Название: @NonNull
Описание: обработка переменных, которые не должны получать null
Код Lombok:

```
public Example(@NonNull P p) {
 super("Hello");
 this.name = p.getName();
}
```

Код обычной Java:

```
public Example(@NonNull P p) {
  super("Hello");
  if (p == null) {
    throw new NullPointerException("p");
  }
  this.name = p.getName();
}
```

Название: @Getter / @Setter
Описание: легкое создание getter'ов и setter'ов
Код Lombok:

```
@Getter
@Setter
private int age = 10;
```

Код обычной Java:

```
private int age = 10;

public int getAge() {
 return age;
}

public void setAge(int age) {
  this.age = age;
}
```

Название: @ToString
Описание: определение аннотации перед классом, для реализации стандартного toString метода
Код Lombok:

```
@ToString(exclude="f")
public class Example
```

Код обычной Java:

```
public class Example {
  @Override
```

```
  public String toString() {
    return ...;
  }
```

Название: @EqualsAndHashCode
Описание: легкое создание методов Equals и HashCode
Код Lombok:

```
@EqualsAndHashCode(
   exclude={"id1", "id2"})
public class Example {
```

Код обычной Java:

```
public class Example {
  ...
  @Override
  public boolean equals(Object o) {
     ...
  }

  @Override
  public int hashCode() {
     ...
  }
```

Название:
@NoArgsConstructor,
@RequiredArgsConstructor,
@AllArgsConstructor
Описание: создания пустого конструктора,
конструктора включающего все final поля,
либо конструктора включающего все возможные поля
Код Lombok:

```
@RequiredArgsConstructor(
     staticName = "of"
)
@AllArgsConstructor(
    access = AccessLevel.PROTECTED
)
public class E<T> {
```

Код обычной Java:

```
public class E<T> {

 private E(T description) {
     ...
 }

 public static <T>E<T> of(
     T description
) {
   return new E<T>(description);
 }
```

Название: @Data
Описание: генерация всех служебных методов,
заменяет сразу команды @ToString, @EqualsAndHashCode,
@Getter, @Setter, @RequiredArgsConstructor
Код Lombok:

```
@Data
public class Example {
  private final String name;
  private int age;
}
```

```
public class Example {
  private final String name;
  private int age;

  public Example(
     String name
  ) {
    this.name = name;
  }

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

  void setAge(int age) {
    this.age = age;
  }

  public int getAge() {
    return this.age;
  }

  @Override
 public String toString() {
    return ...;
  }

  @Override
  public boolean equals(
    Object o
  ) {
    ....
  }

  @Override
  public int hashCode() {
    ...
  }
```

Название: @Value
Описание: создание неизменяемых классов,
аналог Data, но для неизменяемых классов
Код Lombok:

```
@Value
public class Example {
  private final String name;
  private int age;
}
```

Код обычной Java:

```
public class Example {
  private final String name;
  private final int age;

  public Example(
    String name, int age
) {
    this.name = name;
    this.age = age;
  }

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

  public int getAge() {
    return this.age;
  }

  @Override
 public String toString() {
    return ...;
  }

  @Override
  public boolean equals(
     Object o
  ) {
    ....
  }

  @Override
  public int hashCode() {
    ...
  }
```

Название: @Builder
Описание: реализация паттерна bulder,
@Singular – используется для объектов в
единственном экземпляре (добавления элемента
в коллекции и т.п.)
Код Lombok:

```
@Builder
public class Example {
  private String name;
  private int age;
  @Singular
  private Set<String> occupations;
}
```

Код обычной Java:

```
public class Example {
  private String name;
  private int age;
  private Set<String> occupations;

  Example(
    String name,
    int age,
   Set<String> occupations
 ) {
    this.name = name;
    this.age = age;
    this.occupations = occupations;
  }

  public static ExampleBuilder builder() {
    return new ExampleBuilder();
  }

  public static class ExampleBuilder {
    private String name;
    private int age;
    private ArrayList<> occupations;

    ExampleBuilder() {
    }

    public ExampleBuilder name(
       String name
    ) {
      this.name = name;
      return this;
    }

    public ExampleBuilder age(
      int age
    ) {
      this.age = age;
      return this;
    }

    public ExampleBuilder occupation(
      String occupation
   ) {
      if (this.occupations == null) {
        this.occupations =
          new ArrayList<String>();
      }

      this.occupations.add(occupation);
      return this;
    }

   ...
    public Example build() {
      Set<String> occupations = ...;
      return new Example(name, age, occupations);
    }

    @java.lang.Override
    public String toString() {
      ...
    }
```

```
    }
}
```

Название: @SneakyThrows
Описание: обертка проверяемых исключений
Код Lombok:

@SneakyThrows(
UnsupportedEncodingException.class)
public String utf8ToString(byte[] bytes) {
return new String(bytes, «UTF-8»);
}

Код Lombok:

```
public String utf8ToString(byte[] bytes) {
    try {
      return new String(bytes, "UTF-8");
    } catch (UnsupportedEncodingException e) {
      throw Lombok.sneakyThrow(e);
    }
 }
```

Название: @Synchronized
Описание: простое создание synchronized блоков
Код Lombok:

```
private final Object readLock = new Object();

@Synchronized
public static void hello() {
    ...;
 }

 @Synchronized
 public int answerToLife() {
   ...
 }

 @Synchronized("readLock")
 public void foo() {
    ...
 }
```

Код обычной Java:

```
private static final Object $LOCK = new Object[0];
private final Object $lock = new Object[0];
private final Object readLock = new Object();

public static void hello() {
  synchronized($LOCK) {
    ...
   }
}

public int answerToLife() {
```

```
  synchronized($lock) {
    ...
   }
}

public void foo() {
   synchronized(readLock) {
      ...
   }
}
```

Название: @Log
Описание: добавление инницилизации логирования,
так же позволяет выбрать вид логгера: @CommonsLog,
@JBossLog, @Log, @Log4j, @Log4j2, @Slf4j, @XSlf4j
Код Lombok:

```
@Slf4j
public class Example {
  public static void main(String... args) {
    log.error("error");
  }
```

Код обычной Java:

```
public class Example {
  private static final org.slf4j.Logger log = org.slf4j.LoggerFactory.getLogger(LogExampleOther.class);

  public static void main(String... args) {
    log.error("error");
  }
```

Val простое создание финальной
переменной с выводом типа,
то есть то самый val о котором
спорили

```
val map = new HashMap<Integer, String>();
for (val entry : map.entrySet()) {
      ...
}
```

```
final HashMap<Integer, String> map = new HashMap<Integer, String>();
...
for (final Map.Entry<Integer, String> entry : map.entrySet()) {
      ...
}
```

Название: @Cleanup
Описание: простое определение ресурсов,
так чтобы они автоматически закрывались
после окончания работы кода.

(не так актуально при использовании
try with resources)

Код Lombok:

```
@Cleanup InputStream in = new FileInputStream(args[0]);
@Cleanup OutputStream out = new FileOutputStream(args[1]);
...
```

Код обычной Java:

```
InputStream in = new FileInputStream(args[0]);
    try {
      OutputStream out = new FileOutputStream(args[1]);
      try {
        ...
      } finally {
        if (out != null) {
          out.close();
        }
      }
    } finally {
      if (in != null) {
        in.close();
      }
    }
```