# 请大家阅读文档时，在视图里勾选导航窗格，在左边显示章节目录方便浏览。

# 1、实验编程第二题每个类的数据成员和方法说明

Package homework.ch11\_13.p3

|  |  |  |
| --- | --- | --- |
| Class Summary | | **Page** |
| [**Course**](#b16) | 课程类 | 34 |
| [**CourseTest**](#b19) | 测试类 | 34 |
| [**Faculty**](#b34) | 教工类 | 41 |
| [**Person**](#b46) | 父类 | 48 |
| [**Student**](#b61) | 学生类 | **错误!未定义书签。** |

## 1.1 Class Person

Class Person

[**homework.ch11\_13.p3**](#b2)

java.lang.Object

**homework.ch11\_13.p3.Person**

All Implemented Interfaces:

Cloneable

Direct Known Subclasses:

[Faculty](#b34), [Student](#b61)

public class **Person**

extends Object

implements Cloneable

父类

|  |  |  |
| --- | --- | --- |
| Field Summary | | **Page** |
| private int | [**age**](#b36)  年龄 | 3 |
| private String | [**name**](#b35)  姓名 | 3 |

|  |  |
| --- | --- |
| Constructor Summary | **Page** |
| [**Person**](#b37)()  缺省构造函数 | 3 |
| [**Person**](#b38)(String name, int age)  构造函数 | 4 |

|  |  |  |
| --- | --- | --- |
| Method Summary | | **Page** |
| Object | [**clone**](#b45)()  Person的深拷贝克隆 | 5 |
| boolean | [**equals**](#b44)(Object obj)  比较二个Person对象的内容是否相等 | 5 |
| int | [**getAge**](#b41)()  获取年年 | 4 |
| String | [**getName**](#b39)()  获取姓名 | 4 |
| void | [**setAge**](#b42)(int age)  设置年龄 | 5 |
| void | [**setName**](#b40)(String name)  设置姓名 | 4 |
| String | [**toString**](#b43)()  覆盖toString | 5 |

Field Detail

name

private String **name**

姓名

age

private int **age**

年龄

Constructor Detail

Person

public **Person**()

缺省构造函数

Person

public **Person**(String name,  
              int age)

构造函数

**Parameters:**

name - 姓名

age - 年龄

Method Detail

getName

public String **getName**()

获取姓名

**Returns:**

姓名

setName

public void **setName**(String name)

设置姓名

**Parameters:**

name - 姓名

getAge

public int **getAge**()

获取年年

**Returns:**

年龄

setAge

public void **setAge**(int age)

设置年龄

**Parameters:**

age - 年龄

toString

public String **toString**()

覆盖toString

**Overrides:**

toString in class Object

**Returns:**

描述Person对象信息的字符串

equals

public boolean **equals**(Object obj)

比较二个Person对象的内容是否相等

**Overrides:**

equals in class Object

**Parameters:**

obj - 另外一个对象

**Returns:**

当二个对象所有数据成员的内容相等，返回true

clone

public Object **clone**()  
             throws CloneNotSupportedException

Person的深拷贝克隆

**Overrides:**

clone in class Object

**Returns:**

克隆出来的对象

**Throws:**

CloneNotSupportedException - 可能抛出的异常

## 1.2 Class Student

Class Student

[**homework.ch11\_13.p3**](#b2)

java.lang.Object

[homework.ch11\_13.p3.Person](#b46)

**homework.ch11\_13.p3.Student**

All Implemented Interfaces:

Cloneable

public class **Student**

extends [Person](#b46)

学生类

|  |  |  |
| --- | --- | --- |
| Field Summary | | **Page** |
| private String | [**classNo**](#b49)  所在班级 | 9 |
| private String | [**department**](#b48)  s所在院系 | 8 |
| private int | [**studentId**](#b47)  学生Id | 8 |

|  |  |
| --- | --- |
| Constructor Summary | **Page** |
| [**Student**](#b50)()  缺省构造函数 | 9 |
| [**Student**](#b51)(String name, int age, int studentId, String department, String classNo)  构造函数 | 9 |

|  |  |  |
| --- | --- | --- |
| Method Summary | | **Page** |
| Object | [**clone**](#b60)()  Student的深拷贝克隆 | 12 |
| boolean | [**equals**](#b59)(Object obj)  比较二个Student对象的内容是否相等 | 11 |
| String | [**getClassNo**](#b56)()  获取所在班级 | 10 |
| String | [**getDepartment**](#b54)()  获取所在院系 | 10 |
| int | [**getStudentId**](#b52)()  获取学生Id | 9 |
| void | [**setClassNo**](#b57)(String classNo)  设置所在班级 | 11 |
| void | [**setDepartment**](#b55)(String department)  设置所在院系 | 10 |
| void | [**setStudentId**](#b53)(int studentId)  设置学生Id | 10 |
| String | [**toString**](#b58)()  覆盖toString | 11 |

|  |
| --- |
| **Methods inherited from class homework.ch11\_13.p3.**[**Person**](#b46) |
| [getAge](#b41), [getName](#b39), [setAge](#b42), [setName](#b40) |

Field Detail

studentId

private int **studentId**

学生Id

department

private String **department**

s所在院系

classNo

private String **classNo**

所在班级

Constructor Detail

Student

public **Student**()

缺省构造函数

Student

public **Student**(String name,  
               int age,  
               int studentId,  
               String department,  
               String classNo)

构造函数

**Parameters:**

name - 姓名

age - 年龄

studentId - 学号

department - 所在院系

classNo - 所在班级

Method Detail

getStudentId

public int **getStudentId**()

获取学生Id

**Returns:**

学生Id

setStudentId

public void **setStudentId**(int studentId)

设置学生Id

**Parameters:**

studentId - 学生Id

getDepartment

public String **getDepartment**()

获取所在院系

**Returns:**

所在院系

setDepartment

public void **setDepartment**(String department)

设置所在院系

**Parameters:**

department - 所在院系

getClassNo

public String **getClassNo**()

获取所在班级

**Returns:**

所在班级

setClassNo

public void **setClassNo**(String classNo)

设置所在班级

**Parameters:**

classNo - 所在班级

toString

public String **toString**()

覆盖toString

**Overrides:**

[toString](#b43) in class [Person](#b46)

**Returns:**

描述Student对象信息的字符串

equals

public boolean **equals**(Object obj)

比较二个Student对象的内容是否相等

**Overrides:**

[equals](#b44) in class [Person](#b46)

**Parameters:**

obj - 另外一个对象

**Returns:**

当二个对象所有数据成员的内容相等，返回true

clone

public Object **clone**()  
             throws CloneNotSupportedException

Student的深拷贝克隆

**Overrides:**

[clone](#b45) in class [Person](#b46)

**Returns:**

克隆出来的对象

**Throws:**

CloneNotSupportedException - 可能抛出的异常

## 1.3 Class Faculty

Class Faculty

[**homework.ch11\_13.p3**](#b2)

java.lang.Object

[homework.ch11\_13.p3.Person](#b46)

**homework.ch11\_13.p3.Faculty**

All Implemented Interfaces:

Cloneable

public class **Faculty**

extends [Person](#b46)

教工类

|  |  |  |
| --- | --- | --- |
| Field Summary | | **Page** |
| private String | [**email**](#b22)  邮箱 | 15 |
| private int | [**facultyId**](#b20)  教工Id | 14 |
| private String | [**title**](#b21)  职称 | 14 |

|  |  |
| --- | --- |
| Constructor Summary | **Page** |
| [**Faculty**](#b23)()  缺省构造函数 | 15 |
| [**Faculty**](#b24)(String name, int age, int facultyId, String title, String email) | 15 |

|  |  |  |
| --- | --- | --- |
| Method Summary | | **Page** |
| Object | [**clone**](#b33)()  Faculty的深拷贝克隆 | 17 |
| boolean | [**equals**](#b32)(Object obj)  比较二个Faculty对象的内容是否相等 | 17 |
| String | [**getEmail**](#b29)()  获取邮箱 | 16 |
| int | [**getFacultyId**](#b25)()  获取教工Id | 15 |
| String | [**getTitle**](#b27)()  获取职称 | 16 |
| void | [**setEmail**](#b30)(String email)  设置邮箱 | 17 |
| void | [**setFacultyId**](#b26)(int facultyId)  设置教工Id | 16 |
| void | [**setTitle**](#b28)(String title)  设置职称 | 16 |
| String | [**toString**](#b31)()  覆盖toString | 17 |

|  |
| --- |
| **Methods inherited from class homework.ch11\_13.p3.**[**Person**](#b46) |
| [getAge](#b41), [getName](#b39), [setAge](#b42), [setName](#b40) |

Field Detail

facultyId

private int **facultyId**

教工Id

title

private String **title**

职称

email

private String **email**

邮箱

Constructor Detail

Faculty

public **Faculty**()

缺省构造函数

Faculty

public **Faculty**(String name,  
               int age,  
               int facultyId,  
               String title,  
               String email)

**Parameters:**

name - 姓名

age - 年龄

facultyId - 教工Id

title - 职称

email - 邮箱

Method Detail

getFacultyId

public int **getFacultyId**()

获取教工Id

**Returns:**

教工Id

setFacultyId

public void **setFacultyId**(int facultyId)

设置教工Id

**Parameters:**

facultyId - 教工Id

getTitle

public String **getTitle**()

获取职称

**Returns:**

职称

setTitle

public void **setTitle**(String title)

设置职称

**Parameters:**

title - 职称

getEmail

public String **getEmail**()

获取邮箱

**Returns:**

邮箱

setEmail

public void **setEmail**(String email)

设置邮箱

**Parameters:**

email - 邮箱

toString

public String **toString**()

覆盖toString

**Overrides:**

[toString](#b43) in class [Person](#b46)

**Returns:**

描述Faculty对象信息的字符串

equals

public boolean **equals**(Object obj)

比较二个Faculty对象的内容是否相等

**Overrides:**

[equals](#b44) in class [Person](#b46)

**Parameters:**

obj - 另外一个对象

**Returns:**

当二个对象所有数据成员的内容相等，返回true

clone

public Object **clone**()  
             throws CloneNotSupportedException

Faculty的深拷贝克隆

**Overrides:**

[clone](#b45) in class [Person](#b46)

**Returns:**

克隆出来的对象

**Throws:**

CloneNotSupportedException - 可能抛出的异常

## 1.4 Class Course

Class Course

[**homework.ch11\_13.p3**](#b2)

java.lang.Object

**homework.ch11\_13.p3.Course**

All Implemented Interfaces:

Cloneable

public class **Course**

extends Object

implements Cloneable

课程类

|  |  |  |
| --- | --- | --- |
| Field Summary | | **Page** |
| private String | [**courseName**](#b3)  课程名称 | 20 |
| private List<[Person](#b46)> | [**students**](#b4)  选修课程的学生列表，保存在ArrayList里 | 20 |
| private [Person](#b46) | [**teacher**](#b5)  课程的授课老师 | 20 |

|  |  |
| --- | --- |
| Constructor Summary | **Page** |
| [**Course**](#b6)(String courseName, [Person](#b46) teacher)  构造函数 | 21 |

|  |  |  |
| --- | --- | --- |
| Method Summary | | **Page** |
| Object | [**clone**](#b13)()  Course的深拷贝克隆 | 22 |
| boolean | [**equals**](#b15)(Object obj)  比较二个Course对象的内容是否相等 | 23 |
| String | [**getCourseName**](#b8)()  获取课程名称 | 21 |
| int | [**getNumberOfStudent**](#b12)()  获取选修课程的学生总数 | 22 |
| List<[Person](#b46)> | [**getStudents**](#b9)()  获取课程的学生名单 这个方法纯粹是为了测试课程对象的深拷贝。 | 21 |
| [Person](#b46) | [**getTeacher**](#b10)()  获取课程授课老师 | 22 |
| void | [**register**](#b7)([Person](#b46) s)  选修课程。 | 21 |
| String | [**toString**](#b14)()  覆盖toString | 23 |
| void | [**unregister**](#b11)([Person](#b46) s)  取消选修 应该把取消选修的学生从学生名单里删除 | 22 |

Field Detail

courseName

private String **courseName**

课程名称

students

private List<[Person](#b46)> **students**

选修课程的学生列表，保存在ArrayList里

teacher

private [Person](#b46) **teacher**

课程的授课老师

Constructor Detail

Course

public **Course**(String courseName,  
              [Person](#b46) teacher)

构造函数

**Parameters:**

courseName - 课程名称

teacher - 授课老师

Method Detail

register

public void **register**([Person](#b46) s)

选修课程。 应该把选修的学生加入到学生列表里。注意同一个学生只能选修一次，内部的ArrayList里不能出现重复的学生

**Parameters:**

s - 选修课程的学生。

getCourseName

public String **getCourseName**()

获取课程名称

**Returns:**

课程名称

getStudents

public List<[Person](#b46)> **getStudents**()

获取课程的学生名单 这个方法纯粹是为了测试课程对象的深拷贝。实际场景下不应该返回学生名单，破坏了封装性

**Returns:**

课程的学生名单

getTeacher

public [Person](#b46) **getTeacher**()

获取课程授课老师

**Returns:**

课程授课老师

unregister

public void **unregister**([Person](#b46) s)

取消选修 应该把取消选修的学生从学生名单里删除

**Parameters:**

s - 取消选修的学生

getNumberOfStudent

public int **getNumberOfStudent**()

获取选修课程的学生总数

**Returns:**

选修课程的学生总数

clone

public Object **clone**()  
             throws CloneNotSupportedException

Course的深拷贝克隆

**Overrides:**

clone in class Object

**Returns:**

克隆出来的对象

**Throws:**

CloneNotSupportedException - 可能抛出的异常

toString

public String **toString**()

覆盖toString

**Overrides:**

toString in class Object

**Returns:**

描述Course对象信息的字符串(应该包括课程名称、教师的详细信息，每个学生的详细信息，学生总数)

equals

public boolean **equals**(Object obj)

比较二个Course对象的内容是否相等

**Overrides:**

equals in class Object

**Parameters:**

obj - 另外一个对象

**Returns:**

当二个对象当二个对象所有数据成员的内容相等，返回true。注意学生名单内容也要相等（元素个数相等，每个List里的每个对象在另外一个List里都有唯一的内容相等的元素，但次序可以不同）

## 1.5 Class CourseTest

Class CourseTest

[**homework.ch11\_13.p3**](#b2)

java.lang.Object

**homework.ch11\_13.p3.CourseTest**

public class **CourseTest**

extends Object

程序测试类

|  |  |
| --- | --- |
| Constructor Summary | **Page** |
| [**CourseTest**](#b17)() | 24 |

|  |  |  |
| --- | --- | --- |
| Method Summary | | **Page** |
| static void | [**main**](#b18)(String[] args)  程序入口函数 在这里实例化教师对象、课程对象。 | 24 |

Constructor Detail

CourseTest

public **CourseTest**()

Method Detail

main

public static void **main**(String[] args)  
                 throws CloneNotSupportedException

程序入口函数 在这里实例化教师对象、课程对象。同时实例化多个学生对象向课程注册。 需要创建一个Course数组，包含至少二门课程，每门课程至少注册三名学生。最后打印出每门课程的详细信息。 同时测试Person、Student、Faculty、Course的深拷贝功能，深拷贝测试包括： 克隆出来的对象和源对象内容相等； 克隆出来的对象和源对象所有引用类型数据成员指向的是不同对象。

**Parameters:**

args - 命令行参数

**Throws:**

CloneNotSupportedException - 可能抛出的异常

# 2、实验编程第三题每个类的数据成员和方法说明

|  |  |  |
| --- | --- | --- |
| Package Summary | | **Page** |
| [**homework.ch11\_13.p4**](#b2) |  | **错误!未定义书签。** |

Package homework.ch11\_13.p4

|  |  |  |
| --- | --- | --- |
| Interface Summary | | **Page** |
| [***Iterator***](#b52) | 迭代器接口，用于遍历组件树里的每一个组件. | 9 |

|  |  |  |
| --- | --- | --- |
| Class Summary | | **Page** |
| [**AtomicComponent**](#b9) | 原子组件类，不包含任何子组件 | 21 |
| [**Component**](#b27) | 计算机组件的抽象类，任何一个具体的组件键盘、鼠标、主板、主机都是Component。 | 16 |
| [**ComponentFactory**](#b30) | 组件的对象工厂, 由对象工厂返回对象 | 17 |
| [**ComponentList**](#b35) | Component对象的容器类，用于保存复合组件的子组件.从ArrayList派生，实现了自定义Iterator接口. | 3 |
| [**CompositeComponent**](#b44) | 复合组件，包含子组件 | 5 |
| [**CompositeIterator**](#b49) | 复合迭代器, 用于复合组件的迭代 | 9 |
| [**NullIterator**](#b56) | 空迭代器，这个迭代器的hasNext()方法永远返回false, next()方法永远返回null. | 10 |
| [**Test**](#b59) | 测试类 | 11 |

## 2.1 Class AtomicComponent

Class AtomicComponent

[**homework.ch11\_13.p4**](#b2)

java.lang.Object

[homework.ch11\_13.p4.Component](#b27)

**homework.ch11\_13.p4.AtomicComponent**

public class **AtomicComponent**

extends [Component](#b27)

原子组件类，不包含任何子组件

|  |
| --- |
| **Fields inherited from class homework.ch11\_13.p4.**[**Component**](#b27) |
| [id](#b10), [name](#b11), [price](#b12) |

|  |  |
| --- | --- |
| Constructor Summary | **Page** |
| [**AtomicComponent**](#b3)()  缺省构造函数 | 20 |
| [**AtomicComponent**](#b4)(int id, String name, double price)  构造函数 | 20 |

|  |  |  |
| --- | --- | --- |
| Method Summary | | **Page** |
| void | [**add**](#b5)([Component](#b27) component)  添加子组件，对于没有子组件的AtomicComponent如内存条，调用这个方法应该抛出UnsupportedOperati‌onException. | 20 |
| double | [**calcPrice**](#b7)()  计算组件的价格。 | 21 |
| [Iterator](#b52) | [**iterator**](#b8)()  返回组件的迭代器，只需要返回一个NullIterator对象即可。 | 21 |
| void | [**remove**](#b6)([Component](#b27) component)  删除子组件，对于没有子组件的AtomicComponent如内存条，调用这个方法应该抛出UnsupportedOperati‌onException. | 21 |

|  |
| --- |
| **Methods inherited from class homework.ch11\_13.p4.**[**Component**](#b27) |
| [equals](#b25), [getId](#b15), [getName](#b17), [getPrice](#b19), [setId](#b16), [setName](#b18), [setPrice](#b20), [toString](#b26) |

Constructor Detail

AtomicComponent

public **AtomicComponent**()

缺省构造函数

AtomicComponent

public **AtomicComponent**(int id,  
                       String name,  
                       double price)

构造函数

**Parameters:**

id - 组件id

name - 组件名称

price - 组件价格

Method Detail

add

public void **add**([Component](#b27) component)  
         throws UnsupportedOperationException

添加子组件，对于没有子组件的AtomicComponent如内存条，调用这个方法应该抛出UnsupportedOperationException. 相同的子组件不能重复加入

**Overrides:**

[add](#b21) in class [Component](#b27)

**Parameters:**

component - 要添加的子组件

**Throws:**

UnsupportedOperationException - 可能抛出的异常

remove

public void **remove**([Component](#b27) component)  
            throws UnsupportedOperationException

删除子组件，对于没有子组件的AtomicComponent如内存条，调用这个方法应该抛出UnsupportedOperationException.

**Overrides:**

[remove](#b22) in class [Component](#b27)

**Parameters:**

component - 要删除的组件

**Throws:**

UnsupportedOperationException - 可能抛出的异常

calcPrice

public double **calcPrice**()

计算组件的价格。对于复合组件应该计算其子组件的价格之和

**Overrides:**

[calcPrice](#b23) in class [Component](#b27)

**Returns:**

组件的价格

iterator

public [Iterator](#b52) **iterator**()

返回组件的迭代器，只需要返回一个NullIterator对象即可。

**Overrides:**

[iterator](#b24) in class [Component](#b27)

**Returns:**

组件的迭代器

## 2.2 Class Component

Class Component

[**homework.ch11\_13.p4**](#b2)

java.lang.Object

**homework.ch11\_13.p4.Component**

Direct Known Subclasses:

[AtomicComponent](#b9), [CompositeComponent](#b44)

abstract public class **Component**

extends Object

计算机组件的抽象类，任何一个具体的组件键盘、鼠标、主板、主机都是Component。注意主机又由一些更小的Component组成，如内存条、CPU，这种组件为复合组件。

|  |  |  |
| --- | --- | --- |
| Field Summary | | **Page** |
| protected int | [**id**](#b10)  组件的唯一id | 22 |
| protected String | [**name**](#b11)  组件的名字 | 22 |
| protected double | [**price**](#b12)  组件的价格 | 22 |

|  |  |
| --- | --- |
| Constructor Summary | **Page** |
| [**Component**](#b13)()  缺省构造函数 | 22 |
| [**Component**](#b14)(int id, String name, double price)  构造函数 | 23 |

|  |  |  |
| --- | --- | --- |
| Method Summary | | **Page** |
| abstract void | [**add**](#b21)([Component](#b27) component)  添加子组件，对于没有子组件的AtomicComponent如内存条，调用这个方法应该抛出UnsupportedOperati‌onException. | 14 |
| abstract double | [**calcPrice**](#b23)()  计算组件的价格。 | 15 |
| boolean | [**equals**](#b25)(Object obj)  基于组件id判断二个组件对象是否相等 | 15 |
| int | [**getId**](#b15)()  获取组件id | 23 |
| String | [**getName**](#b17)()  获取组件名称 | 24 |
| double | [**getPrice**](#b19)()  获取组件价格 | 34 |
| abstract [Iterator](#b52) | [**iterator**](#b24)()  返回组件的迭代器 | 15 |
| abstract void | [**remove**](#b22)([Component](#b27) component)  删除子组件，对于没有子组件的AtomicComponent如内存条，调用这个方法应该抛出UnsupportedOperati‌onException. | 15 |
| void | [**setId**](#b16)(int id)  设置组件id | 34 |
| void | [**setName**](#b18)(String name)  设置组件名称 | 24 |
| void | [**setPrice**](#b20)(double price)  设置组件价格 | 14 |
| String | [**toString**](#b26)()  返回组件的信息 | 16 |

Field Detail

id

protected int **id**

组件的唯一id

name

protected String **name**

组件的名字

price

protected double **price**

组件的价格

Constructor Detail

Component

public **Component**()

缺省构造函数

Component

public **Component**(int id,  
                 String name,  
                 double price)

构造函数

**Parameters:**

id - 组件id

name - 组件名称

price - 组件价格

Method Detail

getId

public int **getId**()

获取组件id

**Returns:**

组件id

setId

public void **setId**(int id)

设置组件id

**Parameters:**

id - 组件id

getName

public String **getName**()

获取组件名称

**Returns:**

组件名称

setName

public void **setName**(String name)

设置组件名称

**Parameters:**

name - 组件名称

getPrice

public double **getPrice**()

获取组件价格

**Returns:**

组件价格

setPrice

public void **setPrice**(double price)

设置组件价格

**Parameters:**

price - 组件价格

add

public abstract void **add**([Component](#b27) component)  
                  throws UnsupportedOperationException

添加子组件，对于没有子组件的AtomicComponent如内存条，调用这个方法应该抛出UnsupportedOperationException. 相同的子组件不能重复加入

**Parameters:**

component - 要添加的子组件

**Throws:**

UnsupportedOperationException - 可能抛出的异常

remove

public abstract void **remove**([Component](#b27) component)  
                     throws UnsupportedOperationException

删除子组件，对于没有子组件的AtomicComponent如内存条，调用这个方法应该抛出UnsupportedOperationException.

**Parameters:**

component - 要删除的组件

**Throws:**

UnsupportedOperationException - 可能抛出的异常

calcPrice

public abstract double **calcPrice**()

计算组件的价格。对于复合组件应该计算其子组件的价格之和

**Returns:**

组件的价格

iterator

public abstract [Iterator](#b52) **iterator**()

返回组件的迭代器

**Returns:**

组件的迭代器

equals

public boolean **equals**(Object obj)

基于组件id判断二个组件对象是否相等

**Overrides:**

equals in class Object

**Parameters:**

obj - 另外一个对象

**Returns:**

如果二个组件id相等，返回true

toString

public String **toString**()

返回组件的信息

**Overrides:**

toString in class Object

**Returns:**

组件的信息

## 2.3 Class ComponentFactory

Class ComponentFactory

[**homework.ch11\_13.p4**](#b2)

java.lang.Object

**homework.ch11\_13.p4.ComponentFactory**

public class **ComponentFactory**

extends Object

组件的对象工厂, 由对象工厂返回对象

|  |  |
| --- | --- |
| Constructor Summary | **Page** |
| [**ComponentFactory**](#b28)() | 16 |

|  |  |  |
| --- | --- | --- |
| Method Summary | | **Page** |
| static [Component](#b27) | [**create**](#b29)()  创建组件对象, 把创建对象的复杂步骤封装在create方法里 | 16 |

Constructor Detail

ComponentFactory

public **ComponentFactory**()

Method Detail

create

public static [Component](#b27) **create**()

创建组件对象, 把创建对象的复杂步骤封装在create方法里

**Returns:**

创建好的一台计算机

## 2.4 Class ComponentList

Class ComponentList

[**homework.ch11\_13.p4**](#b2)

java.lang.Object

java.util.AbstractCollection<[Component](#b27)>

java.util.AbstractList<[Component](#b27)>

java.util.ArrayList<[Component](#b27)>

**homework.ch11\_13.p4.ComponentList**

All Implemented Interfaces:

Cloneable, Collection<[Component](#b27)>, Iterable<[Component](#b27)>, [Iterator](#b52), List<[Component](#b27)>, RandomAccess, Serializable

public class **ComponentList**

extends ArrayList<[Component](#b27)>

implements [Iterator](#b52)

Component对象的容器类，用于保存复合组件的子组件.从ArrayList派生，实现了自定义Iterator接口. 定义这个类是为了实现组件树的CompositeIterator.由于是从ArrayList派生，因此继承了ArrayList的所有方法.

|  |  |  |
| --- | --- | --- |
| Field Summary | | **Page** |
| private int | [**position**](#b31)  记录自定义迭代器当前迭代的位置 | 17 |

|  |  |
| --- | --- |
| Constructor Summary | **Page** |
| [**ComponentList**](#b32)() | 17 |

|  |  |  |
| --- | --- | --- |
| Method Summary | | **Page** |
| boolean | [**hasNext**](#b33)()  是否还有元素 | 17 |
| [Component](#b27) | [**next**](#b34)()  获取下一个组件 | 41 |

Field Detail

position

private int **position**

记录自定义迭代器当前迭代的位置

Constructor Detail

ComponentList

public **ComponentList**()

Method Detail

hasNext

public boolean **hasNext**()

是否还有元素

**Specified by:**

[hasNext](#b50) in interface [Iterator](#b52)

**Returns:**

如果元素还没有迭代完，返回true;否则返回false

next

public [Component](#b27) **next**()

获取下一个组件

**Specified by:**

[next](#b51) in interface [Iterator](#b52)

**Returns:**

下一个组件

## 2.5 Class CompositeComponent

Class CompositeComponent

[**homework.ch11\_13.p4**](#b2)

java.lang.Object

[homework.ch11\_13.p4.Component](#b27)

**homework.ch11\_13.p4.CompositeComponent**

public class **CompositeComponent**

extends [Component](#b27)

复合组件，包含子组件

|  |  |  |
| --- | --- | --- |
| Field Summary | | **Page** |
| protected [ComponentList](#b35) | [**childs**](#b36)  保存子组件,放在ComponentList里 | 3 |

|  |
| --- |
| **Fields inherited from class homework.ch11\_13.p4.**[**Component**](#b27) |
| [id](#b10), [name](#b11), [price](#b12) |

|  |  |
| --- | --- |
| Constructor Summary | **Page** |
| [**CompositeComponent**](#b37)()  缺省构造函数 | 3 |
| [**CompositeComponent**](#b38)(int id, String name, double price)  构造函数 | 4 |

|  |  |  |
| --- | --- | --- |
| Method Summary | | **Page** |
| void | [**add**](#b39)([Component](#b27) component)  添加子组件，对于没有子组件的AtomicComponent如内存条，调用这个方法应该抛出UnsupportedOperati‌onException. | 4 |
| double | [**calcPrice**](#b41)()  计算组件的价格. | 4 |
| [Iterator](#b52) | [**iterator**](#b42)()  返回组件的迭代器,只需要用childs对象为参数构造一个CompositeIterator对象即可,因为childs对象的类型是‌ComponentList,实现了自定义Iterator接口. | 5 |
| void | [**remove**](#b40)([Component](#b27) component)  删除子组件，对于没有子组件的AtomicComponent如内存条，调用这个方法应该抛出UnsupportedOperati‌onException. | 4 |
| String | [**toString**](#b43)()  返回组件的id，名称、价格 | 5 |

|  |
| --- |
| **Methods inherited from class homework.ch11\_13.p4.**[**Component**](#b27) |
| [equals](#b25), [getId](#b15), [getName](#b17), [getPrice](#b19), [setId](#b16), [setName](#b18), [setPrice](#b20) |

Field Detail

childs

protected [ComponentList](#b35) **childs**

保存子组件,放在ComponentList里

Constructor Detail

CompositeComponent

public **CompositeComponent**()

缺省构造函数

CompositeComponent

public **CompositeComponent**(int id,  
                          String name,  
                          double price)

构造函数

**Parameters:**

id - 组件id

name - 组件名称

price - 组件价格

Method Detail

add

public void **add**([Component](#b27) component)  
         throws UnsupportedOperationException

添加子组件，对于没有子组件的AtomicComponent如内存条，调用这个方法应该抛出UnsupportedOperationException. 相同的子组件不能重复加入

**Overrides:**

[add](#b21) in class [Component](#b27)

**Parameters:**

component - 要添加的子组件

**Throws:**

UnsupportedOperationException - 可能抛出的异常

remove

public void **remove**([Component](#b27) component)  
            throws UnsupportedOperationException

删除子组件，对于没有子组件的AtomicComponent如内存条，调用这个方法应该抛出UnsupportedOperationException.

**Overrides:**

[remove](#b22) in class [Component](#b27)

**Parameters:**

component - 要删除的组件

**Throws:**

UnsupportedOperationException - 可能抛出的异常

calcPrice

public double **calcPrice**()

计算组件的价格. 对于复合组件应该计算其子组件的价格之和.

**Overrides:**

[calcPrice](#b23) in class [Component](#b27)

**Returns:**

组件的价格

iterator

public [Iterator](#b52) **iterator**()

返回组件的迭代器,只需要用childs对象为参数构造一个CompositeIterator对象即可,因为childs对象的类型是ComponentList,实现了自定义Iterator接口.

**Overrides:**

[iterator](#b24) in class [Component](#b27)

**Returns:**

组件的迭代器对象

toString

public String **toString**()

返回组件的id，名称、价格

**Overrides:**

[toString](#b26) in class [Component](#b27)

**Returns:**

组件的id，名称、价格描述字符串

## 2.6 Class CompositeIterator

Class CompositeIterator

[**homework.ch11\_13.p4**](#b2)

java.lang.Object

**homework.ch11\_13.p4.CompositeIterator**

All Implemented Interfaces:

[Iterator](#b52)

public class **CompositeIterator**

extends Object

implements [Iterator](#b52)

复合迭代器, 用于复合组件的迭代

|  |  |  |
| --- | --- | --- |
| Field Summary | | **Page** |
| protected List<[Iterator](#b52)> | [**iterators**](#b45)  保存遍历到的每个节点的迭代器的列表 | 5 |

|  |  |
| --- | --- |
| Constructor Summary | **Page** |
| [**CompositeIterator**](#b46)([Iterator](#b52) iterator)  构造函数 | 48 |

|  |  |  |
| --- | --- | --- |
| Method Summary | | **Page** |
| boolean | [**hasNext**](#b47)()  是否还有元素 | 8 |
| [Component](#b27) | [**next**](#b48)()  获取下一个组件 | 8 |

Field Detail

iterators

protected List<[Iterator](#b52)> **iterators**

保存遍历到的每个节点的迭代器的列表

Constructor Detail

CompositeIterator

public **CompositeIterator**([Iterator](#b52) iterator)

构造函数

**Parameters:**

iterator - 要迭代的组件树的根节点的迭代器

Method Detail

hasNext

public boolean **hasNext**()

是否还有元素

**Specified by:**

[hasNext](#b50) in interface [Iterator](#b52)

**Returns:**

如果元素还没有迭代完，返回true;否则返回false

next

public [Component](#b27) **next**()

获取下一个组件

**Specified by:**

[next](#b51) in interface [Iterator](#b52)

**Returns:**

下一个组件

## 2.7 Interface Iterator

Interface Iterator

[**homework.ch11\_13.p4**](#b2)

All Known Implementing Classes:

[ComponentList](#b35), [CompositeIterator](#b49), [NullIterator](#b56)

public interface **Iterator**

迭代器接口，用于遍历组件树里的每一个组件. 注意这不是java.util.Iterator接口

|  |  |  |
| --- | --- | --- |
| Method Summary | | **Page** |
| boolean | [**hasNext**](#b50)()  是否还有元素 | 9 |
| [Component](#b27) | [**next**](#b51)()  获取下一个组件 | 9 |

Method Detail

hasNext

boolean **hasNext**()

是否还有元素

**Returns:**

如果元素还没有迭代完，返回true;否则返回false

next

[Component](#b27) **next**()

获取下一个组件

**Returns:**

下一个组件

## 2.8 Class NullIterator

Class NullIterator

[**homework.ch11\_13.p4**](#b2)

java.lang.Object

**homework.ch11\_13.p4.NullIterator**

All Implemented Interfaces:

[Iterator](#b52)

public class **NullIterator**

extends Object

implements [Iterator](#b52)

空迭代器，这个迭代器的hasNext()方法永远返回false, next()方法永远返回null. 用于AtomicComponent，因为AtomicComponent没有子组件. 因此AtomicComponent的iterator方法应该返回NullIterator的实例.

|  |  |
| --- | --- |
| Constructor Summary | **Page** |
| [**NullIterator**](#b53)() | 10 |

|  |  |  |
| --- | --- | --- |
| Method Summary | | **Page** |
| boolean | [**hasNext**](#b54)()  是否还有元素 | 10 |
| [Component](#b27) | [**next**](#b55)()  获取下一个组件 | 10 |

Constructor Detail

NullIterator

public **NullIterator**()

Method Detail

hasNext

public boolean **hasNext**()

是否还有元素

**Specified by:**

[hasNext](#b50) in interface [Iterator](#b52)

**Returns:**

如果元素还没有迭代完，返回true;否则返回false

next

public [Component](#b27) **next**()

获取下一个组件

**Specified by:**

[next](#b51) in interface [Iterator](#b52)

**Returns:**

下一个组件

## 2.9 Class Test

Class Test

[**homework.ch11\_13.p4**](#b2)

java.lang.Object

**homework.ch11\_13.p4.Test**

public class **Test**

extends Object

测试类

|  |  |
| --- | --- |
| Constructor Summary | **Page** |
| [**Test**](#b57)() | 11 |

|  |  |  |
| --- | --- | --- |
| Method Summary | | **Page** |
| static void | [**main**](#b58)(String[] args)  测试程序入口 | 11 |

Constructor Detail

Test

public **Test**()

Method Detail

main

public static void **main**(String[] args)

测试程序入口

**Parameters:**

args - 命令行入口