《Java面向对象 - 就业技能》

|  |  |
| --- | --- |
| **章节** | **Ch01 - 认识类与对象** |
| **题目1）** | **什么是面向对象？** |
|  | 面向对象是一种使用封装、继承、多态、抽象等思想进行软件的分析和开发的方法，而java就是一门面向对象编程的语言。 |
| **题目2）** | **什么是类?** |
|  | 现实世界是由很多对象组成的，基于对象抽出了类 |
| **题目3）** | **什么是对象?** |
|  | 对象:真实存在的单个的个体，类:类型/类别，一类个体 |
| **题目4）** | **类的组成部分?** |
|  | 生活中的类：根据对象相似的特征和相似的行为进行归类。例如：桌子、椅子  程序中的类 ：由两个部分组成：1.属性(成员变量)；2.方法(成员函数) |
| **题目5）** | **类和对象的关系?** |
|  | 类是对象的抽象，而对象是类的具体实例。  类是抽象的，不占用内存，而对象是具体的，占用存储空间;类和对象是抽象与具体的关系 |

|  |  |
| --- | --- |
| **章节** | **Ch02 - 类的方法（一）** |
| **题目1）** | **什么是成员变量?** |
|  | 成员变量：  1. 类中，方法外  2. new时存在堆中，对象被回收时消失  3. 成员变量可以设置初始值也可以不设置，如果不设置会有默认值。 |
| **题目2）** | **成员变量又可分实例成员变量和静态成员变量以及各自的特点?** |
|  | 实例成员变量:  1. 属于对象的，存在堆中  2. 有几个对象就有几份实例变量  3. 必须通过对象名.来访问  静态成员变量:  1. 属于类的，存在方法区中  2. 只有一份  3. 常常通过类名.来访问 |
| **题目3）** | **什么是局部变量?** |
|  | 1. 方法中  2. 调方法时存在栈中，方法调用结束时与栈桢一并消失  3. 没有默认值 |
| **题目4）** | **方法的调用方式?** |
|  | 1. 普通类: 实例化一个该类的对象,然后通过对象名.方法名 访问  2. 静态类: 可以通过类名直接访问,而不用实例化对象 |
| **题目5）** | **创建包的关键字package 和 导入包的关键字import的使用方式?** |
|  | package:  1. 作用:避免类的命名冲突  2. 包名命名规范: 建议包名所有字母都小写,且有层次结构  3. 类的完全限定名: 包名.类名  import:  1. 作用:声明类/引入类  2. 同包中的类可以直接访问;  不同包中的类想访问:  2.1.先import声明类再访问类(建议)  2.2.类的全称-----太繁琐(不建议) |

|  |  |
| --- | --- |
| **章节** | **Ch03 - 类的方法（二）** |
| **题目1）** | **什么是方法形参和实参?** |
|  | 形参,就是形式参数，用于定义方法的时候使用的参数，是用来接收调用者传递的参数的。  实参,就是实际参数，用于调用时传递给方法的参数。实参在传递给别的方法之前是要被预先赋值的。 |
| **题目2）** | **调用方法实参注意事项?** |
|  | 实参的个数、数据类型以及次序要和所调用方法声明的参数列表匹配 |
| **题目3）** | **Java中定义的方法形式?以及使用方法的好处?** |
|  | 方法的参数列表可以是有参数的，也可以是没有参数的;  有参方法是指( )中包含一个或多个变量的定义，也称为参数列表  无参方法是指( )中不含参数  使用方法的好处：提高代码的复用性 |
| **题目4）** | **什么是值传递和引用传递？** |
|  | 值传递：方法调用时，实际参数将它的值传递给对应的形式参数，函数接收到的是原始值的副本，此时内存中存在两个相等的基本类型，若方法中对形参执行处理操作，并不会影响实际参数的值。  引用传递：方法调用时，实际参数的引用（是指地址，而不是参数的值）被传递给方法中相应的形式参数，函数接收到的是原始值的内存地址，在方法中，形参与实参的内容相同，方法中对形参的处理会影响实参的值。  注意: 这里要特殊考虑String，以及Integer、Double等几个基本类型包装类，它们都是immutable类型，因为没有提供自身修改的函数，每次操作都是新生成一个对象，所以要特殊对待，可以认为是和基本数据类型相似，传值操作。 |
| **题目5）** | **this的作用？** |
|  | 1. 代表本类当前对象的引用  2. 既可以调用本类成员变量、成员方法，也可以调用本类的构造方法。  3. 用于区别局部变量和成员变量 |

|  |  |
| --- | --- |
| **章节** | **Ch04 - 继承与封装** |
| **题目1）** | **什么是封装？封装的原则？好处？** |
|  | 封装就是隐藏对象的属性和具体实现细节，仅对外提供公共的访问方式。  原则：1. 将不需要对外暴露的信息隐藏；  2. 对外提供公共的访问方式。  好处：将变化隔离；提高了安全性；提高了代码代码重用性，便于使用。 |
| **题目2）** | **封装就是私有，对吗？为什么？get/set访问方式必须成对出现吗？** |
|  | 不对，private(私有)仅仅是封装的一种体现形式。我们常用的类，方法，函数也是封装。只要是对外不可见，就能达到封装的效果，比如：包与包之间的访问。  get/set访问方式不是必须成对出现的，具体看需求，需要设置值就提供set方法，如果需要访问值，就提供get方法 |
| **题目3）** | **构造方法的特点？** |
|  | 1. 方法名和类名完全一致  2. 没有返回值类型，连viod都没有。  3. 没有明确的返回值，但是可以有return关键字; |
| **题目4）** | **构造方法，set方法都可以给成员变量赋值，这两种赋值方式有什么区别？** |
|  | 构造方法主要作用是用来给对象初始化，赋值只是它的捎带工作，也可以不用赋值。  Set方法只能用来赋值，在原有对象的基础上赋值，可以用来修改值。  构造方法重新赋值，相对于重新创建对象。 |
| **题目5）** | **静态代码块和构造代码块的区别？** |
|  | 1. 静态代码块随着类的加载而加载，一般是用来加载驱动的。只在类加载的时候执行一次，优先于构造方法执行  2. 构造代码块里边放的是所有构造方法的共性内容，为了简化书写，调高效率。每创建一次对象，就执行一次，它是优先于构造方法执行的。 |

|  |  |
| --- | --- |
| **章节** | **Ch05 - 接口&抽象类** |
| **题目1）** | **类变量（静态变量）和实例变量（对象变量，成员变量）的区别？** |
|  | 1. 所属不同：类变量属于类，是对象的共性内容；实例变量属于对象，是对象的特性内容。  2. 在内存中位置不同：类变量存在方法区的静态区；实例变量存在堆内存中。  3. 生命周期不同：类变量随着类的加载而存在，随着类的消失而消失；实例变量随着对象的存在而存在，随着对象的消失而消失。  4. 调用方式不同：类变量既能被类名点的形式调用，也能通过对象点的形式调用；而实例变量只能通过对象点的形式调用。 |
| **题目2）** | **什么是继承？** |
|  | 当多个类中有很多共性的内容时，我们可以把这些共性内容抽取出来封装成一个类，让这些类与这个封装的类产生关系。这种关系就是继承。 |
| **题目3）** | **继承的的特点和好处，弊端？** |
|  | 特点：可以从以下两方面来讲：  类与类之间的继承：只能单继承不能多继承，但是可以多层继承。  接口与接口之间的继承：既可以单继承也可以多继承。  好处：  1. 提高了代码的复用性、维护性、可扩展性。  2. 让类与类产生了关系，是多态的前提。  弊端：增强了类与类的耦合性。 |
| **题目4）** | **this和super 的区别？** |
|  | this代表本类当前对象的引用，谁调用我，我就代表谁。  super代表当前对象父类的内存空间标识。(可以理解为父类的引用，通过super可以访问父类的成员) |
| **题目5）** | **super( )和this( )在构造方法能同时使用吗？** |
|  | 不能，super( )调用的是父类的空参构造，this( )调用的是本类的空参构造，因为它们都要求放构造方法的第一行，所以不能同时使用。 |

|  |  |
| --- | --- |
| **章节** | **Ch06 - 多态与方法重写** |
| **题目1）** | **Java的访问修饰符是什么?** |
|  | 访问权限修饰符是表明类成员的访问权限类型的关键字。使用这些关键字来限定程序的方法或者变量的访问权限。  它们包含：  public: 所有类都可以访问  protected: 同一个包内以及所有子类都可以访问  默认: 归属类及相同包下的子类可以访问  private: 只有本类才能访问 |
| **题目2）** | **什么是多态?使用多态的好处以及弊端?** |
|  | 多态是同一个行为具有多个不同表现形式或形态的能力。  多态的好处:  1. 提高了代码的维护性(继承保证)  2. 提高了代码的扩展性(由多态保证)  多态的弊端：不能使用子类的特有功能。 |
| **题目3）** | **Java中实现多态的机制是什么？** |
|  | 方法的重写和重载是Java多态性的不同表现。重写是父类与子类之间多态性的一种表现，重载是一个类中多态性的一种表现。 |
| **题目4）** | **方法重载和方法重写区别?** |
|  | 方法重载：同一个类中,方法名相同,参数的类型、顺序和个数不同,与返回值类型和方法访问修饰符无关  方法重写：不同类中,发生在继承类中,方法名称、参数类型、返回值类型全部相同,被重写的方法不能拥有比父类更严格的权限 |
| **题目5）** | **Java实现多态的必要条件?** |
|  | 继承：在多态中必须存在有继承关系的子类和父类。  重写：子类对父类中某些方法进行重新定义，在调用这些方法时就会调用子类的方法。  向上转型：在多态中需要将子类的引用赋给父类对象，只有这样该引用才能够具备技能调用父类的方法和子类的方法 |

|  |  |
| --- | --- |
| **章节** | **Ch07 - 异常** |
| **题目1）** | **try-catch-finally异常捕获语句的执行流程?** |
|  | try中是可能发生异常的程序段；  catch中依次编写对应的异常处理器方法，当抛出异常后，由运行时系统在栈中从当前位置开始依次回查方法，直到找到合适的异常处理方法，如果未找到，则执行finally或直接结束程序运行。  finally ：无论是否捕获或处理异常，finally块里的语句都会被执行。  注意：当在try块或catch块中遇到return语句时，finally语句块将在方法返回之前被执行;  finally块不会被执行情况: 在前面的代码中用了System.exit(0)退出程序。 |
| **题目2）** | **throw和throws关键字的区别？** |
|  | throw用来抛出一个异常，在方法体内。语法格式为：throw 异常对象。  throws用来声明方法可能会抛出什么异常，在方法名后;  语法格式为： throws 异常类型1，异常类型2...异常类型n。 |
| **题目3）** | **异常的两种类型，Error和Exception的区别?** |
|  | error 表示恢复不是不可能但很困难的情况下的一种严重问题。比如说内存溢出。不可能指望程序能处理这样的情况。  exception 表示一种设计或实现问题。也就是说，它表示如果程序运行正常，从不会发生的情况。 |
| **题目4）** | **运行时异常与一般异常有何异同？** |
|  | 异常表示程序运行过程中可能出现的非正常状态，运行时异常表示虚拟机的通常操作中可能遇到的异常，是一种常见运行错误。java编译器要求方法必须声明抛出可能发生的非运行时异常，但是并不要求必须声明抛出未被捕获的运行时异常。 |
| **题目5）** | **final, finally, finalize的区别?** |
|  | final 用于声明属性，方法和类，分别表示属性不可变，方法不可覆盖，类不可继承。  finally是异常处理语句结构的一部分，表示总是执行。  finalize是Object类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法，可以覆盖此方法提供垃圾收集时的其他资源回收 |

|  |  |
| --- | --- |
| **章节** | **Ch08 - QuickHit** |
| **题目1）** | **抽象类的特点？** |
|  | 1. 抽象方法和抽象类都必须被abstract关键字修饰。  2. 抽象方法一定在抽象类中。  3. 抽象类不可以用new创建和实例化对象。因为抽象类本身就是不完整的。  4. 抽象类中的抽象方法要被使用，必须由子类复写所有的抽象方法后，建立子类对象调用。 |
| **题目2）** | **abstract 关键字，和哪些关键字不能共存？** |
|  | final：被final修饰的类不能有子类（不能被继承）。而被abstract修饰的类一定是一个父类（一定要被继承）  private: 抽象类中私有的抽象方法，不被子类所知，就无法被复写；而抽象方法出现的就是需要被复写。  static：如果static可以修饰抽象方法，那么连对象都省了，直接类名调用就可以了。可是抽象方法运行没意义。 |
| **题目3）** | **接口的特点?** |
|  | 1. 接口不可实例化，可结合多态进行使用(接口 对象=new 对象())  2. 接口里的成员属性全部是以 public(公开)、static(静态)、final(最终) 修饰符修饰  3. 接口里的成员方法全部是以 public(公开)、abstract(抽象) 修饰符修饰  4. 接口里不能包含普通方法  5. 子类继承接口必须实现接口里的所有成员方法，除非子类也是抽象类 |
| **题目4）** | **面向接口编程的好处？** |
|  | 1. 更加抽象,更加面向对象  2. 提高编程的灵活性  3. 实现高内聚、低耦合，提高可维护性，降低系统维护成本。 |
| **题目5）** | **抽象类和接口的区别？** |
|  | 抽象类：是一个不能被实例化的类，因为它其中存在抽象方法，但它的其它行为和一个普通类没什么不同。  接口：是java为了弥补不能多继承提供的概念，接口之间支持多继承，接口中只允许存在公有静态常量或公有的抽象方法，一个类可实现多个接口，从而扩展不同的功能。 |