nas questões 1, 6 e 7.
10. Acrescenta um novo construtor que recebe apenas como argumento a designação da comida, sendo os campos calorias e gramas inicializados respetivamente a 225 e 50.
11. Declara um objeto alface com o novo construtor.
12. Lista os novos valores e executa o programa.
13. Altera as calorias do objeto alface para 12 e as gramas para 100, recorrendo aos Setters.
14. Lista os novos valores e executa o programa.


Bibliografia:
https://www.w3schools.com/java
https://www.tutorialspoint.com/java/
Jesus, C. (2013). Curso Prático de Java. Lisboa: FCA
Coelho, P (2016). Programação em JAVA – Curso Completo. Lisboa: FCA