@Getter(lazy=true)

Laziness is a virtue!

`@Getter(lazy=true)` was introduced in Lombok v0.10.

Overview

You can let lombok generate a getter which will calculate a value once, the first time this getter is called, and cache it from then on. This can be useful if calculating the value takes a lot of CPU, or the value takes a lot of memory. To use this feature, create a `private final` variable, initialize it with the expression that's expensive to run, and annotate your field with `@Getter(lazy=true)`. The field will be hidden from the rest of your code, and the expression will be evaluated no more than once, when the getter is first called. There are no magic marker values (i.e. even if the result of your expensive calculation is `null`, the result is cached) and your expensive calculation need not be thread-safe, as lombok takes care of locking.

If the initialization expression is complex, or contains generics, we recommend moving the code to a private (if possible static) method, and call that instead.

With Lombok

```
import lombok.Getter;

public class GetterLazyExample {
  @Getter(lazy=true) private final double[] cached = expensive();

  private double[] expensive() {
    double[] result = new double[1000000];
    for (int i = 0; i < result.length; i++) {
      result[i] = Math.asin(i);
    }
    return result;
  }
}
```

Vanilla Java

```
public class GetterLazyExample {
  private final java.util.concurrent.AtomicReference<java.lang.Object> cached = new java.util.concurrent.At

  public double[] getCached() {
    java.lang.Object value = this.cached.get();
    if (value == null) {
      synchronized(this.cached) {
        value = this.cached.get();
        if (value == null) {
          final double[] actualValue = expensive();
          value = actualValue == null ? this.cached : actualValue;
          this.cached.set(value);
        }
      }
    }
    return (double[])(value == this.cached ? null : value);
  }

  private double[] expensive() {
    double[] result = new double[1000000];
    for (int i = 0; i < result.length; i++) {
      result[i] = Math.asin(i);
    }
    return result;
  }
}
```

Supported configuration keys:

`lombok.getter.lazy.flagUsage` = [`warning` | `error`] **(default: not set)**
Lombok will flag any usage of `@Getter(lazy=true)` as a warning or error if configured.

Small print

You should never refer to the field directly, always use the getter generated by lombok, because the type of the field will be mangled into an `AtomicReference`. Do not try to directly access this `AtomicReference`; if it points to itself, the value has been calculated, and it is `null`. If the reference points to `null`, then the value has not been calculated. This behaviour may change in future versions. Therefore, *always* use the generated getter to access your field!

Other Lombok annotations such as `@ToString` always call the getter even if you use `doNotUseGetters=true`.