《JAVA 语言程序设计》课程大纲

一、课程名称：JAVA 语言程序设计

二、课程性质：选修、专业课

三、学时与学分：40 学时，2.5 学分

四、课程先导课：计算思维、C 语言或 C++语言、数据结构等。

五、课程简介

“JAVA 语言程序设计”是一门工程性、技术性和实践性都很强的专业课程，对培养学生面向对象的程序设计思维和程序设计能力有很重要的启发和训练作用，并可为学生学习后续专业课程如算法设计、计算机网络打下基础。

课程以面向对象的核心概念为主线，使学生全面、系统地掌握 JAVA 语言及其面向对象程序设计的基本概念、设计思想以及实现方法；具体涵盖了 JAVA 基础语法，包括 JAVA 基本数据类型（值类型）和引用类型、方法重载、数组/多维数组、类、抽象类、接口、泛型、异常处理等；JAVA JDK 里常用类的使用方法，如 String、StringBuffer、输入输入流、JAVA 容器、JAVA 多线程等；面向对象的核心概念，如继承、组合、重载、多态以及这些概念在面向对象程序设计中的应用；面向对象的建模及其在面向对象程序设计中的应用；面向对象的设计模式，如迭代器模式、装饰者模式、组合模式、对象工厂模式及其在面向对象程序设计中的应用。

课程着力加强学生对面向对象的程序设计思想的理解，强化学生面向对象程序设计的能力训练，使得学生能够用面向对象的方式设计较复杂的应用程序。课程主要教学内容与复杂软件工程设计问题的特征相呼应，学生必须熟练掌握 JAVA 语言，同时较好的理解面向对象的概念，才能建立复杂软件的对象模型，并通过业界主流的开发工具设计出足够复杂的应用程序。

六、课程目标

通过相关教学活动，帮助学生理解面向对象的基本概念，掌握面向对象的软件设计方法，并熟练运用面向对象设计语言 JAVA 及其开发工具进行复杂应用程序的设计，提升学生在程序设计过程中分析问题和解决问题的能力。

课程的具体目标包括：

目标 1：深刻理解面向对象的基本概念、面向对象的程序设计思维方式。掌握对象、类、封装、重载、继承、接口、多态、异常、线程、事件、组件等重要概念；理解面向对象的程序设计方法和面向过程的程序设计方法的区别；握面向对象的设计方法、学会建立面向对象的模型。能利用上述知识对计算机计算机软

件系统设计方案进行建模和分析。

目标 2：熟练掌握面向对象程序设计语言 JAVA 的语法，熟练使用 JAVA JDK 所提供的基础类库；掌握基于 JAVA 的面向对象的软件设计、程序编写、调试运行方法。能综合运用面向对象的思想和利用 Java 语言设计实现复杂程序模块，能将 JAVA 语言作为一种思维工具解决实际的工程问题。

目标 3：理解面向对象的设计模式的作用和基本概念，熟悉和掌握各种设计模式的运用场景及实现方法，并能运用这些设计模式来设计复杂软件系统的架构，设计出扩展性好、低耦合、能适应需求变化的软件系统。提升学生的面向对象程序设计能力和对软件系统设计方案进行优化的能力。

目标 4：掌握 JAVA 虚拟机的概念，理解 Java 程序的跨平台运行模式的优势；学习和了解 Java 语言规范的演变和最新进展；了解开源软件的概念、开源软件的许可制度以及 JAVA 开源社区上主流的开源软件的发展现状。培养学生合理、合法地利用开源软件来设计软件系统，解决实际问题的能力。

七、课程目标对毕业要求的支撑关系

支撑的毕业要求二级指标点	对应课程目标
1.2 能应用软硬件工程知识，对计算机复杂工程问题的具体对象进行建模和求解	目标 1
3.2 能为计算机复杂工程问题解决方案设计满足特定需求的软/硬件模块	目标 2
3.3 能为计算机复杂工程问题解决方案进行软、硬件系统和应用系统设计，并能在设计中体现创新意识。	目标 3
8.2 理解诚实公正、诚信守则的工程职业道德和规范，并能在工程实践中自觉遵守	目标 4

八、教学设计及对课程目标的支持

第一章　Java 概述

本章的主要知识点包括面向对象程序设计方法、Java 语言的发展历史和特点、Java 虚拟机、Java 语言规范 JLS、Java 语言开发包 JDK、开源软件的概念。

1.**教学目标**

1)了解面向对象程序设计方法和 Java 语言的发展历史；

2)了解 Java 语言的特点；

3)掌握 Java 语言运行环境 Java 虚拟机的概念，作用；

4)了解 Java 语言规范 JLS，Java 开发包 JDK 及其 API 接口；

5)掌握 Java 运行开发环境 JDK 的安装，运行环境配置，；

6)掌握创建、编译和执行 JAVA 程序的方法，熟悉主流的 Java 开发工具；

7)了解开源软件的概念，开源软件的许可制度，著名的 Java 开源社区；

本章教学支持课程目标 2 和课程目标 4

2.教学重点

1)Java 语言的特征

了解 Java 是纯粹的面向对象程序设计语言，其最大的优势是跨平台，可移植性好；同时和 C/C++进行对比，要求学生能了解面向对象的程序设计方法与面向过程的程序设计方法的区别。

2)Java 语言运行环境 Java 虚拟机的概念和作用

通过介绍 Java 虚拟机的概念，作用，让学生能理解为什么 Java 语言能跨平台运行、同时可移植性好。

3) Java 运行开发环境 JDK 的安装，运行环境配置

要求学生掌握在不同操作系统上安装 Java JDK，同时能不借助任何集成开发环境 IDE 的条件下可以编写源程序，编译源程序并运行。

3.教学难点

1)JAVA 运行环境的配置

理解几个关键环境变量 JAVA_HOME、PATH、CLASSPATH 的作用，配置方法，使得学生能以控制台的方式，利用 Shell 脚本启动 Java 程序；同时掌握在机器上安装不同版本的 JDK 并切换版本的方法。

4.教学环节设计

围绕教学重点和教学难点，综合应用课堂讲授与讨论、作业、课外实践、课外阅读等教学形式。

1)讨论

围绕 Java 虚拟机的概念、Java 运行环境配置等问题展开。

2)作业

围绕利用记事本编写源代码、利用命令行的方式编译、运行程序布置。

3)课外实践

熟悉 Java 主流的集成开发环境 IDE 如 IDEA、Eclipse，掌握在这些 IDE 下编辑、编译、运行 Java 程序的过程，了解 Java 程序的单步跟踪、设置断点、观察变量值等调试程序的方法。

4)课外阅读

网上查阅 Java 语言规范 JLS、了解开源软件许可制度如 GPL、LGPL、Apache License 等；了解著名的 Java 开源社区 Apache。

第二章　基本程序设计

本章的主要知识点包括 Java 语言的标识符，变量，常量；赋值语句和赋值表达式；从控制台和用户界面获取输入；Java 语言的数据类型；Java 编程风格。

1.教学目标

1)掌握 Java 语言的结构；

2)掌握 Java 语言的标识符的命名规则和关键字的含义；

3)掌握 Java 语言的数据类型和运算符；

4)掌握 Java 变量和常量的定义；

5)掌握赋值语句和赋值表达式以及二者的区别

6)掌握 Java 语言从控制台和用户界面获取输入的方法；

7)了解和掌握 Java 的编程风格；

本章教学支持的课程目标为目标 2。

2.教学重点

1)Java 语言的结构

掌握 Java 语言程序的结构特点，理解 Java 语言是纯粹的面向对象设计语言：函数都必须包含在类里，不再有全局变量、全局函数，Java 程序的基本单位不再是函数，而是类。

2)Java 语言数据类型

掌握 Java 语言的数据类型分类：值类型和引用类型；掌握值类型所包括的 8 种内置的基本数据类型的字面量、表示范围大小、变量定义、常量定义。

3)Java 程序获取输入及结果输出方法

掌握利用 JDK 的 Scanner 类从控制台获取输入的方法，利用 JoptionPane 类从用户界面获取输入的方法，将结果输出到控制台的方法。

3.教学难点

1)值类型和引用类型的区别

掌握 Java 语言的 8 种基本数据类型 byte、short、int、long、float、double、char、boolean 为值类型；数组类型、类类型、接口类型为引用类型；掌握值类型变量和引用类型变量在赋值时、作为方法参数传递时行为的区别；理解引用和 C 语言指针的共同特性和区别。

2)Java 基本数据类型的表示范围及数据类型转换

掌握不同数据类型的表示范围大小；理解数据类型的表示范围决定了数据类

型转换的方式；理解和掌握变量赋值时什么时候可以隐式类型转换，什么时候必须强制类型转换。

4.教学环节设计

围绕教学重点和教学难点，综合应用课堂讲授与讨论、作业、课外实践、课外阅读等教学形式。

1)讨论

围绕引用和指针的区别、值类型变量和引用类型变量区别展开。

2)作业

围绕 Java 程序结构、Java 语言标识符命名规则、Java 数据类型、变量常量定义、Java 运算符等内容布置。

3)课外实践

在 IDE 环境里完成编程练习，熟悉从控制台获取输入的方法。

4)课外阅读

网上查询开源的代码风格检查工具 CheckStyle、Google 官方代码规范、Sun 官方代码规范。让学生了解编程风格的重要性和 IT 企业是如何规范代码书写风格的。

第三章　条件分支、循环及方法

本章的主要知识点包括 Java 语言的条件表达式和 if 条件语句，循环语句(while 语句、do-while 语句、for 语句、增强的 for 循环语句)，方法的定义和调用，方法的参数传递(传值和传引用)，方法的重载，方法局部变量的作用域 。

1.**教学目标**

1)掌握条件分支表达式和语句的语法和用途；

2)掌握各种循环语句的语法和用途；

3)掌握嵌套的条件分支和循环的用法；

4)掌握方法定义和调用的语法；

5)理解和掌握方法参数传递的方法；

6)理解和掌握方法重载的概念、作用和语法；

7)理解和掌握作用域的概念和方法局部变量的作用域；

本章教学支持的课程目标为目标 2。

2.教学重点

1)条件分支、循环及其嵌套使用

熟练掌握 Java 语言的流程控制语句的使用方法，并能运用于复杂程序逻辑的实现。

2)Java 语言方法的定义和调用

掌握方法的二种形式化参数类型：值类型参数和引用类型参数。掌握和理解方法调用是通过调用堆栈的方式将实参传递给形参。方法调用时参数传递过程的理解和掌握是后面理解方法局部变量作用域的基础。

3)方法的重载

掌握方法重载的概念，作用和语法。重点强调重载方法必须通过方法参数的差异来区分。

4)方法局部变量的作用域

掌握和理解作用域的概念；掌握方法局部变量作用域取决于变量所定义的程序块：在非嵌套的不同程序块里可以定义同名局部变量，在嵌套程序块里不能定义同名局部变量。

3.教学难点

1)方法的传值调用和传引用调用的区别

理解和掌握方法的值类型参数是传值调用：实参和形参的内容存贮在不同内存单元里，在方法里改变了形参不会影响形参；方法的引用类型参数是引用调用，形参和实参都指向了内存里的同一个对象，因此在方法里通过形参改变了对象的内容，会影响到方法外通过实参读取对象的内容。

2)重载方法的入口地址绑定

理解和掌握编译器根据方法调用的实参类型和参数个数，在编译时就确定了调用哪个重载的方法，即方法的入口地址在编译时就绑定了。

4.教学环节设计

围绕教学重点和教学难点，综合应用课堂讲授与讨论、作业、课外实践等教学形式。

1)讨论

围绕方法传值调用和传引用调用展开。

2)作业

围绕 Java 流程控制结构、方法的定义、方法的重载等内容布置。

3)课外实践

在 IDE 环境里完成布置的编程练习题。

第四章　数学函数、字符和字符串

本章的主要知识点包括 JDK 里 Math、Character、String、StringBuilder 类的使用，Java 语言对字符的编码和对字符类型数据的操作方法，Java 语言里对字符串类型数据的操作方法，Java 的格式化字符串和控制台输出。

1.**教学目标**

1)掌握 Math 类里各种数学计算方法的使用；

2)掌握 Character 类的各种字符操作方法；

3)掌握 String 和 StringBuffer 类对字符串操作的各种方法；

4)掌握格式化字符串和控制台输出的方法；

本章教学支持的课程目标为目标 2。

2.教学重点

1)Math.random 方法的使用

熟练掌握 Math.random 方法；掌握利用 Math.random 产生任意范围[a, b]内的随机整数和任意范围[ch1，ch2]内的随机字符的方法。

2)String 和 StringBuffer 类的使用

熟练使用 String 和 StringBuffer 类对字符串进行各种操作，如字符串的子串查找、字符串的子串替换、判断字符串内容是否相等、字符串的大小写转换、获取字符串长度、改变字符串内容等。

3.教学难点

1)字符串常量池的概念

理解 JVM 维护常量池的作用；理解用字符串字面量和 new 运算符构造字符串对象的区别。

2)String 类和 StringBuffer 类的区别

理解和掌握 String 类型的字符串对象一旦创建内容不可更改，而 StringBuffer 类型的对象其内容是可以动态改变；掌握二种类型的对象相互转换的方法。

3)字符串比较

理解二个字符串对象的引用变量的==比较和利用 equals 方法进行字符串内容比较的区别。

4.教学环节设计

围绕教学重点和教学难点，综合应用课堂讲授与讨论、作业、课外实践、课外阅读等教学形式。

1)讨论

围绕字符串常量池展开，说明内存里常量池和堆的区别。

2)作业

围绕常量池、字符和字符串的操作方法等内容布置。

3)课外实践

在 IDE 环境里完成布置的编程练习题。

4)课外阅读

网上阅读在线 JDK API 文档，了解和熟悉 String、StringBuffer 类的其它方法。

第五章　数组

本章的主要知识点包括数组的声明和定义、数组的复制、数组作为方法的参数和返回值、可变长参数列表和命令行参数、数组的查找和排序、多维数组的声明和定义、Arrays 类的使用。

1.**教学目标**

1)掌握一维数组和多维数组变量的声明和创建；

2)掌握数组的使用，包括数组元素的遍历、数组作为方法的参数和返回值；

3)掌握可变长参数列表和命令行参数的使用；

4)掌握数组的查找和排序算法的实现；

5)掌握数组的复制；

6)掌握利用 JDK Arrays 类实现数组的复制、查找和排序的方法；

本章教学支持的课程目标为目标 2。

2.教学重点

1)一维数组和多维数组变量的声明和创建

掌握一维数组和多维数组变量的声明和创建的语法；掌握多维数组和一维数组之间的关系；掌握数组元素的默认初始化；强调数组类型的变量为引用类型。

2)数组的使用方法

理解和掌握数组的使用场景；强调数组类型作为方法的参数和返回值时，都是引用类型的变量，即方法参数传递是传引用，方法返回值也是引用。

3)数组的复制

掌握数组复制的三种方法：利用 for 循环逐个元素复制、利用数组对象的 clone 方法进行复制、利用 System.arraycopy 方法进行复制；强调数组的复制不能简单对数组引用变量进行赋值。

4)可变长参数列表和命令行参数

掌握方法可变长参数声明的语法和可变长参数的使用(获取可变长参数个数、获取每个参数)；掌握命令行参数的使用(获取命令行参数个数、遍历每个命令行参数)；

3.教学难点

1)不规则多维数组的创建

以不规则二维数组为例，让学生理解二维数组的内存布局；在此基础上掌握

不规则二维数组（即每行的列数不相等）的二步创建方法。

2)数组的浅拷贝复制和深拷贝复制

强调数组类型的变量是引用变量，数组变量的直接赋值是浅拷贝复制；进而让学生理解深拷贝复制和浅拷贝复制的区别，掌握数组深拷贝复制的三种方法。

4.教学环节设计

围绕教学重点和教学难点，综合应用课堂讲授与讨论、作业、课外实践、课外阅读等教学形式。

1)讨论

围绕数组的复制展开，让学生理解浅拷贝复制和深拷贝复制的区别。

2)作业

围绕数组的创建、数组的复制、可变长参数列表、命令行参数、数组的使用等内容布置。

3)课外实践

在 IDE 环境里完成布置的编程练习题。

4)课外阅读

网上阅读在线 JDK API 文档，了解和熟悉 JDK Arrays 类的方法。

第六章　对象和类

本章的主要知识点包括面向对象的建模语言 UML、面向对象的基本概念(类对象、封装)、Java 类的定义、类的构造函数、对象的实例化和 new 运算符、对象的引用和对象访问、对象的实例(或静态)的变量、常量和方法、类和类成员可见性修饰符、类变量的作用域和访问优先级、this 引用、包的概念和使用。

1.**教学目标**

1)掌握面向对象的基本概念，了解面向对象的建模和建模语言 UML；

2)掌握类的定义、对象的实例化和 new 运算符；

3)掌握构造函数的作用、语法规则、调用方式、参数传递；

4)掌握对象引用和对象访问方法；

5)掌握对象的实例(或静态)的变量、常量和方法的定义；

6)理解封装的作用，掌握类和类成员可见性修饰符及访问规则；

7)掌握类变量的作用域和访问优先级，this 引用；

本章教学支持的课程目标为目标 1 和目标 2。

2.教学重点

1)类的定义、对象的实例化和 new 运算符

掌握类定义的语法，包括数据成员和方法的定义。掌握 new 运算符的作用

和类的对象实例化方法。掌握创建对象数组的方法；

2)构造函数

理解构造函数的作用，掌握定义构造函数的三个重要语法规则：函数名必须等于类名、不能有返回值、构造函数可以重载；掌握构造函数的参数传递方法。

3)对象的引用

强调 Class 类型的变量是引用变量；掌握对象引用计数的概念和 Java 虚拟机的对象垃圾回收机制。掌握对象作为方法参数和方法的返回值的使用方法；掌握通过对象引用访问类成员的方法；

4)对象的实例成员和静态成员

掌握类的静态成员的定义方法；理解静态成员的作用和使用场景；掌握类的静态数据成员和实例数据成员的区别；掌握类的静态成员的二种方式：通过类名访问和通过对象访问；

5)类和类成员可见性修饰符

理解面向对象的封装作用；掌握类的二种可见性修饰符 public 和缺省的作用；掌握类成员的可见性修饰符 public、protected、缺省、private 的作用；

6)类变量的作用域和访问优先级及 this 引用

掌握类变量的作用域；强调类变量作用域和方法局部变量作用域的区别；强调当类变量和方法局部变量同名时，优先访问方法局部变量；掌握 this 引用的概念和使用场景：通过 this 引用访问对象的成员，特别是访问和方法局部变量同名的对象成员；掌握在类的构造函数里通过 this 关键字调用其它重载的构造函数的方法；

7)包的定义和引入

理解包的作用和包层次结构与工程目录层次结构之间的关系；掌握包的定义方法，包括包定义 package 语句和包名的命名规则；掌握引入包的作用和方法；

3.教学难点

1)构造函数的调用方式和时机

掌握构造函数的特殊调用方式：只能通过 new 运算符隐式调用，不能像普通方法那样显式调用；理解构造函数的作用是初始化对象的数据成员，以及构造函数的调用时机是创建实例化对象时。

2)对象的生命周期

掌握对象引用计数的概念和 Java 虚拟机的对象自动回收机制；强调 Java 对象的生命周期和 C++对象生命周期的区别，理解 Java 虚拟机的对象自动回收机制的好处。

3)类的保护成员的访问控制

强调子类类体中可以访问从父类继承来的 protected 成员 ；但如果子类和父类不在同一个包里，子类里不能访问另外父类实例（非继承）的 protected 成员。

4)this 引用

理解 this 引用是类的实例方法的一个隐含参数;理解 this 引用所指向的对象;理解在静态函数里不存在 this 引用的原因。

5)包二种引入方式的区别

掌握包的二种引入方式：单类型导入(Single Type Import)和按需类型导入(Type Import on Demand)的区别。

4.教学环节设计

围绕教学重点和教学难点，综合应用课堂讲授与讨论、作业、课外实践、课外阅读等教学形式。

1)讨论

围绕对象引用计数和对象自动回收问题展开，并和 C++的对象回收机制对比，让学生理解 Java 的对象自动回收机制的好处;

2)作业

围绕类的定义、类的构造函数、对象的实例(或静态)的变量、类和类成员可见性修饰符、类变量的作用域和访问优先级、this 引用、包的定义和引入等内容布置。

3)课外实践

在 IDE 环境里完成布置的编程练习题。

4)课外阅读

网上查询资料，深入了解对象建模语言 UML 及其相关建模工具。

第七章　继承和多态

本章的主要知识点包括类的继承、子类和父类的 ISA 关系、子类对象的构造、super 关键字、实例方法覆盖、数据成员和静态方法隐藏、Object 类中的方法、多态性和动态绑定、对象的转型、final 修饰符修饰类方法和类。

1.教学目标

1)掌握类继承的概念、作用(使用场景)、语法;

2)理解子类和父类的 ISA 关系，掌握类继承关系在 UML 里的表示;

3)掌握子类对象的构造过程;

4)掌握 super 关键字的作用和使用方法;

5)掌握实例方法覆盖、数据成员和静态方法隐藏；

6)掌握 Object 类的作用及其重要的方法；

7)掌握多态的概念、作用、原理和使用方法；

8)掌握对象的转型；

9)掌握 final 修饰符修饰类方法和类

本章教学支持的课程目标为目标 1 和目标 2。

2.教学重点

1)类的继承

掌握类继承的语法；让学生理解继承的作用和使用场景；掌握父类和子类的概念、父类和子类之间的 ISA 关系；掌握 UML 模型表示类的继承关系的方法；掌握父类成员在子类里的可访问性。

2)子类对象构造过程

掌握实例初始化块和静态初始化块的作用、执行时机、使用方法；掌握子类对象构造过程；掌握 super 关键字的二种使用方式：通过 super 关键字显式调用父类构造函数和访问从父类继承的成员；掌握在什么情况下子类构造函数会自动调用父类缺省构造函数。

3)实例方法覆盖

掌握实例方法覆盖的概念、作用、语法；掌握数据成员和静态方法隐藏的概念、语法；掌握覆盖与隐藏的区别：覆盖具有多态性，而隐藏没有多态性。

4)Object 类

理解 Object 类的重要地位：所有 Java 类的祖先类；掌握 Object 类的几个重要方法：toString、equals、clone 的功能、方法原型、在 Object 类里的默认实现；理解子类通常需要覆盖这几个方法的原因；掌握子类如何覆盖 Obiect 类的 toString、equals、clone 方法。

5)多态和动态绑定

理解和掌握多态的概念、多态在面向对象程序设计中的作用；掌握多态的实现方法；掌握多态实现的原理和动态绑定的概念。

6)对象的转型

掌握向上转型和向下转型的区别；掌握运算符 instanceof 的用法和作用；掌握向下转型存在的风险。

3.教学难点

1)对象的深拷贝 clone

掌握对象浅拷贝 clone 和深拷贝 clone 的区别；理解浅拷贝 clone 存在的风险；

掌握覆盖 Object 默认实现的 clone 以实现深拷贝 clone 的方法；理解和掌握深度克隆的方法。

2)对象申明类型和运行时类型

掌握对象声明类型和运行时类型的概念；掌握区分对象声明类型和运行时类型的方法；理解和掌握重载方法的绑定由对象的声明类型决定、覆盖方法的绑定是在运行时由运行时类型决定；理解和掌握在编译时，编译器只根据声明类型进行类型检查和方法入口地址绑定。

3)多态特性的作用和实现机制

掌握如何利用多态特性来提高面向对象程序的通用性；掌握动态绑定是如何在运行时重新确定方法的入口地址从而实现多态的。

4.教学环节设计

围绕教学重点和教学难点，综合应用课堂讲授与讨论、作业、课外实践、课外阅读等教学形式。

1)讨论

围绕多态在面向对象程序设计中的作用展开，通过一个 案例引出多态的概念，阐述多态的作用。

2)作业

围绕类的继承、子类对象的构造、super 关键字使用、实例方法覆盖、数据成员和静态方法隐藏、覆盖 Object 类的方法、多态性和动态绑定、对象的转型、等内容布置。

3)课外实践

在 IDE 环境里完成布置的编程练习题。

4)课外阅读

网上阅读在线 JDK API 文档，了解和熟悉 Object 类的方法。

第八章　异常处理和 IO

本章的主要知识点包括异常的概念、异常处理基本机制、异常声明、异常抛出、异常捕获、异常的捕获顺序、自定义异常类，Java IO 类的使用。

1.**教学目标**

1)掌握异常的概念、面向对象的异常处理机制；

2)掌握 Java JDK 定义的异常类继承体系、JDK 定义的常用异常类型；

3)掌握异常的声明、抛出、捕获；

4)掌握异常的捕获次序；

5)掌握自定义异常类的方法；

6)掌握 Java IO(输入输出)类的基本使用方法；

本章教学支持的课程目标为目标 2。

2.教学重点

1)异常的概念和基本的异常机制

掌握异常的基本概念；掌握异常产生的原因；掌握运行时异常系统处理异常的过程。

2) JDK 异常类继承树、预定义的常用异常类型

掌握 JDK 预定义的异常类的继承关系；掌握 Exception 及其派生类和 Error 及其派生类之间的区别；掌握 JDK 预定义的常用异常类型，如：NullPointerException、ArrayIndexOutOfBoundsException、ArithmeticException、IOException 等异常类型发生的场景。掌握 JDK 预定义异常类型的常用方法。

3)异常的声明、抛出、捕获

掌握异常的间接抛出和直接抛出的方法、掌握异常的捕获方法、掌握 try/catch/finally 语句的使用；掌握异常声明的语法和作用。

4)自定义异常类

掌握继承 Exception 自定义异常的方法，以及自定义异常的应用场景。

5)Java IO 类

掌握 Java 读写文本文件的基本方法；掌握和文本文件读写有关的 JDK 类的使用方法，如 File 类、Scanner 类、PrinterWriter 类。

3.教学难点

1)必检异常和非必检异常

掌握必检异常和非必检异常的区别；掌握对于必检异常的二种处理方式：异常声明或捕获异常，以及这二种处理方式的区别和各种优缺点。

2)异常的捕获次序

掌握 catch 子句声明的可捕获异常类型和能捕获的实际异常对象之间的类型相容性；掌握保证每个 catch 子句有机会捕获到异常的方法。

3) try/catch/finally 的执行流程

掌握 try/catch/finally 语句的执行流程；掌握 try/catch/finally 语句的嵌套使用；特别是要理解和掌握在任何情况下，同层的 finally 总是会执行。

4.教学环节设计

围绕教学重点和教学难点，综合应用课堂讲授与讨论、作业、课外实践等教学形式。

1)讨论

围绕异常处理机制和面向过程的程序设计中错误处理机制的区别展开，帮助学生理解异常处理机制。

2)作业

围绕异常的概念、异常处理基本机制、异常声明、异常抛出、异常捕获、异常的捕获顺序、自定义异常类内容布置。

3)课外实践

在 IDE 环境里完成布置的编程练习题。

第九章　抽象类和接口

本章的主要知识点包括抽象方法和抽象类的作用和定义、接口的作用和定义、JDK 常用接口如 Comparable 接口的定义和使用方法、接口和抽象类的区别、。Java 基本值类型的包装类。

1.教学目标

1)掌握抽象方法和抽象类的概念、作用(使用场景)、语法；

2)掌握接口的概念、作用(使用场景)；

3)掌握接口的定义；掌握接口的继承；掌握接口的具体实现；

4)掌握接口、接口和类之间的实现关系在 UML 里的表示；

5)掌握接口和抽象类的区别；

6)掌握 JDK 预定义的常用接口；

7)掌握 Java 基本值类型的包装类的使用；

本章教学支持的课程目标为目标 1 和目标 2。

2.教学重点

1)抽象方法和抽象类

掌握抽象方法的概念、语法和作用；掌握抽象类的概念、语法和作用；掌握抽象方法与抽象类之间的关系；掌握抽象方法和抽象类在 UML 里的表示；掌握抽象方法和抽象类的使用场景；掌握抽象类的性质：不能实例化，但是可以作为变量声明类型、方法参数类型、方法返回类型。

2)接口的定义和接口的实现

掌握接口的定义方法；掌握类和接口之间的实现关系；掌握接口、类和接口之间的实现关系在 UML 中的表示；掌握接口的继承，以及接口继承与类继承的区别；掌握实现接口的方法；掌握接口的使用场景。

3) JDK 预定义的常用接口

掌握 JDK 的 Comparable 接口的定义和作用；掌握 Comparable 接口的使用方法；掌握 JDK 的 Cloneable 接口的定义和作用。

4) Java 基本值类型的包装类

掌握 Java 基本值类型的包装类的继承树；掌握 Byte、Short、Integer、Long、Float、Double 类的作用和使用方法；掌握装箱操作和拆箱操作的概念。

3.教学难点

1)抽象类和接口的区别

通过具体 UML 案例让学生掌握继承和接口的区别：继承描述的是类之间的 ISA 关系，而接口描述的是类的 CANDO 能力；掌握抽象类和接口各自优缺点，以及在使用场景上的区别。

2)基于接口的编程

通过具体案例让掌握基于接口的编程的概念、作用；掌握基于接口的编程的具体设计实践方法。

4.教学环节设计

围绕教学重点和教学难点，综合应用课堂讲授与讨论、作业、课外实践、课外阅读等教学形式。

1)讨论

围绕抽象类和接口的区别展开，通过设计案例说明二者的区别，以及在实际设计中选择二种方案之一时需要考虑的因素。

2)作业

围绕抽象方法和抽象类、接口、JDK 常用接口如 Comparable 接口、Java 基本值类型的包装类等内容布置。

3)课外实践

在 IDE 环境里完成布置的编程练习题。

4)课外阅读

网上阅读在线 JDK API 文档，了解和熟悉 JDK 还定义了哪些重要的接口、它们的接口方法、使用场景。

第十章　面向对象的程序设计及设计模式

本章的主要知识点包括面向对象的软件开发过程、类的关系及其 UML 表示、类的设计原则、设计模式的概念和作用、设计模式的基本原则、具体设计模式迭代器模式、装饰者模式、对象工厂模式。

1.**教学目标**

1)了解面向对象的软件开发过程；

2)掌握类的关系及其 UML 表示方法；

3)了解类的设计原则；

4)掌握设计模式的概念和作用；

5)掌握设计模式的基本原则；

6)掌握迭代器模式、装饰者模式、对象工厂模式；

7)掌握利用类的设计原则和设计模式对软件进行建模、实现、优化的方法

本章教学支持的课程目标为目标 1、目标 2、目标 3。

2.教学重点

1) 类的关系及其 UML 表示

掌握类的继承关系、接口关系、关联关系、依赖关系、聚合关系、组合关系的概念、UML 里的表示；掌握这些类的关系在 Java 程序里的实现方法。

2) 设计模式的概念和作用

掌握设计模式的概念；理解采用设计模式进行面向对象的程序设计的好处、掌握设计模式的基本要素；掌握设计模式的描述方法。

3) 设计模式的基本原则

通过具体案例让学生掌握设计模式的几个重要的基本原则："开-闭"原则、单一职责原则、里氏代换原则、依赖倒置原则、接口隔离原则的概念；这些设计原则在软件设计和开发过程中的作用。

4) 具体设计模式

掌握三种具体的设计模式：对象工厂模式、迭代器模式、装饰者模式；包括每种设计模式的适用场景、每种模式具体实现方案和具体设计案例。

3.教学难点

1)类之间不同关系的区别

掌握关联关系和依赖关系的区别：关联关系通常用数据成员来实现，而依赖关系通常用方法参数来实现；掌握聚合关系和组合关系的区别：聚合关系表示整体与部分之间的关系，但一个对象可以被多个聚集者拥有，即是 Weak HAS-A 的关系，而组合关系里一个从属者只能被一个聚集者所拥有，聚集者负责从属者的创建和销毁，即是 Strong HAS-A 的关系；掌握这些关系在 UML 里表示的区别，以及在 Java 实现代码里的区别。

2)迭代器模式

通过一个具体的设计案例，引入迭代器模式，让学生掌握迭代器模式要解决

的问题和应用场景是什么；掌握迭代器模式应用了哪些设计模式的基本原则；掌握迭代器模式的设计结构及其 UML 表示；掌握迭代器模式里的迭代器接口的作用、迭代器接口的定义；掌握利用迭代器接口去遍历不同数据结构里数据的方法。掌握 Java JDK 里 Java Collection 的迭代器的使用方法。

3)装饰者模式

通过一个具体的设计案例，引入装饰者模式，让学生掌握迭代器模式要解决的问题和应用场景是什么；通过装饰者模式让学生理解和掌握另一个设计模式的重要原则：少用继承，多用组合。掌握装饰者模式的设计结构及其 UML 表示；掌握装饰者模式的关键要素：装饰者和被装饰者有相同的超类型；了解装饰者模式在 Java JDK 里 IO Stream API 里的具体应用，加深对装饰者模式的理解；掌握 Java IO Stream API 的使用方法。

4.教学环节设计

围绕教学重点和教学难点，综合应用课堂讲授与讨论、作业、课外实践、课外阅读等教学形式。

1)讨论

围绕设计模式的基本原则展开，通过设计案例说明每个具体原则的含义和具体设计实践方法。

2)作业

围绕类的关系、掌握设计模式的概念和作用、设计模式的基本原则、迭代器模式、装饰者模式、对象工厂模式等内容布置。

3)课外实践

设计融合了课堂讲授的几种设计模式的编程练习题，在 IDE 环境里完成布置的编程练习题。

4)课外阅读

网上查阅资料，了解和熟悉其它的具体设计模式，如观察者模式、组合模式、策略模式。

第十一章　Java 泛型

本章的主要知识点包括泛型的概念、引入泛型的动机和优点、运行时类型识别 RTTI、Class 类和 Class 对象、Java 的反射、泛型类/接口/方法的定义、通配泛型、泛型擦除和对泛型的限制、Java 泛型集合类的使用。

1.教学目标

1)掌握泛型的概念、动机、优点；

2)掌握 Java 反射的概念、反射 API 的基本使用、反射在 Java 泛型中的作用；

3)掌握定义和实现泛型类/接口/方法；

4)掌握通配泛型的概念、作用和使用方法；

5)掌握泛型擦除和对泛型的限制；

6)掌握 Java 泛型集合类的使用；

本章教学支持的课程目标为目标 2。

2.教学重点

1)泛型的概念、动机、优点

掌握类型参数化的概念；掌握类型参数的定义；掌握类型实参和参数化类型的概念；掌握引入泛型的动机：尽量在编译时根据类型参数检查出类型错误。

2)Java 的反射

掌握运行时类型识别 RTTI 和类型信息的概念和作用；掌握 Class 类和 Class 对象的作用；掌握获取一个类的 Class 对象的三种方式；掌握泛化的 Class 引用的作用和定义方法；掌握利用 Class 对象和反射 API，在运行时动态实例化对象的方法。

3)泛型类/接口/方法的定义

掌握泛型类/接口/方法的定义方法；掌握类型形参在泛型类/接口/方法里面的使用方法；掌握传递类型实参是如何传递给泛型类/接口/方法的；掌握编译器是如何在编译时利用类型实参进行类型检查的；掌握受限的类型参数使用方法；掌握原始类型和泛型的区别与兼容性。

4)通配泛型

掌握数组协变性的概念；理解数组协变性存在的问题，从而掌握通配泛型的作用；掌握通配泛型的三种形式：非受限通配、上界通配符、下界通配符；掌握带上界通配符的泛型类所具备的性质；掌握带下界通配符的泛型类所具备的性质；掌握通配类型和具体类型之间的继承关系；掌握带具体参数类型的泛型类或者带通配符的泛型类之间的继承关系。

5)类型擦除机制

掌握类型擦除的基本思想：类型参数只是在编译时用于类型检查，一旦编译通过，所有的类型参数使命完成而全部被擦除；掌握编译器实现类型擦除的方法：编译泛型类、接口和方法时，会用 Object 代替非受限类型参数 E，而如果一个泛型的参数类型是受限的，编译器会用该受限类型来替换类型参数；掌握当类型擦除后，所有泛型的实例类型都共享擦除后形成的原始类型。

6)Java 泛型集合类

掌握 JDK 定义的容器接口：Collection 接口、List 接口、Map 接口；掌握 JDK 定义的各种容器类：ArrayList、HashMap 等的使用方法；掌握 JDK 定义的各种容器类各自的特性、彼此之间的区别。

3.教学难点

1)通配泛型的 PECS 使用原则

理解和掌握带上界通配符的泛型类所具有的可读但不可写的特性；理解和掌握带下界通配符的泛型类所具有的可写但读取部分失效的特性；通过具体设计案例，进一步理解和掌握 Producer Extends，Consumer Super（PECS）原则；掌握 PECS 原则在实际程序设计时的应用。

2)泛型的使用限制

在理解和掌握泛型类型在运行时已经擦除而不可用的性质；掌握泛型的几个重要的使用限制：不能在 new 运算符后面使用类型参数、不能在静态上下文里使用类型参数、不能在异常类里使用类型参数；掌握在代码上下文里正确使用类型参数的方法。

4.教学环节设计

围绕教学重点和教学难点，综合应用课堂讲授与讨论、作业、课外实践等教学形式。

1)讨论

围绕原始 Comparable 接口与泛型 Comparable 接口展开，通过二种接口的具体例子说明引入泛型的动机和优点。

2)作业

围绕运行时类型识别 RTTI、Class 类和 Class 对象、Java 的反射、泛型类/接口/方法的定义、通配泛型、泛型擦除和对泛型的限制、Java 泛型集合类的使用等内容布置。

3)课外实践

在 IDE 环境里完成布置的编程练习题。

第十二章　Java 多线程

本章的主要知识点包括线程的概念、Runnable 接口和线程类 Thread、线程池、线程的同步和协作、信号量、同步合集。

1.教学目标

1)掌握线程的概念、作用；

2)掌握线程的状态及其状态转换；

3)掌握创建 Java 线程的方法；

4)掌握线程池的概念、作用及利用线程池来管理线程的方法；

5)掌握线程之间的同步、协作方法；

6)掌握 Java JDK 提供的同步集合类的使用；

本章教学支持的课程目标为目标 2。

2.教学重点

1)线程的概念

掌握程序、进程和线程的概念；掌握程序、进程和线程之间的区别；掌握进程和线程之间的关系；掌握多线程程序的作用。

2)线程的创建和运行

掌握创建并启动 Java 线程的二种方式:实现 Runnable 接口或继承 Thread 类；掌握 JDK Runnable 接口的定义和作用；掌握 JDK Thread 类的定义和重要方法如 start、join、sleep 的作用和使用方法；掌握线程调度和线程优先级的基本概念；掌握线程的几种状态：就绪状态、运行状态、休眠状态、等待状态、阻塞状态的定义；掌握导致线程的状态之间转换的原因；掌握编写简单的 Java 多线程程序的方法。

3)线程池

掌握线程任务和线程的区别：线程任务是实现了 Runnable 接口的类的实例，定义了线程的要完成的程序逻辑，而线程是 Thread 类的实例，是线程任务的运行载体；掌握线程池的概念和作用；掌握利用 Executor 接口、ExecutorService 接口、Executors 类来创建线程池，并运行多个线程任务的方法；掌握 Executor 接口和接口方法 execute 的使用；掌握 ExecutorService 接口及其接口方法如 shutdown、isTerminated 的使用；掌握 Executors 类的 newFixedThreadPool、newCachedThreadPool 方法的使用；掌握编写简单的 Java 多线程程序，并利用线程池来执行和管理多线程的方法。

4)线程之间的同步

理解和掌握临界区的概念；理解和掌握如果多个线程访问临界区而没有同步会导致的问题；掌握利用 synchronized 关键字进行线程同步的方法；掌握利用 Lock 锁进行线程同步的方法；掌握利用信号量进行线程同步的方法；掌握 Lock 锁和信号量的区别；掌握编写能正确同步的多线程程序的方法；

5)线程之间的协作

通过具体案例让学生掌握线程协作的应用场景；掌握线程协作和线程同步的区别；掌握条件对象的作用；掌握条件对象的创建；掌握条件对象的

await/signal/signalAll 方法；掌握利用条件对象实现线程之间协作的方法；了解和掌握 Object 类的 wait/signal/signalAll 方法，以及利用 Object 类的 wait/signal/signalAll 方法实现线程之间的协作的方法；掌握编写能相互协作、正确同步的多线程程序的方法。

3.教学难点

1)基于 synchronized 关键字和 Lock 锁线程同步的区别

掌握基于 synchronized 关键字进行同步的二种方式：同步方法和同步语句块；掌握利用 Lock 锁对临界区上锁和解锁的方法；通过不同的具体案例，帮助学生掌握基于 synchronized 关键字和 Lock 锁进行同步的区别：synchronized 关键字是对对象上锁、Lock 锁是对临界区上锁；掌握利用 synchronized 关键字和 Lock 锁二种方式正确实现线程同步的方法。

2)线程同步和协作时状态变化

掌握线程同步时，线程的状态变化：线程等待锁时从运行状态转入等待状态，而锁被释放时等待锁的线程从等待状态进入就绪状态；掌握线程协作时，线程状态的变化：调用条件对象的 await 方法的线程从运行时状态进入等待状态，如果另外的线程调用同一个条件对象的 signal/signalAll 方法，会唤醒处于等待状态的线程使之转入就绪状态；掌握正确判断线程状态的方法；

4.教学环节设计

围绕教学重点和教学难点，综合应用课堂讲授与讨论、作业、课外实践等教学形式。

1)讨论

围绕程序、进程、线程之间的区别展开，引入线程的概念和多线程的作用。

2)作业

围绕 Runnable 接口和线程类 Thread、线程池、线程的同步和协作、信号量等内容布置。

3)课外实践

在 IDE 环境里完成布置的编程练习题。

第十三章　Java FX 基础

本章的主要知识点包括 JAVA FX 程序的基本结构、MVC 设计模式、Java FX 类库的 UML 模型、舞台 Stage 的使用、场景 Scene 的使用、布局面板的使用、UI 组件的使用、属性绑定、节点（Node）的通用属性和方法、编写 Java FX 的应用程序。

1.**教学目标**

1)了解 AWT、 Swing、JavaFX 这几种不同的 UI 编程 API；

2)掌握 Java FX 程序的基本结构和 MVC 设计模式；

3)掌握舞台 Stage、场景 Scene 和节点 Node 之间的关系；

4)掌握 Java FX 的属性绑定；

5)掌握节点（Node）的通用属性和方法；

6)掌握利用布局面板进行 UI 布局；

7)掌握编写 Java FX 应用程序；

本章教学支持的课程目标为目标 2 和目标 3。

2.教学重点

1)Java FX 程序的基本结构

掌握 Java FX 程序的 MVC 结构；掌握 Java FX 程序的主视图对应的 fxml 配置文件、Controller 类、Main 类的作用；掌握在 Main 类的 start 方法里加载 fxml 配置文件以创建视图，以及加载 Controller 对象的方法；掌握舞台 Stage 和场景 Scene 的概念和作用；掌握在 Main 类的 start 方法里创建 Scene 对象，设置 Scene 对象到 Stage 对象并显示出来的方法；掌握在 Java FX 程序的 fxml 配置文件定义视图内容、定义视图对应的控制器、将视图里 UI 组件绑定到 Controller 里的实例变量、将视图里 UI 组件的事件属性绑定到 Controller 里的事件处理器的方法；掌握编写 Java FX 程序并运行的方法。

2)Java FX 类库的 UML 模型

掌握 Stage.Scene、Node、Control 以及 Pane 之间关系的 UML 模型；掌握以 Node 为祖先节点的可视化组件的继承关系；掌握 Parent 类及子类与从 Node 节点直接派生的 Shape、ImageView 类的区别；掌握 Pane 类及其子类的作用：可以作为容器包含任意的 Node 类型对象；掌握 Stage、Scene、Scene 里面的可视化控件之间的树状组合关系。

3)属性绑定

通过具体案例，让学生理解属性绑定的作用和使用场景，并掌握属性绑定的使用方法；掌握 Java FX API 预定义的基本数据类型和 String 所对应的绑定属性；掌握自定义绑定属性的方法；

4)节点（Node）的通用属性和方法

了解 Node 节点通用属性的二种设置方法：同过 Java FX CSS 样式表设置和通过 Java FX API 设置；掌握通过 Java FX CSS 样式表设置 Node 节点属性的语法；掌握设置 Node 节点属性的 Java FX API；通过具体案例，让学生了解 Java

代码与样式分离的好处，同时也掌握完全基于 fxml 配置文件和 CSS 样式表定义 Java FX 应用程序 UI 的方法；了解 JavaFX UI 界面设计工具 SceneBuilder 及其基本使用方法。

5)UI 布局面板

掌握布局面板类Pane及其子类的作用；掌握FlowPane、GridPane、BorderPane、HBox、VBox、StackPane 这几种不同布局面板的布局方式和使用方法；掌握利用面板嵌套，实现复杂 UI 布局的基本方法。

3.教学难点

1)MVC 设计模式

掌握MVC设计模式的概念、应用场景、作用；掌握MVC模式里的模型Model、视图 View、控制器 Controller 三部分的作用和相互关系；掌握 MVC 里所包含的基本设计模式如观察者模式、组合模式和策略模式；掌握 MVC 设计模式是如何被应用到 Java FX 程序里。

2)属性绑定的实现

通过具体案例，让学生了解属性绑定实现所采用的设计模式：观察者模式；掌握属性绑定 Property 接口的作用：对观察者的抽象；掌握 ObservableValue 接口的作用：对被观察者的抽象；掌握 Java FX API 预定义的基本数据类型和 String 所对应的绑定属性同时实现了 Property 接口和 ObservableValue 接口，即这些绑定属性即可以作为观察者，也可以作为被观察者。

4.教学环节设计

围绕教学重点和教学难点，综合应用课堂讲授与讨论、作业、课外实践、课外阅读等教学形式。

1)讨论

围绕 MVC 设计模式展开，引入 Java FX 程序的基本结构。

2)作业

围绕舞台 Stage 的使用、场景 Scene 的使用、布局面板的使用、UI 组件的使用、属性绑定、节点（Node）的通用属性和方法、编写 Java FX 的应用程序等内容布置。

3)课外实践

在 IDE 环境里完成布置的编程练习题。

4)课外阅读

网上查阅资料，了解和熟悉 JavaFX UI 界面设计工具 SceneBuilder。

第十四章　事件驱动编程

本章的主要知识点包括基于事件驱动的应用程序、事件和事件源、事件处理器、事件处理器注册和处理事件、匿名类、匿名内部类、Lambda 表达式、利用内部类、内部匿名类和 Lambda 表达式简化事件处理器的实现。

1.教学目标

1)了解事件驱动编程；

2)掌握事件、事件源、事件处理器、处理器泛型接口 EventHandler；

3)掌握定义事件处理器类、实现事件处理器接口方法以对事件进行处理；

4)掌握事件源和事件处理器对象的绑定；

5)掌握内部类、利用内部类简化事件处理器的实现；

6)掌握内部匿名类、利用内部匿名类简化事件处理器的实现；

7)掌握 Lambda 表达式、利用 Lambda 表达式简化事件处理器的实现；

8)掌握具有人机交互功能的 Java FX 应用程序编写方法；

本章教学支持的课程目标为目标 2 和目标 3。

2.教学重点

1)事件、事件源、事件处理器

掌握事件、事件源、事件处理器的概念和三者之间的关系; 掌握 Java FX 事件类的继承关系；掌握几种不同类型的事件如动作事件 ActionEvent、鼠标事件 MouseEvent、键盘事件 KeyEvent、窗口事件 WindowEvent 的区别，以及事件对象的常用方法如 getSource；掌握事件处理器泛型接口 EventHandler；掌握定义事件处理器类以实现 EventHandler 接口的方法。

2)事件源和事件处理器对象的绑定

掌握 Java FX 事件处理所采用的设计模式：观察者模式；掌握将事件处理器对象(观察者)绑定到事件源对象(被观察者)的方法；通过具体案例，让学生掌握实现不同类型的事件处理器，以处理动作事件 ActionEvent、鼠标事件 MouseEvent、键盘事件 KeyEvent、窗口事件 WindowEvent 的方法；掌握具有人机交互功能的 Java FX 应用程序编写方法。

3)内部类、内部匿名类

掌握内部类的概念、使用场景；掌握内部类的二种不同类型：实例内部类和静态内部类；掌握实例内部类的定义、实例内部类对象的实例化方法；掌握静态内部类的定义、静态内部类对象的实例化方法；掌握内部匿名类的概念、内部匿名类对象的实例化方法；掌握利用内部类和内部匿名类简化事件处理器实现的方法。

4)Lambda 表达式

掌握 Lambda 表达式的概念和使用场景；掌握 Java Lambda 表达式的语法；掌握函数式接口的概念；掌握函数式接口和 Lambda 表达式之间的关系；掌握 Java 编译器处理 Lambda 表达式的方式：将 Lambda 表达式编译为实现了函数式接口的匿名内部类对象；掌握利用 Lambda 表达式简化事件处理器实现的方法。

3.教学难点

1)实例内部类与静态内部类的区别

掌握实例内部类的三个性质：不允许定义静态成员，可以访问包含类的实例成员和静态成员、必须通过包含类的对象来实例化实例内部类对象(即只有当有了包含类的实例，才能实例化实例内部类的对象）；

掌握静态内部类的三个性质：可以定义静态成员，不可以访问包含类的实例成员，可以直接实例化静态内部类对象(不需要包含类实例的存在)。

4.教学环节设计

围绕教学重点和教学难点，综合应用课堂讲授与讨论、作业、课外实践等教学形式。

1)讨论

围绕事件处理器接口展开，讲解 Java FX 对各种不同类型事件的统一处理。

2)作业

围绕事件处理器、事件处理器注册和处理事件、匿名类、匿名内部类、Lambda 表达式、利用内部类、匿名内部类和 Lambda 表达式简化事件处理器的实现等内容布置。

3)课外实践

在 IDE 环境里完成布置的编程练习题。

九、教与学

1.教学方法

主要的教学环节包括课堂授课、课堂讨论、书面作业、编程实践课后阅读等环节。本课程的教学设计特色主要体现在如下四个方面：

1)以案例驱动教学。通常的编程语言教学都是通过非常简单的例子来说明语言的特性，但是这些例子都是简单、零散的，彼此缺乏关联；本课程针对每章知识点，设计若干综合设计案例将各章节的知识点串接起来，使得学生能从总体的角度来理解 Java 语言各种特性的作用以及这些特性之间的关系。

2)以设计模式驱动教学。通常的编程语言教学在教学内容和习题上过于强调语法层面的知识及其训练，而忽略了面向对象的程序设计方法的训练。其教学效果往往是学生能把语言的语法记得烂熟于心，但是完全不知道该怎么用面向对象的程序设计方法来实现一个综合性的应用系统。本课程通过介绍一些经典的面向对象设计模式来进行这方面的训练，同时教学内容里的案例设计也会紧扣这个目标进行设计。

3)强调系统建模和工程化设计。通过面向对象建模语言 UML 的教学，强调软件开发过程中系统建模的重要性。通过对象模型的学习，帮助学生从具体和抽象二个层次理解对象建模与代码实现之间的关系，理解系统设计与代码实现之间的关系，培养学生对复杂软件的系统建模和系统设计能力；另一方面，设计模式的教学能够培养学生的工程化设计能力，如：代码复用能力、降低代码耦合度能力、代码优化设计能力。

4)强调动手实践。该课程的教学与独立设置的课程实验相配合，实验内容与理论课程教学进度同步，通过实验加深对所学理论知识的理解，提升学生应用理论知识解决复杂问题的能力,通过实验也可以检验理论课程的学习效果。

2.学习方法

“Java 语言程序设计”是是一门工程性、技术性和实践性都很强的专业课程，学习过程中，第一需要熟练掌握 Java 语言的语法；第二需要掌握 Java JDK API 里的常用类和接口的用法，能熟练应用 API 解决实际的问题；第三需要掌握面向对象建模方法和设计模式，并能将对象建模和设计模式应用到系统设计和代码实现中；第四独立完成编程作业和课程配套开设的独立实验，通过编程作业和实验，加强对课程理论知识的理解。

十、学时分配

序号	主要内容	学时
1	第一章　Java 概述	2
2	第二章　基本程序设计	2
3	第三章　条件分支、循环及方法	2
4	第四章　数学函数、字符和字符串	2
5	第五章　数组	4
6	第六章　对象和类	4

7	第七章　继承和多态	4
8	第八章　异常处理和 IO	2
9	第九章　抽象类和接口	2
10	第十章　面向对象的程序设计及设计模式	4
11	第十一章 Java 泛型	4
12	第十二章 Java 多线程	4
13	第十三章 Java FX 基础	2
14	第十四章 事件驱动编程	2
总计		40

十一、课程考核与成绩评定

1.课程成绩构成

课程最终成绩由书面作业成绩、编程作业成绩、课程期末考试成绩综合而成，各部分成绩的比例如下：

1)书面作业成绩：10%。书面作业将引导学生复习和巩固课堂讲授的内容，主要考查作业完成率和质量。

2)编程作业成绩：20%。编程作业训练学生用 Java 语言进行面向对象程序设计的能力，培养分析问题和解决问题的能力，通过人工检查和自动测试结合的形式考察学生完成编程题的数量和质量。

2)**期末考试成绩：70%**。主要考核学生对 Java 语言基础语法、Java JDK API、面向对象程序设计方法的掌握程度，是对学生学习情况的全面检验。考试采用书面开卷考试形式，在注重考查基础知识及其覆盖面的同时，通过综合型试题考核学生综合运用所学知识解决复杂工程问题的能力。

课程考核成绩评定如下表所示。

表 1 Java 语言程序设计课程考核与成绩评定

课程目标	考核与评价方式及成绩比例		
	作业	编程实践	期末考试
1	1	3	10
2	4	8	40
3	4	7	20
4	1	2	0

2.考核与评价标准

1)书面作业成绩考核与评价标准

表 2 Java 语言程序设计书面作业考核与成绩评定

评价标准				
优秀	**良好**	**中等**	**及格**	**不及格**
按时提交，概念准确，分析充分，步骤清晰，结果正确。	按时提交，概念准确，分析较充分，步骤清晰，结果大多正确。	按时提交，概念较准确，分析基本充分，步骤基本清晰，结果存在少量错误。	按时提交，概念基本准确，分析不充分，步骤不清晰，结果存在少量错误。	未按时提交交，概念欠准确，结果错误较多。

2) 编程作业成绩考核与评价标准

通过人工检查和自动测试结合的形式考察学生完成编程题的数量和质量。

表 3 Java 语言程序设计编程作业考核与成绩评定

评价标准				
优秀	**良好**	**中等**	**及格**	**不及格**
按时提交，通过测试。 代码结构好，代码风格规范。	按时提交，通过测试。 代码结构较好，代码风格较规范。	按时提交，通过测试。 代码结构不好，代码风格不规范。	按时提交，部分通过测试。 代码结构不好，代码风格不规范。	按时提交，完全没有通过测试。

3)课程考核与成绩评定

根据期末考试的试卷评分标准进行评定。

Java 语言程序设计课程组

2021-05-04 修订