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 | | |