1. 通用规范：
   1. 严禁使用无意义字符、拼音、拼音缩写表示变量，如a, yonghu, SJKLJ....，应使用单词或单词的组合;
   2. 整个项目每一个实体只能存在一个统一名字，严禁一时出现ApplicationInfo一时出现ProgramInfo、Information与Message表示同一个实体等现象;
   3. 待补充;
2. 变量规范：
   1. 统一采用小驼峰命名法：如overTimeException, isHumanItem;
   2. 集合变量命名只需在实体名称后加s：如 item => items, user => users;
   3. 布尔型变量命名：
      1. 表示能力：XXX + able，如：enable;
      2. 表示是否：XXX，如：default;
      3. 表示状态：XXX + ed 或 ing，如：connected, stopped, calling;
   4. 定义自定义类的变量时，直接把类名首字母改成小写作为变量名： 如 private UserService userService;
   5. 待补充;
3. 常量规范：
   1. 禁止在代码内出现“魔法值”，如 if(user.age > 30 && user.type == 2){...}，应该把“30”，“2”等数值定义成常量;
   2. 统一采用全大写加下划线命名法：如，USER\_YOUNG\_AGE, USER\_YOUNG\_OLD, USER\_TYPE\_VIP, USER\_TYPE\_NORMAL;
   3. 待补充;
4. 函数规范：
   1. 采用小驼峰命名法；
   2. 应使用动词、动宾结构或系表结构作为函数名：如getTime(), setUser, call(), isConnected();
   3. 几个典型类型的函数命名：
      1. 获取一个对象或属性：getXXX();
      2. 获取一个布尔型属性：isXXX();
      3. 获取一个集合：listXXX();适用于List, Set
      4. 设置属性：setXXX();
      5. 添加对象：addXXX();
      6. 删除对象：deleteXXX();
      7. 更新对象：updateXXX();
   4. 参数命名规范与变量一致;
   5. 一个函数应该只完成一个功能，如果函数超过50行，请检查是否有可复用代码，若有则提取成私有函数，并调用。
   6. 待补充;
5. 类与封装规范：
   1. 采用大驼峰命名法：如，AppointmentInfo, UserController;
   2. 除非特殊情况下，类均为public，成员变量均为private，并且每个成员变量应具有getXXX()，setXXX()方法，并且严禁在该两种方法内添加业务逻辑;
   3. 成员常量均为 private static final XXX\_XXX\_XXX = XXX，若该常量被多个类引用，则单独建立一个常量包，并且建立相应的常量类，把常量定义在内，修饰符改为public;
   4. 重写父类方法必须加注解：@Override;
   5. 注解尽量写在变量上而不是方法上，提高代码可读性（如在springmvc中，@AutoWired写在成员变量上而不写在setXXX()方法上）;
   6. 严禁出现“局部变量未被引用”、“私有函数未被调用”、“包或类未被使用”等警告;
   7. 封装修饰符写法：
      1. 在本类使用的变量、方法必须为private;
      2. 打算被继承的变量、方法必须为protected;
      3. 打算被包内其他类调用的方法必须为默认（空修饰）;
      4. 打算被全局调用的方法必须为public;
      5. 严禁出现默认或public的变量，但可以出现public的常量;
   8. 类内的编码结构要按以下顺序从上到下编码（每部分并不是必须有的）：
      1. 常量;
      2. 成员变量;
      3. 公有构造方法;
      4. 公有方法;
      5. 非公有方法;
      6. get/set方法;
   9. 待补充;
6. 注释与格式规范：
   1. 严禁过度使用注释，或过度懒得写注释;
   2. 使用行注释时建议写在上边，而不写在右边，代码更加美观;
   3. 公有常量、公有方法必须有块注释，可（快捷键：Shift + Alt + J）快速生成;
   4. 每个类必须有块注释，内容至少为两部分：作者、类的功能描述;
   5. 每个公有方法必须有块注释，内容为关键参数描述、功能描述、返回值描述（若很关键）
   6. 函数内，把若干行代码作为一个子操作，每个子操作间空一行，每个关键子操作使用行注释在代码上方进行说明，便于日后维护;
   7. 间隔规范写法：
      1. package与import之间空一行;
      2. import与类之间空两行;
      3. 每个类内第一行为空;
      4. 写类时，按照四h)的顺序进行编码，每个常量（变量）之间空一行;
      5. 若常量（变量）之间存在行注释或注解，可以不空行;
      6. 常量区域、变量区域、方法之间均空两行（自动生成的get/set方法在最后可以不理会）
   8. 任何代码块使用这样的花括号方式，即第一个花括号在函数名右边：

private void getTime(){

//do Sometings...

}

* 1. 任何if, while, for严禁不添加花括号;
  2. 在if.try,while,finally,foreach，方法，类，代码块等区域，一定要知道缩进一个Tab键，大大提高代码可读性;
  3. 凡是能使用foreach的地方尽量使用，避免使用while, for，提高代码的优雅度;
  4. 凡是能throws 异常的方法，尽量不去try/catch异常，提高代码的优雅度;
  5. 凡是能逐步判断并中断的方法，严禁嵌套判断：

反例：

if(a == 1 ){

if(b == 2){

if(c == 3){

doSometings();

}

}

}

return;

正例：

if(a != 1){

return;

}

if(b != 2){

return;

}

if(c != 3){

return;

}

doSomethings();

return;

* 1. 待补充;

1. 其他规范：
   1. 严禁抛出非自定义异常类对象;如，throw new RuntimeException(“hello bug”);
   2. 多写非空判断，提高代码健壮性，节省调试时间;
   3. 在开发有较高安全需求服务器端应用时，永远不要相信客户端传来的数据，必须做好各种校验;
   4. 项目开始时必须统一包名、各层名称，可参考规范：
      1. 控制器：Controller;
      2. 业务层：Service;
      3. 持久层：DAO;
      4. 实体：Entity;
      5. 表示层实体：VO;
      6. 业务层实体：BO;
      7. 工具：Util;
      8. 拦截器：Interceptor;
   5. 数据库表名与实体类一致，字段名与实体类属性名一致;
   6. 数据库表设计必须达到第三范式;
   7. 项目需求分析时必须有UML各种图其中的一种或几种;
   8. 在复杂的业务操作编写前，必须画算法流程图;
   9. 项目进行中需要有一方（前端、后端）提供接口文档，并且定期维护;
   10. 后端需要公开数据库表设计文档;
   11. 待补充;