第六章作业

2017221105003 黄文杰

1.Java语言中的包是什么含义？如何使用？

Java语言中，对象是以类的形式体现的。因此，对象重用也就体现在类的重用。可以把类存放在包中，实现类在多个场合反复调用。包实际上是一组类组成的集合，也称之为类库。

包的层次结构与文件系统的文件目录结构是相似的。包名是Java的合法标识符，一般都用小写的字母单词表示。

Java语言提供了一些常用的基本类包，如java.io和Java.lang。包名java.lang中存放的有如System和String类等系统的基础类。编程人员也可以把开发的类存放在自己定义的包中，把相关类集存放在一个包中。

2.package语句和import语句的作用是什么？举例说明。

package语句作为java源文件的第一条语句，指明该文件中定义的类所在的包，若缺省该语句，则指定为无名包。它的格式为：

package packageName1[.packageNmae2[.packageName3...]];

其中，packageName1~packageNameN表示包的目录层次。它对应于文件系统的目录结构。Java编译器把包对应于package语句中指明的这种目录层次关系。例如：

package user.java.sample;

为了能使用Java中提供或用户已创建的类，可以用import语句来引入所需要的类。import语句引用的格式为：

import package1[.package2...].(className|\*);

其中，package1~packageN为包的层次结构，它对应着要访问的类所在文件目录结构，className则指明所要引入的类，如果要从一个包中引入多个类时，则可以用星号（\*）来表示。例如：

import java.awt.\*;

import java.lang.\*;

import sum.tools.debug.\*;

3.Java源程序的组成中至少要有一个什么定义？举例说明。

至少要有一个类或一个接口的定义。

|  |
| --- |
| package sample;  interface One{  int MAX\_INT = 100;  void setData(int a, int b);  int getMax();  }  class Int implements One{  int x,y;  public void setData(int a, int b){  x = a;  y = b;  }  public int getMax(){  return x>y?x:y;  }  }  public class TestInt{  public static void main(String[] args){  Int i = new Int();  i.setData(10, 50);  System.out.println("Return max value = " + i.getMax());  }  } |

4.Java语言中对成员的访问有几种情况？举例说明。

①同一类中

②同一包中

③不同包中子类

④不同包中非子类

举例：

|  |
| --- |
| pack1/AccessControl1.java |
| package pack1;  class Original{  private int nPrivate = 1;  int nDefault = 2;  protected int nProtected = 3;  public int nPublic = 4;  void access(){  System.out.println("\*\*同一类中可以访问的成员：\*\*");  System.out.println("Private member = " + nPrivate);  System.out.println("Default member = " + nDefault);  System.out.println("Protected member = " + nProtected);  System.out.println("Public member = " + nPublic);  }  }  class Derived extends Original{  void access(){  System.out.println("\*\*子类类中可以访问的成员：\*\*");  //System.out.println("Private member = " + nPrivate);  System.out.println("Default member = " + nDefault);  System.out.println("Protected member = " + nProtected);  System.out.println("Public member = " + nPublic);  }  }  class SamePackage{  void access(){  Original o = new Original();  System.out.println("\*\*同一包中可以访问的成员：\*\*");  //System.out.println("Private member = " + o.nPrivate);  System.out.println("Default member = " + o.nDefault);  System.out.println("Protected member = " + o.nProtected);  System.out.println("Public member = " + o.nPublic);  }  }  public class AccessControl1{  public static void main(String[] args){  Original o = new Original();  o.access();  Derived d = new Derived();  d.access();  SamePackage s = new SamePackage();  s.access();  }  } |
| 运行结果：  \*\*同一类中可以访问的成员：\*\*  Private member = 1  Default member = 2  Protected member = 3  Public member = 4  \*\*子类类中可以访问的成员：\*\*  Default member = 2  Protected member = 3  Public member = 4  \*\*同一包中可以访问的成员：\*\*  Default member = 2  Protected member = 3  Public member = 4 |

|  |
| --- |
| pack2/AccessControl2.java |
| package pack2;  import pack1.\*;  class Derived extends Original{  void access(){  System.out.println("\*\*在不同包的子类中，可访问的成员\*\*");  //System.out.println("Private member = " + nPrivate);  //System.out.println("Default member = " + nDefault);  System.out.println("Protected member = " + nProtected);  System.out.println("Public member = " + nPublic);  }  }  class AnotherPackage{  void access(){  Original o = new Original();  System.out.println("\*\*在不同包的非子类中，可访问的成员\*\*");  //System.out.println("Private member = " + o.nPrivate);  //System.out.println("Default member = " + o.nDefault);  //System.out.println("Protected member = " + o.nProtected);  System.out.println("Public member = " + o.nPublic);  }  }  public class AccessControl2{  public static void main(String[] args) {  Derived d = new Derived();  d.access();  AnotherPackage s = new AnotherPackage();  s.access();  }  } |
| 运行结果：  \*\*在不同包的子类中，可访问的成员\*\*  Protected member = 3  Public member = 4  \*\*在不同包的非子类中，可访问的成员\*\*  Public member = 4 |

5.Java语言中有几种对成员访问控制的修饰符，它们是如何使用的？

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 同一类中 | 同一包中 | 不同包中子类 | 不同包中非子类 |
| public | √ | √ | √ | √ |
| protected | √ | √ | √ |  |
| default（缺省） | √ | √ |  |  |
| private | √ |  |  |  |

6.接口与抽象类的主要区别在哪些方面？

①接口中的方法都是由public、static和abstract修饰的抽象方法，而抽象类中则可以有抽象方法，也可以含有非抽象方法；

②接口中的变量都是由public、final和static修饰的常量，而抽象类中即可以有一般的成员变量，也可以有自己声明的变量；

③接口可以用extends关键字实现多重继承，而抽象类继承性是类的单一继承，同时也可以实现接口；

④接口实现的类由关键字implements声明，而抽象类的子类由关键字extends声明；

⑤实现接口的类必须实现接口中的所有方法，而抽象类的子类（非抽象类）只必须实现抽象类中的全部的抽象方法；

⑥接口中的变量（即常量）可以用接口名直接访问，而抽象类的变量则不完全可以用类名直接访问；

⑦接口不是类分级结构的一部分，而没有联系的类可以执行相同的接口。

7.创建一个接口Print。

|  |
| --- |
| S6E7.java |
| 源程序：  interface Print{  void print(String str);  }  class PrintStr implements Print{  public void print(String str){  System.out.print(str);  }  }  public class S6E7{  public static void main(String args[]){  PrintStr p = new PrintStr();  p.print("Interface test");  }  } |
| 运行结果：  Interface test |

8.创建Person接口。

|  |
| --- |
| S6E8.java |
| 源程序：  interface Person{  void setData(String name, char sex, String birthday);  String getData();  }  class Student implements Person{  String name;  char sex;  String birthday;  int sID;  String speciality;  public void setData(String name, char sex, String birthday){}  public void setData(String name, char sex, String birthday, int sID, String speciality){  this.name = name;  this.sex = sex;  this.birthday = birthday;  this.sID = sID;  this.speciality = speciality;  }  public String getData(){  return "name:" + name + "\nsex:" + sex + "\nbirthday:" + birthday +  "\nsID:" + sID + "\nspeciality:" + speciality;  }  }  public class S6E8{  public static void main(String args[]){  Student s = new Student();  s.setData("Tom", 'M', "1999-10-1", 201722, "CS");  System.out.println(s.getData());  }  } |
| 运行结果：  name:Tom  sex:M  birthday:1999-10-1  sID:201722  speciality:CS |