

# 实现 (Implementation)

# outline

- 编码
- 配置管理
- 质量管理
- 软件度量

# 概述

- 编码：把软件设计结果翻译成用某种程序设计语言书写的程序。
  - a natural consequence of design
  - programming language characteristics
  - good programming practice
  - coding style

# 程序设计语言

- 语法（syntax）用来表示构成语言的各个记号之间的组合规则。
  - for（表达式1；表达式2；表达式3）语句
  - 语法中不涉及到这些记号的含义，也不涉及使用者

# 程序设计语言

- 语义（semantic）用来表示按照各种表示方式所表示的各个记号的特定含义，但它不涉及到使用者。
  - for语句中：表达式1表示循环初值；表达式2表示循环条件；表达式3表示循环的增量；语句为循环体。整个语句的语义是：
    - ✓ （1）计算表达式1
    - ✓ （2）计算表达式2，若计算结果为0，则终止循环；否则转（3）
    - ✓ （3）执行循环体
    - ✓ （4）计算表达式3
    - ✓ （5）转向（2）

# 程序设计语言

- 语用（pragmatic）用来表示构成语言的各个记号和使用者的关系。
  - 语言是否允许递归？是否要规定递归层数的上界？这种上界如何确定？这些都属于语用上的问题。

# 选择的标准

- 系统用户的要求
- 测试和维护的要求
- 工程规模
- 程序员的知识
- 软件可移植性要求
- 软件的应用领域


# 良好编程实践

- 变量的命名要有意义且一致
  - 有意义和一致主要是对将来的维护程序员而言
    - *averageFreq, frequencyMaximum, minFr, frqncyTotl*
    - *average & mean*
  - 变量名中单词的顺序要一致
    - *frequencyMaximum & minimumFrequency*
    - *frequencyAverage, frequencyMaximum, frequencyMinimum, frequencyTotal*
    - *averageFrequency, maximumFrequency, minimumFrequency, totalFrequency*
  - 命名约定
    - Hungarian Naming Conventions
    - *ptrChTmp(pointer+character+temporary)*



# 良好编程实践

## ■ 代码的self-documenting

- ❑ 程序员根据命名约定精心选择变量名，认为无需注释了
- ❑ 问题是其他程序员，包括软件质量保证小组、维护程序员能否读懂程序
- ❑ *xCoordinateOfPositionOfRobotArm*  *xCoord*
- ❑ 在代码的起始位置加上变量名的解释， prologue comments
- ❑ 在代码内部使用inline comments

# 例

```
/**
 * Robot movement.
 * @param xCoord the x coordinate of the position of the robot arm.
 * @param yCoord the y coordinate of the position of the robot arm.
 */
private void robotMovement(double xCoord, double yCoord)
{
    Robot waterRobot = new Robot();
    .....
    waterRobot.move();           //move the robot
    .....
}
```

# 良好编程实践

## ■ 使用参数

```
public class GeneticAlgorithm {  
    int generation = 1;  
    while(generation <= 10000) {  
        doCrossOver();  
        doMutate();  
        generation++;  
    }  
}
```

```
public class GeneticAlgorithm {  
    private static final int MaxGenerations = 10000;  
    int generation = 1;  
    while(generation <= MaxGenerations) {  
        doCrossOver();  
        doMuatae();  
        generation++;  
    }  
}
```

# 良好编程实践

## ■ 使用参数

**config.xml**

```
<properties>
  <entry key="PopulationSize">1000</entry>
  <entry key="MaxGenerations">10000</entry>
</properties>
```

```
public class GeneticAlgorithm {
    private static final int MaxGenerations =
        Config.getMaxGenerations();
    int generation = 1;
    while(generation <= MaxGenerations) {
        doCrossOver();
        doMuatae();
        generation++;
    }
}
```

# 编码风格

程序实际上也是一种供人阅读的文章，有一个文章的风格问题，应该使程序具有良好的风格。

## 1、程序内部的文档

- 恰当的标识符
- 适当的注解
- 程序的视觉组织
  - 空格
  - 阶梯形式

# 编码风格

## 2、 数据说明

- 在设计阶段已经确定了数据结构的组织及其复杂性。在编写程序时，则需要注意数据说明的风格。
- 为了使程序中数据说明更易于理解和维护，必须注意以下几点：
  - 数据说明的次序应该标准化。
  - 说明语句中变量安排有序化。
  - 使用注释说明复杂数据结构。

# 编码风格

## 3、语句构造

- 语句构造力求简单、直接，不能为了片面追求效率而使语句复杂化。
  - 在一行内只写一条语句；
  - 尽量避免复杂的条件测试；
  - 尽量减少对“非”条件的测试；
  - 避免大量使用循环嵌套和条件嵌套；
  - 利用括号使表达式次序清晰。

# Case Analysis

编码规范等文档示例