**代码规范**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 类别 | 内容 | | 说明 | 状态 | 正确示例 | 错误示例 |
| 格式规范 | 缩进 | 缩进要求采用4个空格,大括号的开始 在代码块开始的行尾，闭合在和代码块同一缩进的行首 | 提高可读性 | 建议 | public class UserBean {  private String username;  } | public class UserBean  {  private String username;  } |
| 空格 | 变量、类、常量数据和函数在其类型，修饰名称之间适当空格并据情况对齐 | 提高可读性 | 建议 | public class UserBean {  private String username;  private String password;  private int status;  } | public class UserBean {  private String username;  private int status;  private String password;  } |
| 对齐 | 关系密切的行应对齐，对齐包括类型、修饰、名称、参数等各部分对齐 | 提高可读性 | 建议 | public class UserBean {  private String username;  private String password;  } | public class UserBean {  private String username;  private String password;  } |
| 空行 | 方法之间需要添加空­­­行 | 提高可读性 | 建议 | Public String getUsernam(){  return username;  }  public void setUsername(){  this.username = user;  } | public String getUser (){  return username;  }  public void setUser (){  this.user = user;  } |
| 注释 | 代码中加一些说明 | 提高可读性 | 建议 | public class UserBean {  private int status;//用户身份  } | public class UserBean {  private int status;  } |
| 命名规范 | 通用规范 | 命名见名知意，以增加代码可读性 | 提高可读性 | 规则 | String username; | String a; |
| 包命名 | 包的名称全部采用小写字母 | Java的一般规则 | 规则 | com.java.dao | com.java.Dao |
| 方法命名 | 方法的命名应为动词或动词性词组，首字母小写，第二个及以后的单词首字母大写 | 将提高代码的易读性。 | 建议 | public String getUsername(); | public int a(); |
| 变量命名 | 名词或名词性词组，首字母小写，第二个及以后的单词首字母大写 | Java的一般规则。 | 建议 | String userName; | String username; |
| 常量命名 | 每一个单词都要大写，单词之间用"\_"分开 | Java的一般规则 | 建议 | final static int MIN\_WIDTH = 4 | final static int minwindth = 4 |
| 注释 | 块注释 | 较长的代码块要用/\*---start---\*/和/\*---end---\*/包围 | 提高代码可读性 | 规则 | /\*---start---\*/  //获取信息  String name = user.getUsername();  String password = user.getPassword();  /\*---end---\*/ | String name = user.getUsername();  String password = user.getPassword(); |
| 行注释 | 用//写在行尾 | 提高代码可读性 |  | String name = user.getUsername();//获取用户名 | String name = user.getUsername(); |
| 编写规范 | {}的使用 | 每个if while for语句都不要省略{} | 可以增加代码的可读性和可维护性。 | 规范 | if （a>b）{  a++;  } | if （a>b）  a++; |
| 减少代码嵌套层次 | 代码嵌套层次应小于等于三层 | 增加代码可读性，有利于以后的代码维护工作。 | 建议 | if (a>b&&b>c){  doJobA();  }  if (a>b&&c>b){  doJobB();  }  if (a<=b&&c<b&&a<c){  doJobC();  } | if (a>b){  if (b>c){  doJobA();  }else if (b<c){  doJobB();  }else{  if (b>c){  if (a<c){  doJobC();  }  }  }  } |
| 变量的使用 | 变量的声明、初始化和使用尽量放到一起 | 降低出错率 | 建议 | //do something without orderNum here  int orderNum = getOrderNum();  call (orderNum); | int orderNum = getOrderNum();  //do something without orderNum here  call (orderNum); |