@ToString

No need to start a debugger to see your fields: Just let lombok generate a `toString` for you!

Overview

Annotating a class with `@ToString` will cause lombok to generate an implementation of the `toString()` method. You use configuration options to specify whether field names should be included but otherwise the format is fixed: the class name followed by parentheses containing fields separated by commas, e.g. `MyClass(foo=123, bar=234)`.

By setting the `includeFieldNames` parameter to *true* you can add some clarity (but also quite some length) to the output of the `toString()` method.

By default, all non-static fields will be printed. If you want to skip some fields, you can annotate these fields with `@ToString.Exclude`. Alternatively, you can specify exactly which fields you wish to be used by using `@ToString(onlyExplicitlyIncluded = true)`, then marking each field you want to include with `@ToString.Include`.

By setting `callSuper` to *true*, you can include the output of the superclass implementation of `toString` to the output. Be aware that the default implementation of `toString()` in `java.lang.Object` is pretty much meaningless, so you probably don't want to do this unless you are extending another class.

You can also include the output of a method call in your `toString`. Only instance (non-static) methods that take no arguments can be included. To do so, mark the method with `@ToString.Include`.

You can change the name used to identify the member with `@ToString.Include(name = "some other name")`, and you can change the order in which the members are printed via `@ToString.Include(rank = -1)`. Members without a rank are considered to have rank 0, members of a higher rank are printed first, and members of the same rank are printed in the same order they appear in the source file.

With Lombok

```java
import lombok.ToString;

@ToString
public class ToStringExample {
  private static final int STATIC_VAR = 10;
  private String name;
  private Shape shape = new Square(5, 10);
  private String[] tags;
  @ToString.Exclude private int id;

  public String getName() {
    return this.name;
  }

  @ToString(callSuper=true, includeFieldNames=true)
  public static class Square extends Shape {
    private final int width, height;

    public Square(int width, int height) {
      this.width = width;
      this.height = height;
    }
  }
}
```

Vanilla Java

```java
import java.util.Arrays;

public class ToStringExample {
  private static final int STATIC_VAR = 10;
  private String name;
  private Shape shape = new Square(5, 10);
  private String[] tags;
  private int id;

  public String getName() {
    return this.name;
  }

  public static class Square extends Shape {
    private final int width, height;

    public Square(int width, int height) {
      this.width = width;
      this.height = height;
    }

    @Override public String toString() {
      return "Square(super=" + super.toString() + ", width=" + this.width + ", height=" + this.height + ")"
    }
  }

  @Override public String toString() {
    return "ToStringExample(" + this.getName() + ", " + this.shape + ", " + Arrays.deepToString(this.tags)
  }
}
```

Supported configuration keys:

`lombok.toString.includeFieldNames` = [`true` | `false`] (default: true)
Normally lombok generates a fragment of the toString response for each field in the form of `fieldName = fieldValue`. If this setting is set to `false`, lombok will omit the name of the field and simply deploy a comma-separated list of all the field values. The annotation parameter '`includeFieldNames`', if explicitly specified, takes precedence over this setting.
`lombok.toString.doNotUseGetters` = [`true` | `false`] (default: false)
If set to `true`, lombok will access fields directly instead of using getters (if available) when generating `toString` methods. The annotation parameter '`doNotUseGetters`', if explicitly specified, takes precedence over this setting.
`lombok.toString.callSuper` = [`call` | `skip` | `warn`] (default: skip)
If set to `call`, lombok will generate calls to the superclass implementation of `toString` if your class extends something. If set to `skip` no such call is generated. If set to `warn` no such call is generated either, but lombok does generate a warning to tell you about it.
`lombok.toString.onlyExplicitlyIncluded` = [`true` | `false`] (default: false)
If set to `false` (default), all fields (unless `static`, name starts with a dollar, or otherwise excluded for obvious reasons) serve as the default set of things to include in the toString, modifiable by using the `@ToString.Exclude` and `@ToString.Include` options. If set to `true`, nothing is included unless explicitly marked with `@ToString.Include`.
`lombok.toString.flagUsage` = [`warning` | `error`] (default: not set)
Lombok will flag any usage of `@ToString` as a warning or error if configured.

Small print

If there is *any* method named `toString` with no arguments, regardless of return type, no method will be generated, and instead a warning is emitted explaining that your `@ToString` annotation is doing nothing. You can mark any method with `@lombok.experimental.Tolerate` to hide them from lombok.

Arrays are printed via `Arrays.deepToString`, which means that arrays that contain themselves will result in `StackOverflowError`s. However, this behaviour is no different from e.g. `ArrayList`.

If a method is marked for inclusion and it has the same name as a field, it replaces the toString output for that field (the method is included, the field is excluded, and the method's output is printed in the place the field would be printed).

Prior to lombok 1.16.22, inclusion/exclusion could be done with the `of` and `exclude` parameters of the `@ToString` annotation. This old-style inclusion mechanism is still supported but will be deprecated in the future.

Having both `@ToString.Exclude` and `@ToString.Include` on a member generates a warning; the member will be excluded in this case.

We don't promise to keep the output of the generated `toString()` methods the same between lombok versions. You should never design your API so that other code is forced to parse your `toString()` output anyway!

By default, any variables that start with a $ symbol are excluded automatically. You can only include them by using the `@ToString.Include` annotation.

If a getter exists for a field to be included, it is called instead of using a direct field reference. This behaviour can be suppressed:
`@ToString(doNotUseGetters = true)`

`@ToString` can also be used on an enum definition.

If you have configured a nullity annotation flavour via `lombok.config` key `lombok.addNullAnnotations`, the method or return type (as appropriate for the chosen flavour) is annotated with a non-null annotation.