1. 回顾Java
2. java基础

数据类型(八大基本类型、引用类型:数组、接口、类)、变量、常量、运算符、数组(一维数组、了解二维数组)、字串符处理

控制结构:选择结构(if语句|swich)、循环结构(while循环、do.while循环、for循环、foreach循环)、二重循环、跳转语句(break、continue)

1. java面向对象(高级): 面向对象是编程思想

2.1 类、属性、方法、对象、构造方法、访问修饰符

2.2 类和对象的关系:类是对象抽象、而对象是类的实例

2.3 方法架构程序:重点理解一个方法一个功能(业务)

2.4(高级):面向对象三大特征:

封装（指类私有成员的封装）、继承、多态(同一个引用等于不同子类对象)

2.5接口:定义接口、实现接口、使用接口(多态:接口等于实现接口类的对象)

总结:

类:普通类、抽象类、密封类

方法:普通方法、重载方法、构造方法、抽象方法、重写方法、静态方法

1. java高级特性
   1. java异常机制（try,catch,finally,throw,throws）
   2. java集合框架(面试问的比较多)
   3. java io操作
   4. 多线程(面试问到): 线程同步(锁)问题
   5. 网络编程(扩展):socke编程
   6. Xml与xml解析

总结:存储数据的角度

变量:存一个学生姓名

数组:存多个学生姓名

类(结构):存一个学生信息(包括姓名，年龄…)

对象数组(类数组):存多个学生信息

集合:存多个学生信息(无长度限制)

(永久)文件:存学生相关信息-<--操作--io流

Xml文件:有规范的存储学生

数据库(后续)

1. 面向加强回顾(模拟收费站)
2. 需求:模拟实现高速收费站的收费功能【根据不同的车型实现收费】

假设车型只有两种:

轿车[通过品牌收费，假设只有两种车型:宝马收20 ，非宝马15元]

客车[能过座位数收费，每座2元 ]

实现思路:

定义收费站类: 实现收费的方法

定义车父类:实现计算费用的方法

定义轿车类:重写计算费用的方法

私有属性为车的品牌

定义客车类:重写计算费用的方法

私有属性为客人的数量

实现代码:

1. 定义车类

车的父类

|  |
| --- |
| *//车的父类* **public abstract class** Car {  *//计费的方法* **public abstract int** cal(); } |

车的子类

SmallCar

|  |
| --- |
| *//小轿车* **public class** SmallCar **extends** Car{   **private** String **brand**;   **public** SmallCar() {  }  **public** SmallCar(String brand) {  **this**.**brand** = brand;  }   **public** String getBrand() {  **return brand**;  }   **public void** setBrand(String brand) {  **this**.**brand** = brand;  }   @Override  **public int** cal() {  **if**(**this**.**brand**.equals(**"宝马"**))  **return** 20;  **else  return** 15;  } } |

Bus

|  |
| --- |
| *//客车类型* **public class** Bus **extends** Car {   **private int seats**; *//座位数* **public** Bus() {  }  **public** Bus(**int** seats) {  **this**.**seats** = seats;  }   **public int** getSeats() {  **return seats**;  }   **public void** setSeats(**int** seats) {  **this**.**seats** = seats;  }   @Override  **public int** cal() {  **return this**.**seats**\*2;  } } |

1. 定义收费站类

|  |
| --- |
| *//收费站* **public class** Sation {  *//收费方法* **public void** getMoney(Car car){ *//父类作为参数* **int** money=car.cal();  System.***out***.println(**"请支付"**+money+**"元"**);  } } |

1. 定义测试类

|  |
| --- |
| **public class** Test {  **public static void** main(String[] args) {  *//多态  /\* Car c=new Bus(24);//new SmallCar("大众");  int money=c.cal();  System.out.println("请支付"+money+"元");\*/   //测试  //创建车对象* List<Car> list=**new** ArrayList<>();  SmallCar smallCar=**new** SmallCar(**"宝马"**);  list.add(smallCar);  SmallCar smallCar2=**new** SmallCar(**"大众"**);  list.add(smallCar2);  Bus bus=**new** Bus(12);  list.add(bus);  BiglCar biglCar=**new** BiglCar(10);  list.add(biglCar);   *//1.创建收费站对象* Sation sation=**new** Sation();  **for** (Car c:list) {  sation.getMoney(c);  }  } } |

1. 服务器支持多客户端(使用多线程)
2. 服务器代码采用多线程接收多个客户端

|  |
| --- |
| 1. *//服务器端* **public class** Server {  **public static void** main(String[] args) **throws** InterruptedException {  **try** {  *//1.创建ServerSocket对象* ServerSocket serverSocket=**new** ServerSocket(6699); *//指定端口  //2.等待客户端来连* System.***out***.println(**"客户端快来连我。。。我等你"**);  **while**(**true**) {  Socket socket = serverSocket.accept(); *//启动多线程  //开启线程* MyThread myThread=**new** MyThread(socket);  myThread.start();  }  *// serverSocket.close();* } **catch** (IOException e) {  e.printStackTrace();  }  } } |

1. 编写多线程收发消息的线程类

|  |
| --- |
| **public class** MyThread **extends** Thread {  **private** Socket **socket**;  **public** MyThread(){}  **public** MyThread(Socket socket){  **this**.**socket**=socket;  }  @Override  **public void** run(){  **try** {  *//负责接收消息  //3.获取流对象进行收发信息  //3.1获取流对象* InputStream inputStream=**socket**.getInputStream();  OutputStream os=**socket**.getOutputStream();  *//3.2接收消息* **int** len=0; *//确定流中有没有数据* **while**(len==0){ *//死循环* len=inputStream.available();  }  **byte** [] bs=**new byte**[inputStream.available()];  inputStream.read(bs); *//读* String str=**new** String(bs); *//将字节数组转化为字符串* System.***out***.println(**"服务器收到客户端的消息:"**+str);   *//3.3发送消息* String msg=**"恭喜你！登入成功"**;  os.write(msg.getBytes()); *//发消息 写个图片数据    //关闭资源* os.flush();  os.close();  inputStream.close();  **socket**.close();  }**catch** (Exception e){  e.printStackTrace();  }  } } |

1. Xml文件更新

加载文件🡪修改docment🡪保存文件

添加节点:

|  |
| --- |
| **public static void** main(String[] args) {  *//使用DOM4J解析xml的步骤  //1.导入dom4j的jar包  //2.创建SaxReder对象加载xml文件  //File file=new File("E:\\IdeaProjects\\Java\_ch01\\src\\stulist.xml"); //绝对路径* **try** {  *//获取项目目录* String path=System.*getProperty*(**"user.dir"**);  File file=**new** File(path+**"\\src\\stulist.xml"**); *//相对路径* SAXReader reader=**new** SAXReader();  Document doc=reader.read(file);  *//3.获取根元素操作xml* Element root=doc.getRootElement();   *//添加节点* Element student=root.addElement(**"student"**);  student.addElement(**"name"**).addText(**"小哥"**);  student.addElement(**"age"**).addText(**"34"**);  student.addElement(**"sex"**).addText(**"不男不女"**);  student.addAttribute(**"xh"**,**"301"**);   *//保存文件* **try** {  FileWriter fw=**new** FileWriter(file);  OutputFormat outputFormat=OutputFormat.*createPrettyPrint*();  XMLWriter xw=**new** XMLWriter(fw,outputFormat);  xw.write(doc);  xw.close();  fw.close();  System.***out***.println(**"添加成功"**);  } **catch** (IOException e) {  e.printStackTrace();  }       } **catch** (DocumentException e) {  e.printStackTrace();  } } |

删除节点:

|  |
| --- |
| **public static void** main(String[] args) {  *//使用DOM4J解析xml的步骤  //1.导入dom4j的jar包  //2.创建SaxReder对象加载xml文件  //File file=new File("E:\\IdeaProjects\\Java\_ch01\\src\\stulist.xml"); //绝对路径* **try** {  *//获取项目目录* String path=System.*getProperty*(**"user.dir"**);  File file=**new** File(path+**"\\src\\stulist.xml"**); *//相对路径* SAXReader reader=**new** SAXReader();  Document doc=reader.read(file);  *//3.获取根元素操作xml* Element root=doc.getRootElement();   *//删除节点* List<Element> list=root.elements();  **for** (Element e:list) {  *//取学号* String xh= e.attributeValue(**"xh"**);  **if**(xh.equals(**"301"**)){  e.getParent().remove(e); *//父级移除子集* }   }    *//保存文件* **try** {  FileWriter fw=**new** FileWriter(file);  OutputFormat outputFormat=OutputFormat.*createPrettyPrint*();  XMLWriter xw=**new** XMLWriter(fw,outputFormat);  xw.write(doc);  xw.close();  fw.close();  System.***out***.println(**"删除成功"**);  } **catch** (IOException e) {  e.printStackTrace();  }       } **catch** (DocumentException e) {  e.printStackTrace();  } } |