西工大计算机学院高级语言程序设计

实 验 报 告

|  |  |  |  |
| --- | --- | --- | --- |
| 课程名称 | 高级语言程序设计 | 实验课次 | Java第1次 |
| 实验项目 | 实验五 包、接口、类库 | 实验时间 | 2023/05/06 |
| 指导老师评议 | **成绩评定：（满分为100分）**  **指导教师评语：**  **指导教师签名：** | | | |
| 实验内容（要求、算法、步骤和方法） | **请参见** 实验说明书 | | |
| 实  验  结  果 | 具体代码请参见附件中电子版的java源程序。源程序的简要说明如下：  **------------------------------------------------------------**  **Java实验：**  DefiPackage.java —— 同指导内容  说明：无  TestPackage.java —— 同指导内容  说明：无  TestInterface.java —— 同指导内容  说明：目标为思考内容的代码  InterfaceDemo.java —— 上机作业 interfaceDemo 的源文件  说明：无  DemoPackage.java —— 上机作业 DemoPackage 的源文件  说明：无  SubClass.java —— 上机SubClass的源文件  说明：无  ParentClass.java —— 上机作业ParentClass 的源文件  说明：无  **------------------------------------------------------------**  实验结果:(包括运行结果，UML图，以及实验要求中需要书面回答的内容)  **1. 指导内容：DefiPackage 运行结果**      **2. 练习内容**  说明：由于个人编码习惯问题，对相关符号命名做出下列更改：   * Speakable→ISpeakable * Runner→IRunnable   **① 示例部分**    **② 思考部分**  1) 示例代码编译后生成 5 个字节码文件，分别是 Dog.class、Person.class、Runner.class、Speakable.class、TestInterface.class。  2) Bird 的实现及运行结果如下，考虑到指导要求中并未要求 Bird 实现 Runner 接口，故在该思考例题中仅实现 ISpeakable 接口。      3) 抽象类实现对比：  说明：基于抽象类的实现和基于接口类的实现已经全部包含在 TestInterface.java 中。    基于接口的实现通过覆写接口方法实现，基于抽象类的实现通过覆写父类的抽象方法实现。  其中抽象类可以具备自己的非抽象成员变量与方法，反之接口类不行。  在具体实现上，一个只有抽象方法的抽象基类与接口类在行为上类似，但是由于Java 单继承的模式，所以在绝大部分场景中，使用接口的实现要优于继承（只能单继承，但是可以多实现）。  **3. 上机内容：Vehicle**  说明：关于符号命名的更改   * interfaceDemon→InterfaceDemo   实现：    运行结果：    **4. 上机内容：MainPackage**  ① 目录结构：    ② 实现：        ③ 运行结果： | | |
| 实 验 心 得 体 会 | 1. **实验中遇到的问题以及解决方法**   无   1. **实验体会**   无 | | |

西工大计算机学院高级语言程序设计

实 验 报 告

|  |  |  |  |
| --- | --- | --- | --- |
| 课程名称 | 高级语言程序设计 | 实验课次 | Java第1次 |
| 实验项目 | 实验六 异常处理 | 实验时间 | 2023/05/06 |
| 指导老师评议 | **成绩评定：（满分为100分）**  **指导教师评语：**  **指导教师签名：** | | | |
| 实验内容（要求、算法、步骤和方法） | **请参见** 实验说明书 | | |
| 实  验  结  果 | 具体代码请参见附件中电子版的java源程序。源程序的简要说明如下：  **------------------------------------------------------------**  **Java实验：**  CatchDemo.java —— 同指导内容  说明：目标为包含附加内容的完整代码  TestException.java —— 同指导内容  说明：无  ConvException.java —— 上机作业 parseInt 异常捕获源码  说明：无  **------------------------------------------------------------**  实验结果:(包括运行结果，UML图，以及实验要求中需要书面回答的内容)  **1. 指导内容：CatchDemo 运行结果**    若没有捕获异常，则除零异常上抛至最顶层，由 JVM 处理打印 backtrace 并退出。    **2. 练习内容：UserException**  **① 示例部分**    **② 思考部分**  1) throw 关键字引导了一个控制语句，其将抛出异常。  2) throws 关键字用于修饰方法，其表示指定的异常类型不被当前方法处理，而上抛至上层  3) 通过 try-catch 语句实现异常的捕获，在 catch 列表中，可能产生的异常将依次传递，直到匹配到匹配异常类型的 catch 语句。在本例中，非法 regist 调用抛出 UserException 类型的异常，并在匹配的 UserException catch 语句块中被捕获处理。该例程在异常抛出时打断程序进程并跳转至 catch handler 语句块，顺利执行完成后跳出 try-catch 语句块继续执行之后的程序。  4) 若不使用 throws 关键字修饰，则 regist 方法中显式抛出的异常将被标记为未处理的异常，这将导致一个警告并最终导致编译错误。  **3. 上机作业：ConvExceptionTest**  在该实现中，以单元测试的形式测试异常捕获-处理程序的完整性。  为了获得较高的测试覆盖率，我查看了反编译的 Integer.class 并查阅了 parseInt 方法的主体处理部分：    根据其异常抛出的情形，测试内容被划分为以下几个单元：   * 非法符号前缀 * 多符号前缀 * 数值溢出   测试代码如下：    运行结果如下： | | |
| 实 验 心 得 体 会 | 1. **实验中遇到的问题以及解决方法**   无   1. **实验体会**   无 | | |

西工大计算机学院高级语言程序设计

实 验 报 告

|  |  |  |  |
| --- | --- | --- | --- |
| 课程名称 | 高级语言程序设计 | 实验课次 | UML第1次 |
| 实验项目 | 实验一 UML建模基础 | 实验时间 | 2023/05/13 |
| 指导老师评议 | **成绩评定：（满分为100分）**  **指导教师评语：**  **指导教师签名：** | | | |
| 实验内容（要求、算法、步骤和方法） | **请参见** 实验说明书 | | |
| 实  验  结  果 | 具体代码请参见附件中电子版的java源程序。源程序的简要说明如下：  **------------------------------------------------------------**  **UML实验：**  main.pu—— UML01的所有内容  说明：PlantUML源码格式  **------------------------------------------------------------**  实验结果:(包括运行结果，UML图，以及实验要求中需要书面回答的内容)  **说明：观察到本课程对UML的实验要求在规范之内，且相较于UML的绘制，课程的实践更侧重于建模过程与理解。因此，没有选择强大但老旧、沉重，偏向于界面操作，使用lisp作为UML源码，和软件耦合度过高而难以手动维护的Rational ROSE，而是选用了轻量级的开源Java工具 PlantUML（https://github.com/plantuml/plantuml.git），以易读性高的源码形式完成UML部分的所有内容。**  **一、UML实验内容**  **1）UML关系的认识**    注：  ①指导书中的实现关系的方向性存在明显的错误，已纠正。  ②图中同名的 Class\* 仅用作关系的演示，互相独立互不关联  **2）可见性的认识**    注：在PlantUML中，分别使用“○”“□”“◇”“△”作为“公有”“私有”“受保护”“包内公有”的默认渲染样式。  **二、总结与分析**  **1）总结UML在软件工程中的作用以及使用UML建模的必要性**  UML（Unified Modeling Language）是一种用于软件工程的标准建模语言，它提供了一套图形化符号和标记，用于描述和设计软件系统的结构、行为和交互。UML在软件工程中起着重要的作用，并且使用UML建模具有必要性，具体如下：  1. 描述和设计系统：UML提供了一种统一的方法来描述和设计软件系统。通过使用UML建模，开发团队可以共享和交流对系统的理解，准确地捕捉系统的需求和设计方案。  2. 可视化系统：UML通过图形化的表示方式，使得系统的结构、行为和交互更加直观和可理解。这有助于开发团队和利益相关者更好地理解系统的功能和工作原理。  3. 提高沟通效率：UML提供了一种通用的语言，能够帮助开发团队成员之间更加清晰地沟通和交流。使用UML建模，可以避免因为术语和概念的歧义而导致的沟通障碍。  4. 发现问题和风险：通过使用UML建模，可以更早地发现系统设计中的问题和潜在风险。通过分析UML模型，可以检测到潜在的冲突、不一致或者缺陷，从而及时进行修正和改进。  5. 支持系统分析和设计：UML提供了一系列的图形化工具和技术，如用例图、类图、时序图、活动图等，能够支持系统的分析和设计过程。这些工具可以帮助开发团队更好地理解系统需求，设计出高质量的软件架构和设计方案。  6. 促进重用和维护：UML建模可以帮助开发团队识别和抽象出可重用的组件和模块。通过良好的模型设计，可以提高代码的可重用性和可维护性，减少开发时间和成本。  注：以上回答来自ChatGPT | | |
| 实 验 心 得 体 会 | 1. **实验中遇到的问题以及解决方法**   无   1. **实验体会**   无 | | |

西工大计算机学院高级语言程序设计

实 验 报 告

|  |  |  |  |
| --- | --- | --- | --- |
| 课程名称 | 高级语言程序设计 | 实验课次 | UML第1次 |
| 实验项目 | 实验二 UML类图 | 实验时间 | 2023/05/13 |
| 指导老师评议 | **成绩评定：（满分为100分）**  **指导教师评语：**  **指导教师签名：** | | | |
| 实验内容（要求、算法、步骤和方法） | **请参见** 实验说明书 | | |
| 实  验  结  果 | 具体代码请参见附件中电子版的java源程序。源程序的简要说明如下：  **------------------------------------------------------------**  **UML实验：**  main.pu—— UML02的所有内容  说明：PlantUML格式  **------------------------------------------------------------**  实验结果:(包括运行结果，UML图，以及实验要求中需要书面回答的内容)  **一、问题简答**  **1）简述类的定义，以及类的三要素**  类是面向对象编程中的基本概念，用于描述一组具有相似特征和行为的对象的模板或蓝图。类定义了对象的属性（数据）和行为（方法），是创建对象的模板。  类的三要素/基本特征是：封装性、继承性、多态性。  1，封装是一种将数据和操作（方法）封装在类内部的特性。通过封装，类将数据和方法组合在一起，并对外部隐藏了内部的实现细节。只有通过类提供的公共接口（方法）才能访问和操作类的数据。封装提供了数据的保护和安全性，同时也隐藏了实现细节，使得类的使用者只需要关注如何使用类提供的功能，而不需要了解其内部实现。  2. 继承是一种通过从已有类派生出新类的方式来扩展和复用代码的特性。通过继承，一个新的类（子类）可以继承并扩展已有类（父类）的属性和方法。子类可以继承父类的属性和方法，并且可以添加自己的特定属性和方法，或者对父类的方法进行重写。继承提供了代码的重用性和层次性，使得类之间可以建立继承关系，形成类的层次结构。  3. 多态是一种允许使用同一接口来处理不同类型对象的特性。通过多态，一个对象可以表现出多种形态，即同一方法可以根据对象的不同类型产生不同的行为。多态通过继承和方法的重写实现，它提供了代码的灵活性和可扩展性。通过多态，可以编写出更通用和灵活的代码，增加代码的可读性和可维护性。  注：以上回答来自ChatGPT  **2）类的属性和方法的可见性有哪些？UML中如何表示？**  可见性：Public（公有），Protected（受保护），Private（私有），Package（包内公有）  在UML规范中，可见性分别用以下符号标识：  ·+ Public  ·- Private  ·# Protected  ·~ Package  **二、UML实验内容**  **1）A、B、C关系表示**    注：图中关于多重度的表示皆在B、C一侧，由于箭头长度的问题渲染不明显。  **2）学生、老师、课程、教材之间的关系表示**    **3）商品系统的三功能**    注：约定商品的id唯一，故id可以作为商品的唯一索引值。  **4）带打折和查询功能的商品系统**    注：  ① 关于该系统作出以下约定：  1. 未损坏的产品皆为原价，即不打折。  2. 对损坏商品的打折不为十折，即不为原价。  3. 任意的查询参数有效，即存在商品满足查询条件。  ② 在服从上述约定①的条件下，以下逻辑成立：  1. 查询结果总不为空。  2. 可以使用折扣discount同时表示是否损坏与折扣力度两项属性。  ③ 考虑到新增的查询功能涉及的业务过于简单，故选择直接内嵌属性而非额外抽象商品特征类用于描述商品的附加属性。  ④ 由于折扣的存在，添加getPrice()以获取实际属性。  ⑤ 为了方便属性的构造与修改，此处及前一个实验内容的Product成员皆置为公有，在实际应用上可等价为私有成员+配对的公有getter/setter封装，由于加入UML中过于冗杂故此处使用前者描述。  **5）Invoice类**    注：此处的下划线为PlantUML中静态成员/方法的表示形式。  **6）纠正仓库管理系统类图错误**    根据题意，OrderItem与唯一的Product对应，而IncomeOrder是OrderItem的集合，故可纠正如上。  注：依据个人经验，关于多重度的标记**或许**可以是下图形式，即表示可以有多个OrderItem包含同一个Product的订单记录。而若考虑OrderItem是Product的订单汇总，则OrderItem与Product一一对应关联，即指导书所描述的内容。鉴于两种标注都能够理解，且不考虑指导书中额外的可有可无的“错误”的话，此处遵从指导书的内容，使用一一对应关联的多重度标注，即最后的类图图示为上图。    **三、分析与讨论**  **1）讨论类图在建模过程中的重要作用。**  1. 结构可视化：类图以图形化的方式展示系统中的类、对象和它们之间的关系，使系统结构一目了然。  2. 概念抽象：类图帮助开发人员从具体实现中抽象出系统的核心概念和业务逻辑，促进高层次的设计思考。  3. 关系建立：类图表达类之间的关联、继承和依赖关系，帮助开发人员理解和管理类之间的交互和依赖。  4. 设计指导：类图指导软件设计和重构过程，帮助开发人员设计合适的类接口、抽象和继承关系，提高系统的可扩展性和可维护性。  5. 代码生成：类图可以作为生成代码的基础，提供代码框架和结构，加速软件开发过程。  6. 文档和交流：类图作为一种图形化表示方式，用于文档和交流，促进团队协作和沟通。  7. 发现问题：类图可以帮助发现系统中的潜在问题，如代码重复、耦合度高等，促进系统的优化和重构。  注：回答来自ChatGPT  **2）总结在涉及绘制类图中应该注意哪些问题。**  1. 关系的方向性  2. 对于必要的，关键处的多重度的标注 | | |
| 实 验 心 得 体 会 | 1. **实验中遇到的问题以及解决方法**   无   1. **实验体会**   不论是类图还是其它的图，UML的建立不可能脱离具体场景而具有唯一性，即便是对于确定的场景，依旧可能存在多种有效可行的UML表示。故不应该盲目地，脱离实际地，教条地过度苛求UML的正确性，或者应该进一步明确自身的目的，从而筛选出之中最优的UML设计。 | | |