* 学习目标

* 能够理解json的概述

    * Javascript Object Notation--JSON

    * 作为配置文件，文本传输，独立语言

        * properties,xml,json,yaml

* 能够掌握json和xml的区别

    * XML与JSON传输：JSON比XML更小，更快，更易解析

* 能够掌握json的展现形式

    * {[{"key1":"value1","key2":"value2"},{}]}

* 能够掌握通过HiJson工具查看json数据

* 能够掌握json的语法

    * {[{"key1":数字|boolean|数组|string|null,"key2":"value2"},{}]}

* 能够掌握js中使用json

    * var obj={[{"key1":"value1","key2":"value2"},{}]}
    * eval()--jsonStr--js的对象

* 能够掌握通过gson让json字符与javabean互相转换

    * Gson ---> Google---> fromJson,toJson

* 能够掌握通过fastjson让json字符与javabean互相转换

    * 阿里巴巴

    * JSON.toString(), JSON.parseObject,JSON.parseArray

    * Gson,fastJson,jackson

* 能够理解注解的概述

    * Java的标注，JDK1.5

    * 注释和注解区别

    * 在编码阶段，编译阶段，运行阶段（通过反射获取注解值）

    * 在编码阶段：JDK注解：
@Override,@Deprecated,@SupressWarning,@SafeVarargs,.

* 编译阶段: LomBok

* 运行阶段：替换配置文件

* 现在JAVA开发是注解天下

* 能够掌握jdk常用的注解

* 能够掌握LomBok常用的注解

* LomBok-->框架--->自定生存get/set 构造器，toString,hashcode equals,非空，日志对象，

* @Data,@Setter,@Getter,@AllArgsConstructor,@NoArgsConstructor

* @RequiedArgsConstructor,@Log,@Log4j,@Log4j4,.....@NonNull,@CleanUp

* var list=new ArrayList();--- 非final

val list=new ArrayList();---final

---

* 能够理解json的概述

* JSON 指的是 JavaScript(JS) 对象表示法（JavaScript Object Notation）

* JSON 是轻量级的文本数据交换格式

* JSON 独立于语言

* JSON 具有自我描述性，更易理解

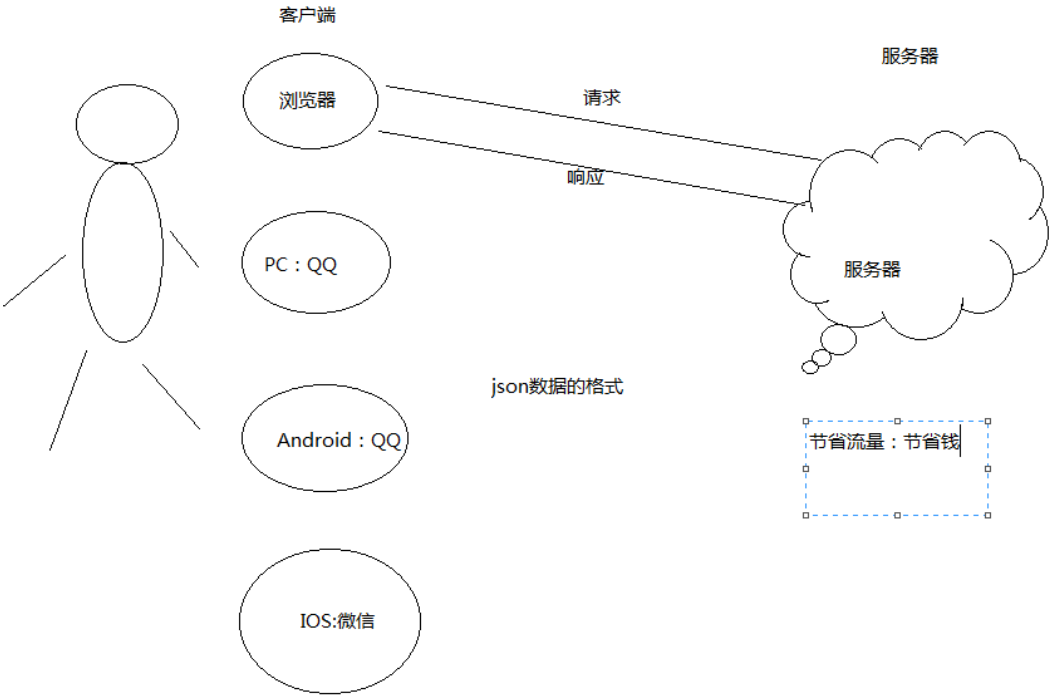
* JSON 使用 JavaScript 语法来描述数据对象，但是 JSON 仍然独立于语言和平台。

* JSON 解析器和 JSON 库支持许多不同的编程语言。

* 能够掌握json和xml的区别

* JSON 是存储和交换文本信息的语法。类似 XML。

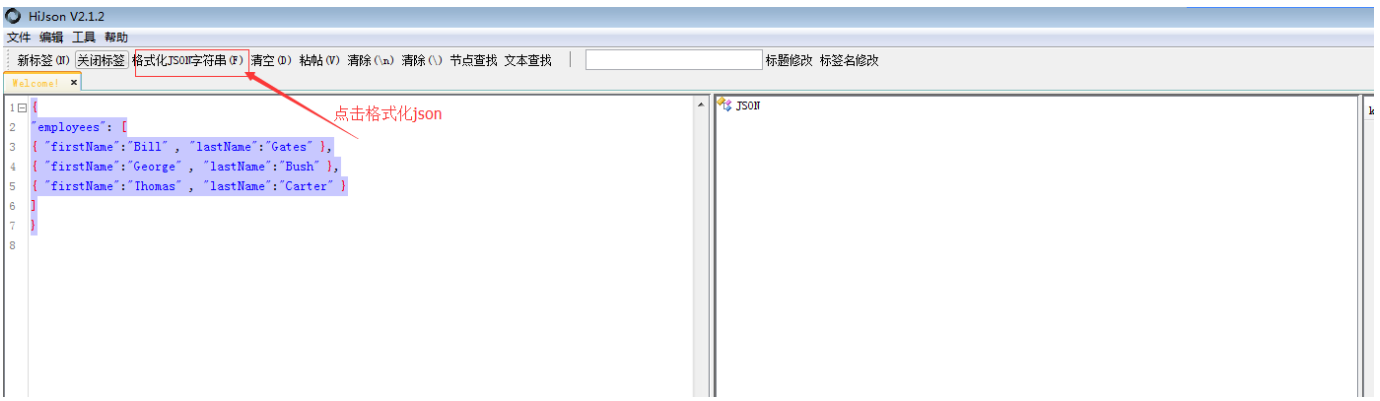
* JSON 比 XML 更小、更快，更易解析。

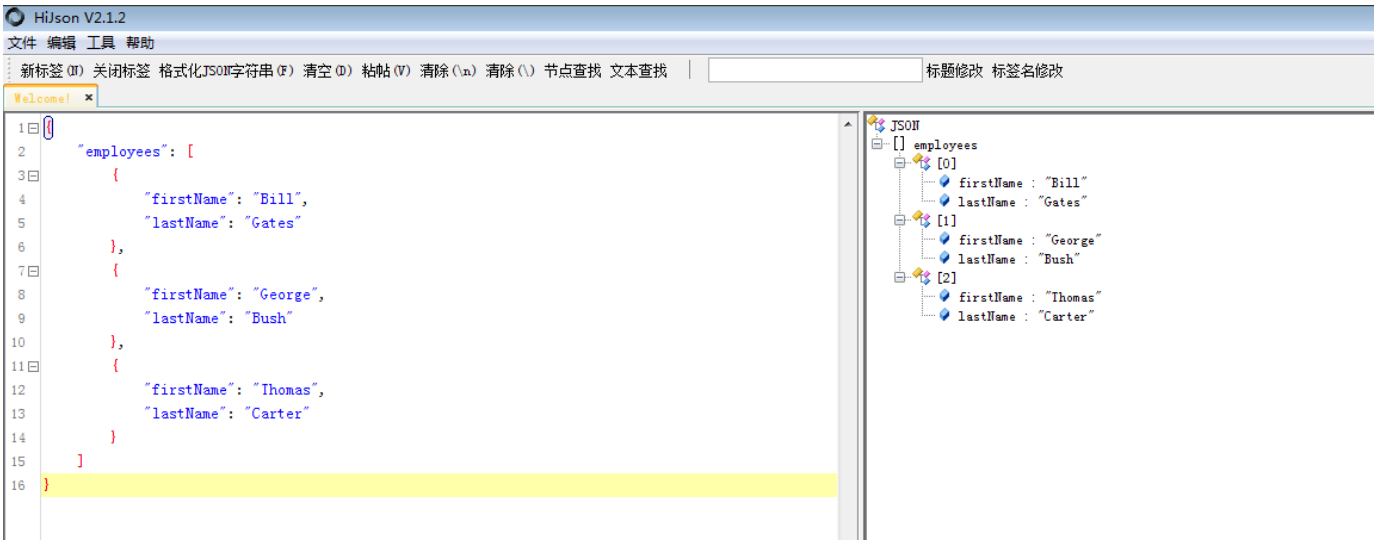


* 能够掌握json的展现形式

```
{ "employees": [
    { "firstName":"Bill" , "lastName":"Gates" },
    { "firstName":"George" , "lastName":"Bush" },
    { "firstName":"Thomas" , "lastName":"Carter" }
    ]
}
```

* 能够掌握通过HiJson工具查看json数据





* 能够掌握json的语法
  * Json的语法规则
    * 数据在名称/值对中
    * 数据由逗号分隔
    * 花括号保存对象
    * 方括号保存数组
  * Json的值可以是那些
    * 数字（整数或浮点数）
    * 字符串（在双引号中）
    * 逻辑值（true 或 false）
    * 数组（在方括号中）
    * 对象（在花括号中）
    * null

* 能够掌握js中使用json

```
* 案例一：在js中定义对象
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<script type="text/javascript">
    function clickMe(){
        alert("come on baby");
        var jsonObject={
                "id":"1001",
                "name":"thinking in java",
                "price":"90",
                "author":"xiaohei"
        };
        document.getElementById("jid").innerHTML=jsonObject.id;
        document.getElementById("jname").innerHTML=jsonObject.name;
        document.getElementById("jprice").innerHTML=jsonObject.price;
        document.getElementById("jauthor").innerHTML=jsonObject.author;
    }
</script>
</head>
```

```
<body>
        <h2>在JavaScript创建Json对象</h2>
        <p>
            id:<span id="jid"></span><br>
            name:<span id="jname"></span><br>
            price:<span id="jprice"></span><br>
            author:<span id="jauthor"></span><br>
        </p>
        <input type="button" onclick="clickMe()" value="获取的Json注解"/>
</body>
</html>

* 案例二(Json字符串转换成Json对象)
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<script type="text/javascript">
    var txt = '{"id":"1001","name":"thinking in java","price":"90","author":"xi
    function clickMe() {
        alert("come on baby");
        // json字符串转换成json对象
        var jsonObject = eval("(" + txt + ")");
        document.getElementById("jid").innerHTML = jsonObject.id;
        document.getElementById("jname").innerHTML = jsonObject.name;
        document.getElementById("jprice").innerHTML = jsonObject.price;
        document.getElementById("jauthor").innerHTML = jsonObject.author;
    }
</script>
</head>
<body>
    <h2>在JavaScript创建Json对象</h2>
    <p>
        id:<span id="jid"></span><br> name:<span id="jname"></span><br>
        price:<span id="jprice"></span><br> author:<span id="jauthor"></span><b
    </p>
    <input type="button" onclick="clickMe()" value="获取的Json注解" />
</body>
</html>
```

{“id”:"10001",“name”:"java","price":99.9,"author":"lg"}

* 能够掌握通过gson让json字符与javabean互相转换

* Gson概述

* Gson 是 Google 提供的用来在 Java 对象和 JSON 数据之间进行映射的 Java 类库。可以将一个 JSON 字符串转成一个 Java 对象

* 开发步骤

* 导入gson的jar包

* idea添加jar：参考02-添加jar到idea

```
 * 案例一： JavaBean（Book）对象转换成Json字符串
 package com.lg.bean;

/**
 * @author xiaozhao
 */
public class Book {
    private int id;
    private String name;
    private double price;
    private String author;
    public Book() {
        super();
    }
    public Book(int id, String name, double price, String author) {
        super();
        this.id = id;
        this.name = name;
        this.price = price;
        this.author = author;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
```

```
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public double getPrice() {
        return price;
    }
    public void setPrice(double price) {
        this.price = price;
    }
    public String getAuthor() {
        return author;
    }
    public void setAuthor(String author) {
        this.author = author;
    }
    @Override
    public String toString() {
        return "Book [id=" + id + ", name=" + name + ", price=" + price + ", au
    }

}
 @Test
  public void test1() {
        // 0 构建Book对象
        Book book=new Book();
        book.setId(1002);
        book.setName("thinking in java");
        book.setPrice(89.9d);
        book.setAuthor("xiaobai");
        // 1 构建Gson的对象
        Gson gson=new Gson();
        // 2 调用Gson的对象的方法：book对象转换成json字符串
        String json = gson.toJson(book);
        // 3 打印json字符串
        System.out.println(json);
   }
* 结果
```

```
{"id":1002,"name":"thinking in java","price":89.9,"author":"xiaobai"}

* 案例二：Json字符串转换成JavaBean
  @Test
 public void test2() {
  // 1 定义一串json字符串
 String jsonStr="{\"id\":1002,\"name\":\"thinking in java\",\"price\":89.9,\"au
  // 2 构建Gson的对象
  Gson gson=new Gson();
  // 3 调用Gson的对象方法
  Book book = gson.fromJson(jsonStr, Book.class);
  // 4 打印这个JavaBean（Book）
   System.out.println(book);
    }

结果：
 Book [id=1002, name=thinking in java, price=89.9, author=xiaobai]

* 案例三：集合List<Bean>对象转换成Json字符串
@Test
    public void test3() {
        // 1 构建List集合的对象
        List<Book> books=new ArrayList<Book>();
        Book book1=new Book();
        book1.setId(1001);
        book1.setName("thinking in C++");
        book1.setPrice(89.9d);
        book1.setAuthor("xiaohei");
        Book book2=new Book();
        book2.setId(1002);
        book2.setName("thinking in Java");
        book2.setPrice(88.9d);
        book2.setAuthor("xiaobai");
        books.add(book1);
        books.add(book2);
        // 2 构建Gson的对象
        Gson gson=new Gson();
        // 3 调用Gson的转换集合的方法
        String json = gson.toJson(books);
        // 4 打印json字符串
```

```
        System.out.println(json);
    }

* 结果
[{"id":1001,"name":"thinking in C++","price":89.9,"author":"xiaohei"},
 {"id":1002,"name":"thinking in Java","price":88.9,"author":"xiaobai"}]

* 案例四：Json字符串转换成集合
@Test
    public void test4() {
//        1 定义json字符串
        String jsonStr="[{\"id\":1001,\"name\":\"thinking in C++\",\"price\":89
//        2 构建Gson的对象
        Gson gson=new Gson();
//        3 调用Gson的转换json的方法
//      gson.fromJson(jsonStr, List<Book>.class)
        List<Book> books = gson.fromJson(jsonStr, new TypeToken<List<Book>>(){}
//        4 打印集合字符串
        System.out.println(books);
    }
* 结果
[Book [id=1001, name=thinking in C++, price=89.9, author=xiaohei],
 Book [id=1002, name=thinking in Java, price=88.9, author=xiaobai]]
```

* 能够掌握通过fastjson让json字符与javabean互相转换

  * fastjson概述

    * fastjson是阿里巴巴开发的一款专门用于Java开发的包

    * fastjson是一个Java库，可用于将Java对象转换为其JSON表示。它还可用于将JSON字符串转换为等效的Java对象

  * 开发步骤

    * 导入fastjson的jar包

```
案例一：JavaBean（Book）对象转换成Json字符串
  @Test
 public void test5() {
        Book bk = new Book(101, "java", 12.5, "Bruce Eckle");
```

```
        String jsonString = JSON.toJSONString(bk);
        System.out.println(jsonString);
 }
 * 结果
  * {"author":"Bruce Eckle","id":101,"name":"java","price":12.5}
 * 案例二：Json字符串转换成JavaBean
   @Test
 public void test6() {
        String json="{\"author\":\"Bruce Eckle\",\"id\":101,\"name\":\"java\",\
        Book bk = JSON.parseObject(json,Book.class);
        System.out.println(bk);
  }
* 结果
Book [id=101, name=java, price=12.5, author=Bruce Eckle]
* 案例三：集合List<Bean>对象转换成Json字符串
@Test
    public void test7() {
        List<Book> list = new ArrayList<Book>();
        Book bk = new Book(101, "javaaaaa", 12.5, "Bruce Eckle");
        Book bk1 = new Book(102, "c+++++", 88.5, "Bruce Tomson");
        Book bk2 = new Book(103, "pythonnnn", 72.8, "Bruce Eraln");
        Book bk3 = new Book(104, "c#####", 145.5, "Bruce Cook");
        list.add(bk);
        list.add(bk1);
        list.add(bk2);
        list.add(bk3);
        System.out.println(JSON.toJSONString(list));
    }
* 案例四：Json字符串转换成集合
 @Test
    public void test8() {
        String json="[{\"author\":\"Bruce Eckle\",\"id\":101,\"name\":\"javaaaa
        List<Book> list = JSON.parseArray(json, Book.class);
        for (Book b : list) {
            System.out.println(b);
        }
    }

```

* 能够理解注解的概述

* Java 注解（Annotation）又称 Java 标注，是 JDK5.0 引入的一种注释机制。
* Java 语言中的类、方法、变量、参数和包等都可以被标注

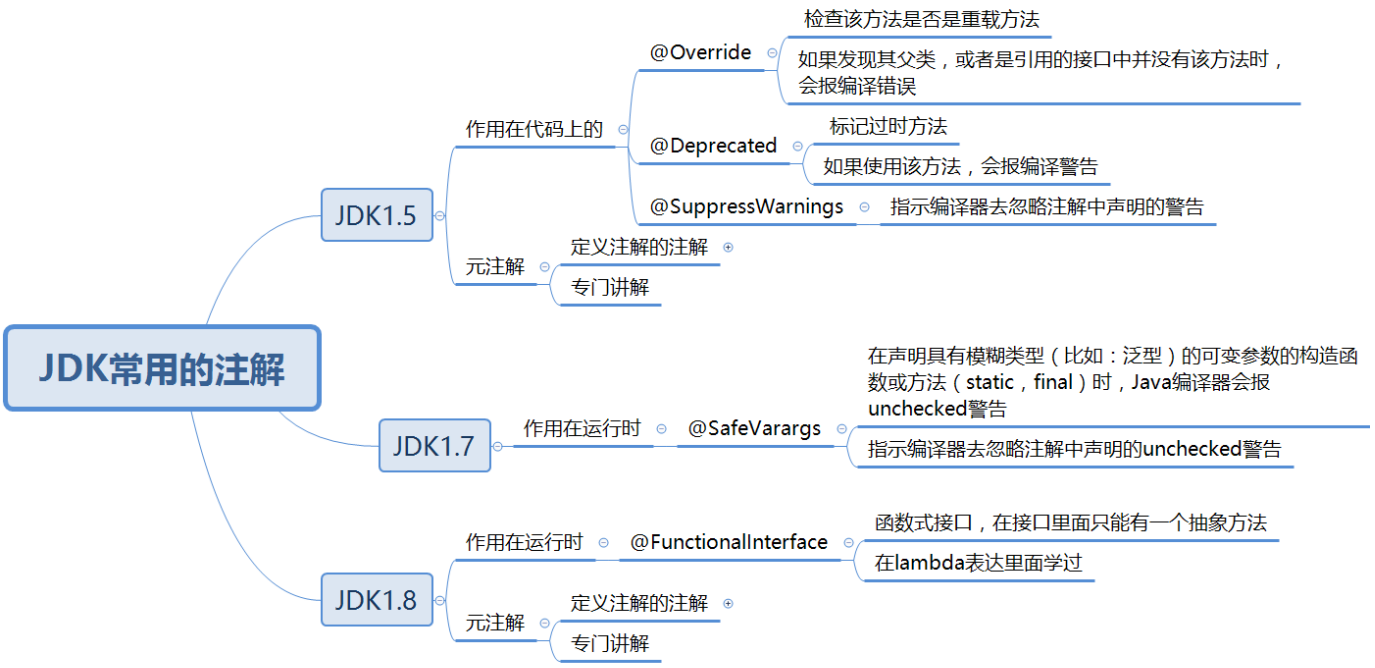
* 和注释不一样，注释是程序员写的，给程序员看，而Java 注解可以通过反射获取标注内容

* 在编译器生成类文件时，注解可以被嵌入到字节码中。Java 虚拟机可以保留注解内容，在运行时可以获取到注解内容 。

* Java支持在编码，编译，运行的时期自定义注解，从而实现不同功能

* 在开发中，注解可以用来替换配置文件，简化代码的开发，实现具体的功能，现在JAVA开发是注解的天下。

* 能够掌握jdk常用的注解



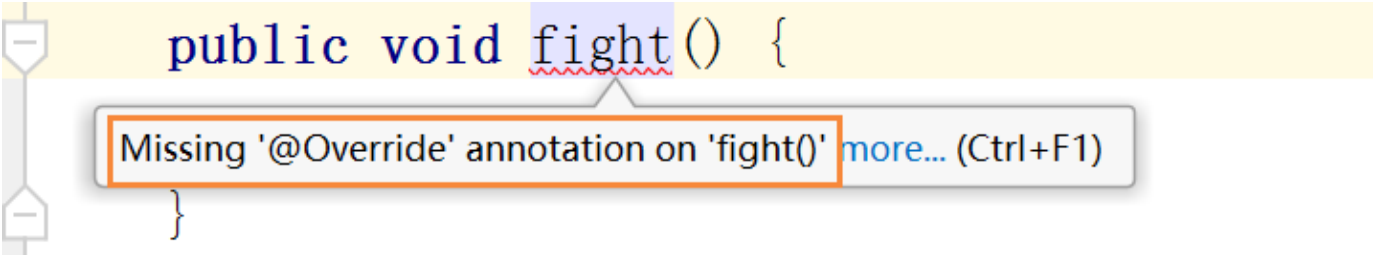
* @Override演示

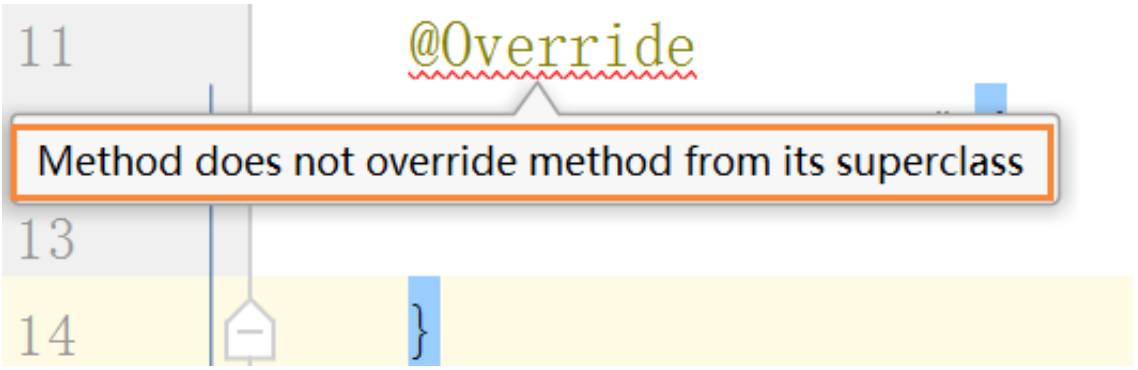
```
* 基础代码
public abstract class Pet {
    public abstract void fight();
}
public class Dragon extends Pet {
```

```
    @Override
    public void fight() {

    }
    public void eat(){

    }
}
* 假如在fight去掉 @Override，会报编译错误
* 假如在eat加上@Override也会报编译错误
```



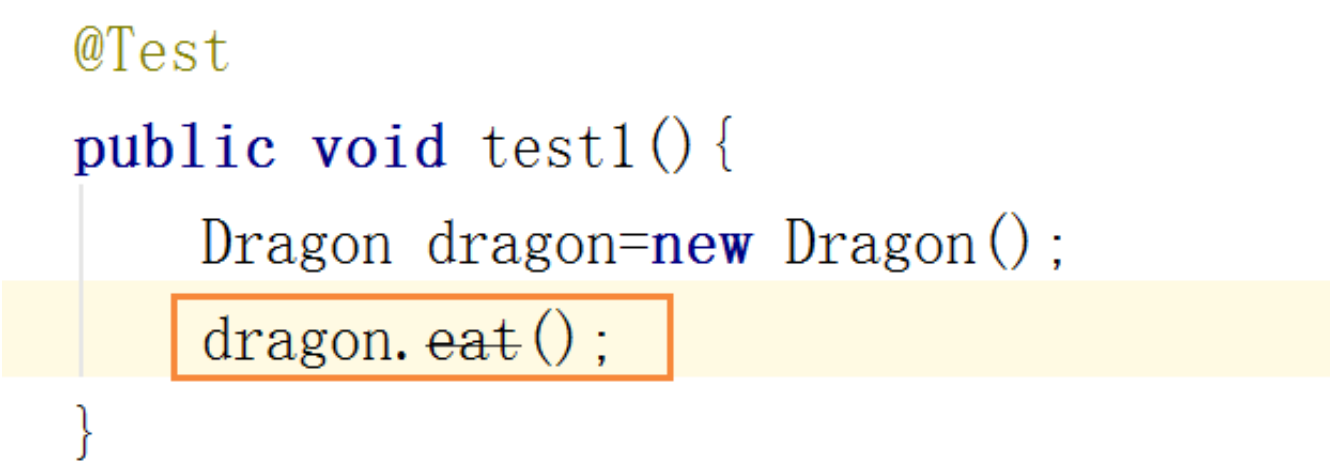


* @Deprecated演示

```
* 代码
    @Deprecated
    public void eat(){

    }
  @Test
  public void test1(){
        Dragon dragon=new Dragon();
        dragon.eat();
```

```
10    }
11
12 * 在eat添加@Deprecated注解，在test1调用的时候，会有一条划线
```



* @SuppressWarnings演示

```
1  * 代码
2     @Test
3     @SuppressWarnings("unchecked")
4     public void test2(){
5         ArrayList list = new ArrayList();
6         list.add("xiaohei");
7         list.add(1);
8     }
9
10    * 集合中没有指定泛型，添加会报unchecked，加上@SuppressWarnings抑制警告
```

```java
@Test
public void test2(){
    ArrayList list = new ArrayList();
    list.add("xiaohei");
    list.add(1);
}
```

```java
@Test
@SuppressWarnings("unchecked")
public void test2(){
    ArrayList list = new ArrayList();
    list.add("xiaohei");
    list.add(1);
}
```

* @SafeVarargs演示

```
* 代码
    @SafeVarargs
    Dragon(T... ts){

    }

    @SafeVarargs
    public final void say2(T... ts){

    }
    @SuppressWarnings("unchecked")
    public void say3(T... ts){

    }
```

```java
Dragon(T... ts){

}
```

```java
@SafeVarargs
Dragon(T... ts){

}
```

```java
public void say3(T... ts){

}
```

```java
@SuppressWarnings("unchecked")
public void say3(T... ts){

}
```

```java
public final void say2(T... ts){

}
```

```java
@SafeVarargs
public final void say2(T... ts){

}
```

* @FunctionalInterface

```
* 代码
@FunctionalInterface
public interface MInterface {
    void say();
}
```

```
@FunctionalInterface
public interface MInterface {
    void say();
}
```

```
@FunctionalInterface
public interface MInterface {
    void say();
    void say1();
}
```

* 能够掌握LomBok常用的注解

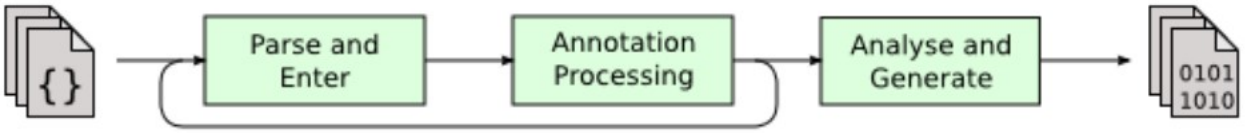
* lombok概述

* lombok是一个java库，它自动插入到编辑器和构建工具中，增强java的功能。

* 使用注解替代了编写getter，setter，equals，hashcode,toString等代码

* 自动化创建日志变量

* lombok的原理



* javac对源代码进行分析，生成了一棵抽象语法树（AST）

* 运行过程中调用实现了的Lombok程序

* 此时Lombok就对第一步骤得到的AST进行处理，找到@Data注解所在类对应的语法树（AST），然后修改该语法树（AST），增加getter和setter方法定义的相应树节点

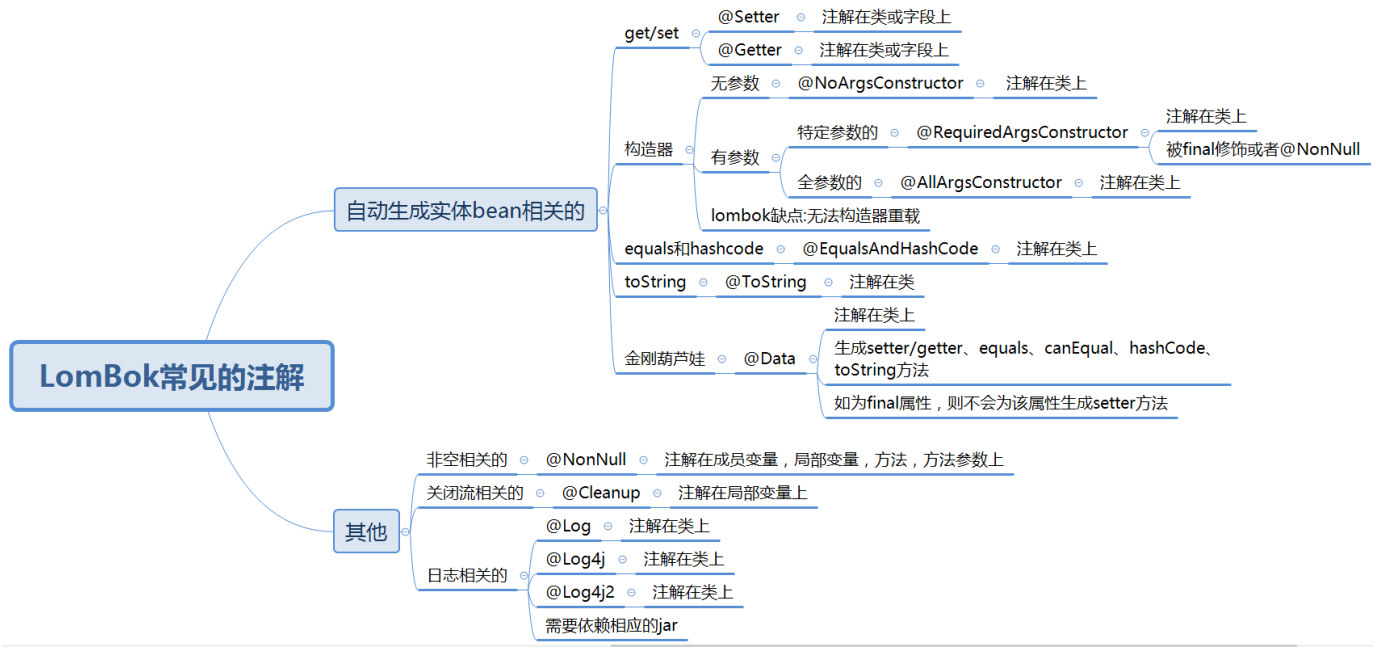
* javac使用修改后的抽象语法树（AST）生成字节码文件，即给class增加新的节点

* 简而言之：lombok其实是在编译阶段，根据标记的注解，动态生成新class文件

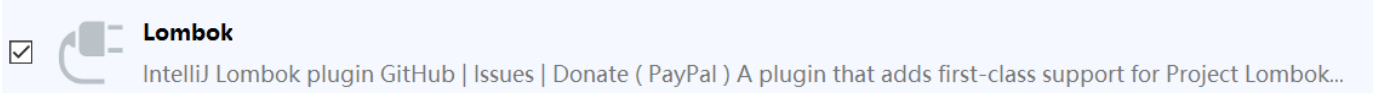
* lombok的优缺点

* 优点：
  * 能通过注解的形式自动代码，提高的开发效率
  * 让代码变得简洁
  * 便于代码的维护
* 缺点：
  * 不支持多种参数构造器的重载
  * 降低了源代码的可读性和完整性，降低了阅读源代码的舒适度

* lombok常用的注解



* 开发步骤
  * 导入lombok.jar
  * 下载lombok插件



```
* @Data
@Data
public class Book1 {
    private int id;
    private String name;
    private double price;
    private String author;
}
* 查看生成Book1.class文件，
  发现自动生成setter，getter，toString，equals,hashcode，canEquals
* 其他类似这样测试，查看生成的class文件对比
```