一． 简答题（共 5 题，共 20 分，每个小题 4 分）

（1.1）一个".java"源文件中是否可以包含多个类？是否可以包含多个 public 类？若包含一个 public 的类时，此时 java 源代码文件名有什么限定？为什么？

解答：一个“.java”源文件里面可以包含多个类，但是只允许有一个 public 类，并且类名必须和文件名一致。每个编译单元只能有一个 public 类。这么做的意思是，每个编译单元只能有一个公开的接口，而这个接口就由其 public 类来表示。你可以根据需要，往这个文件里面添加任意多个提供辅助功能的 package 权限的类。但是如果这个编译单元里面有两个或两个以上的 public 类的话，程序就不知道从哪里导入了，编译器就会报错。

（1.2）switch 语句中表达式的数据类型可以是哪些？若一个 case 语句分支之后的语句块中 break 语句没有写会出现什么情况？default 分支语句是否必需？若有 default 分支语句，其什么情况下会被执行？

解答：在 java 中 switch 后的表达式的类型只能为以下几种：byte、short、char、int（在 Java1.6 中是这样），在 java1.7 后支持了对 string 的判断。如果不在 switch 结构的 case 中使用 break 语句。程序就会接着执行下面的语句。switch 的 default 子句可以省略不用，它不是必须的。default 用于处理所有 switch 结构的非法操作。当表达式的值与任何一个 case 都不匹配时，则执行 default 语句。

（1.3）如何分别获取一个数组 arr 的长度、一个字符串 s 的长度和一个集合 c 的大小？数组和后两者的区别是什么？

解答：其分别为数组 arr 的长度：arr.length; 字符串 s 的长度：s.length();集合 c 的大小：c.size(); 数组没有单独定义成一个类，数组使用一个字段（length）来表示长度，而字符串 String 和集合 Collections 都定义为类，因此使用方法来获取长度。

（1.4）一个 Java 文件中入口 main()方法的声明格式是什么？一个 java 文件中是否可以有多个 main()方法用作程序执行的入口？main()方法是否可以重载？main()方法是否可以重写？

解答：主类 main()方法的声明为 public static void main(String[] args)；可以有多个 main()方法； 同其他静态方法一样，main()方法可以重载； 同其他静态方法一样，

main()方法不可以重写。

（1.5）什么是泛型？泛型的作用？泛型的使用方式有哪几种？什么是泛型擦除机制？为什么要擦除?

解答：Java 泛型（Generics）是 JDK 5 中引入的一个新特性。使用泛型参数，可以增强代码的可读性以及稳定性。编译器可以对泛型参数进行检测，并且通过泛型参数可以指定传入的对象类型。比如 ArrayList<Persion> persons = new ArrayList<String>() 这行代码就指明了该 ArrayList 对象只能传入 Persion 对象，如果传入其他类型的对象就会报错。可以用于构建泛型集合。原生 List 返回类型是 Object ，需要手动转换类型才能使用，使用泛型后编译器自动转换。泛型一般有三种使用方式：泛型类、泛型接口、泛型方法。Java 的泛型是伪泛型，这是因为 Java 在编译期间，所有的泛型信息都会被擦掉，这也就是通常所说类型擦除。编译器会在编译期间动态将泛型 T 擦除为 Object 或将 T extends xxx 擦除为其限定类型 xxx。泛型本质上是编译器的行为，为了保证引入泛型机制但不创建新的类型，减少虚拟机的运行开销，所以通过擦除将泛型类转化为一般类。

二． 程序阅读与分析（请阅读与分析下面程序是否正确？正确请写出运行结果；错误请修改程序代码。）（共 4 小题，共 20 分，每题 5 分）

注意答题格式为：

(1) 程序代码正确,输出出结果为：********.

(2) 程序代码第*行和第*行有错误，错误原因为***，应该修改为***，修改后输出结果为:****。

（2.1） **程序代码如下：**

```
00      public class Test {
01          public int aMethod() {
02              static int i = 5;
03              System.out.println(i++);
04              return i;
05          }
06          public static void main (String args[]) {
07              Test test = new Test();
08              test.aMethod();
```

```
09                  int j = test.aMethod();
10                  System.out.println(j);
11              }
12          }
```

解答：第 02 行 static 用来修饰全局变量，不能修饰局部变量
去掉 static 限定符
输出结果为：
5
5
6

（2.2） **程序代码如下：**

```
00          Class Test{
01              public static void main(String[] args){
02                  String s1 = new String("ab");
03                  String s2 = new String("ab");
04                  String ss1 = "ab";
05                  String ss2 = "ab";
06                  System. out.println(ss1 == ss2);
07                  System.out.println(s1 == s2);
08                  System.out.println(s1.equals(s2)) ;
09              }
10          }
```

解答：
程序代码正确,输出结果为：
True
False
True

（2.3） **程序代码如下：**

```
00          class TestA {
01              int num;
02              public Integer getLength() {
03                  num=6;
04                  return new Integer(num);
05              }
06          }
07          public class TestB extends TestA {
08              public Long getLength()
09              {
10                  num=8;
```

```
11                  return new Long(num);
12              }
13              public static void main(String[] args) {
14                  TestA aa = new TestA();
15                  TestB bb = new TestB ();
16                  System.out.println(aa.getLength().toString() +","+bb.getLength().toString() );
17              }
18          }
```

解答：第 8、11 行有错误，方法重写不能改变返回值类型
将第 8、11 行 Long 改为　Integer，
输出结果为：
6,8

（2.4） **程序代码如下：**

```
00          interface    A{
01              int x = 6;
02          }
03          class B{
04              int x =8;
05          }
06          class Test extends B implements A {
07              public void pX(){
08                  System.out.println("父类属性 x 是"+x);
09                  System.out.println("接口属性 x 是"+x);
10              }
11              public static void main(String[] args) {
12                  new Test().pX();
13              }
14          }
```

解答：第 8、9 行未明确的 x 调用，分别修改为 super.x 和 A.x 来明确，
输出结果：
为父类属性 x 是 8
接口属性 x 是 6

三、程序实现题（共 3 小题，共 30 分，每题 10 分）

（3.1） 原先设计了一个系统：包含 Person 抽象类，含有姓名（name）年龄（age）两个私有属性以及吃饭（eat）和睡觉（sleep）两个抽象方法，其包含有带参构造方法，另外设计了学生（Student）和工人（Worker）两个类，继承 Person 类，学生类多出了私有属性学号（sid）和学习（study）方法（输出我爱学习），工人类多出了私有属性工号（wid）和工作（work）方法（输出我努力工作）。现在发现漏设计了一个在职学生（EmployedStudent）类，其具备学生类的属性和方法以及工人类的方法（输出我爱学习同时努力工作），请设计两个接口 Studying 和 Working，并在此基础上按需求重新编写代码。

（3.2） 下列程序逐个下载 arrUrl 数组中的 url，请改写为多线程并行下载的形式（忽略 import 和异常处理）：

```
public class MainApp {
    private static String[] arrUrl;
    private static String localPath;
    public static void main(String[] args) {
        ... //经过某些解析操作，构建了 arrUrl 数组，每个元素为一个 url 地址，
            //并设置了 localPath 本地保存路径
        for(int i = 0; i < arrUrl.length; i++) {
            URL url = new URL(arrUrl[i]);
            String fileName = url.getFile();
            fileName                                    =
fileName.substring(fileName.lastIndexOf('/') + 1);
            InputStream is = url.openStream();
            byte[] temp = new byte[1024];
            int len = -1;
            FileOutputStream fos = new FileOutputStream(
                                        new  File(localPath  +
fileName));
            while ((len = is.read(temp)) != -1) {
                fos.write(temp, 0, len);
            }
            fos.close();
            is.close();
            System.out.println(url.getFile() + "下载完毕");
        }
    }
}
```

（3.3） 编写程序，要求用户连续 5 轮输入整数，输入的同时计算前面输入各数的乘积，若乘积超过 100000，则认为是异常，请自定义异常类，捕获并处理该异常，输出信息。

三、程序实现题（共 3 小题，共 30 分，每题 10 分）

（3.1） 原先设计了一个系统：包含 Person 抽象类，含有姓名（name）年龄（age）两个私有属性以及吃饭（eat） 和睡觉（sleep）两个方法，其包含有带参构造方法，另外设计了学生（Student）和工人（Worker） 两个类，继承 Person 类，学生类多出了私有属性学号(sid)和学习（study）方法（输出我爱学习），工人类多出了私有属性工号(wid)和工作(work)方法（输出我努力工作）。现在发现漏设计了一个在职学生（Employedstudent）类，其具备学生类的属性和方法以及工人类的方法（实现类似多继承的效果），请按需求重新设计并完成代码编写。

解答：

```
abstract class Person {
    private String name;
    private int age;
    Person(String name, int age) {
      this.name=name;
      this.age=age;
    }
    Person() { }
    public void eat(){
      System.out.println("吃饭“);
    }
    public void sleep(){
      System.out.println("睡觉“);
    }
}

public interface Working {
    public abstract void work();
}

public interface Studying {
    public abstract void study();
}


class Student extends Person implements Studying {
    private int sid;
    public Student{
      super();
    }
    public Student(String name, int age) {
      super(name,age);
    }
    public Student(String name, int age, int sid) {
```

```
        super(name,age);
        this.sid=sid;
    }

    public void study(){

        System.out.println("我爱学习");

    }
}

class Worker extends Person implements Working {
    private int wid;
    public Worker{
        super();
    }
    public Worker (String name, int age) {
        super(name,age);
    }
    public Worker (String name, int age, int wid) {
        super(name,age);
        this.wid=wid;
    }

    public void work(){

        System.out.println("我努力工作");

    }
}

class Employedstudent extends Student implements Working{
    public Employedstudent {
        super();
    }
    public Employedstudent(String name, int age) {
        super(name,age);
    }
    public Employedstudent (String name, int age, int sid) {
        super(name,age,sid);
    }

    public void work(){

        System.out.println("我努力工作同时学习");

    }
```

```
}
```

（3.2） 设计一个系统：实现生产电脑和销售电脑类，要求生产一台电脑就销售一台电脑，如果没有新电脑的生产就等待新电脑生产；如果生产出的电脑没有销售，则要等待电脑销售之后再生产，并统计出电脑生产的数量（num）。

解答:

```
public class ComputerSell {


   public static void main(String[] args) {
      Resource res=new Resource();
      new Thread(new Producer(res)).start();
      new Thread(new Consumer(res)).start();
   }


}
class Producer implements Runnable{
   private Resource resource;
   public Producer(Resource resource) {
      this.resource=resource;
   }
   public void run() {
      for(int x=0;x<50;x++) {
         try {
            resource.make();
```

```
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
class Consumer implements Runnable{
    private Resource resource;
    public Consumer(Resource resource) {
        this.resource=resource;
    }
    public void run() {
        for(int x=0;x<50;x++) {
            try {
                resource.get();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
class Resource{
```

```
    private Computer computer;


    public synchronized void make()throws Exception {
        if(computer!=null) {
            wait();
        }
        Thread.sleep(100);
        computer=new Computer("电脑");
        System.out.println("生产电脑");
        notifyAll();
    }
    public synchronized void get() throws Exception {
        if(this.computer==null) {
            wait();
        }
        Thread.sleep(100);
        System.out.println("取走电脑"+computer.toString());
        this.computer=null;
        notifyAll();
    }
}
class Computer{
```

```
    private static int count=0;
    private String name;
    public Computer(String name) {
        this.name=name;
        count++;
    }
    public String toString() {
        return "第"+count+"台电脑:name:"+this.name;
    }
}
```

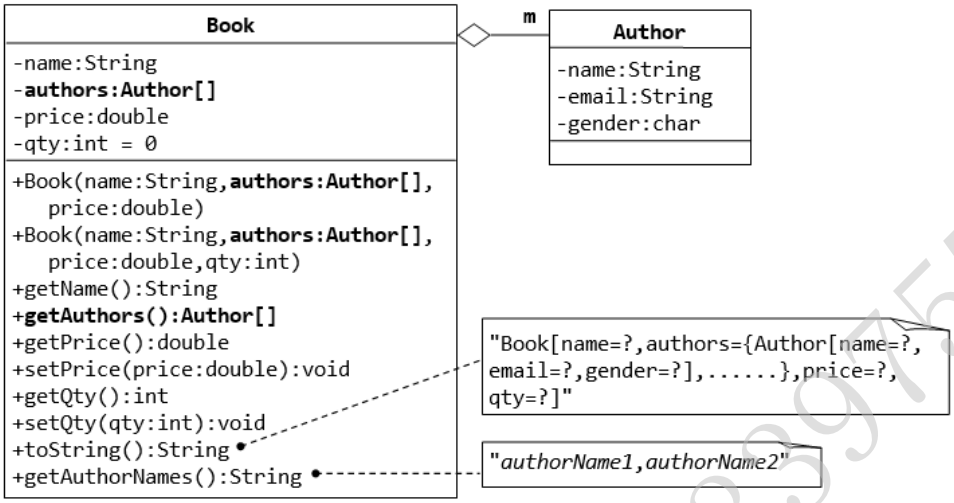
（3.3） 编写程序，要求输入若干整数,输入的同时计算前面输入各数的乘积，若乘积超过 100000，则认为是异常，捕获并处理该异常，输出信息。

解答:

```
public class Test {
   public static void main (string [ ] args){
      scanner sc=new Scanner (system.in) ;
      system. out.println ( "请输入一个整数") ;
      int i =sc.nextInt ( ) ;
      try {
          while ( i<=10000 ){
             system. out.println ( "请再次输入一个整数");
             int j =sc.nextInt ( ) ;
             i*=j;
             system.out.println ( "两个数相乘="+i) ;
          }
      } catch (Exception e ) {
          e.printstackTrace ( ) ;
      }
   }
}
```

四、综合题（共 2 题，共 30 分，每题 15 分）

（4.1） 下图中描述了对 Book（图书）和 Author（作者）这两个类的建模要求：



说明:

- 编写 Book 类和 Author 类的定义。注意，一本书可能有一个或多个作者；
- Book 类中包含相关私有属性和公有方法，这些方法根据其名称，请给出适当的实现代码；Author 类的方法没有明确给出，根据需要编写；
- Book 类的 toString 方法和 getAuthorNames 方法，请按照图中提供的输出要求，给出适当的实现代码；
- 在 main 方法，创建如下两本图书及其作者信息

“Java 编程思想”，价格 98.00，数量 10 本	姓名：埃克尔　邮件：eckel@gmail.com　性别：M
“Java 大学教程”，价格 89.00	姓名：保罗　邮件：paul@gmail.com　性别：M
	姓名：哈维　邮件：harvey@gmail.com　性别：M

然后，打印出各 Book 的字符串表示形式和 getAuthorNames 方法的返回结果。

答：（参考答案）

```
import java.util.Arrays;
public class Book {
    private String name;
    private Author[] authors;
    private double price;
    private int qty=0;                                          （2 分）
    public Book(String name, Author[] authors, double price) {
        this.name = name;
        this.authors = authors;
        this.price = price;
    }                                                           （2 分）
```

```
    public Book(String name, Author[] authors, double price, int qty) {
        this.name = name;
        this.authors = authors;
        this.price = price;
        this.qty = qty;
    }                                                              (2 分)

    public String getName() {          return name;     }
    public Author[] getAuthors() {          return authors;  }
    public double getPrice() {          return price;      }
    public void setPrice(double price) {       this.price = price;    }
    public int getQty() {       return qty;   }
    public void setQty(int qty) {       this.qty = qty;    }                (2 分)

    public String toString() {
        return "Book [name=" + name + ", authors=" + Arrays.toString(authors)
+ ", price=" + price + ", qty=" + qty   + "]";
    }                                                              (2 分)

    public String getAuthorNames()
    {          return Arrays.toString(authors); }                  (2 分)


    public static void main(String[] args) {
        Book book1=new Book("Java 编程思想",new Author[]{new Author("埃克
尔","eckel@gmail.com",'M')},98.0,10);

        Book book2=new Book("Java  大学教程",new Author[]{new Author("保
罗","paul@gmail.com",'M'),new Author("哈维","harvey@gmail.com",'M')},89.0);

        System.out.println(book1);
        System.out.println(book1.getAuthorNames());
        System.out.println(book2);
        System.out.println(book2.getAuthorNames());                (3 分)

    }
}

class Author{
    private String name;
    private String email;
    private char gender;
```

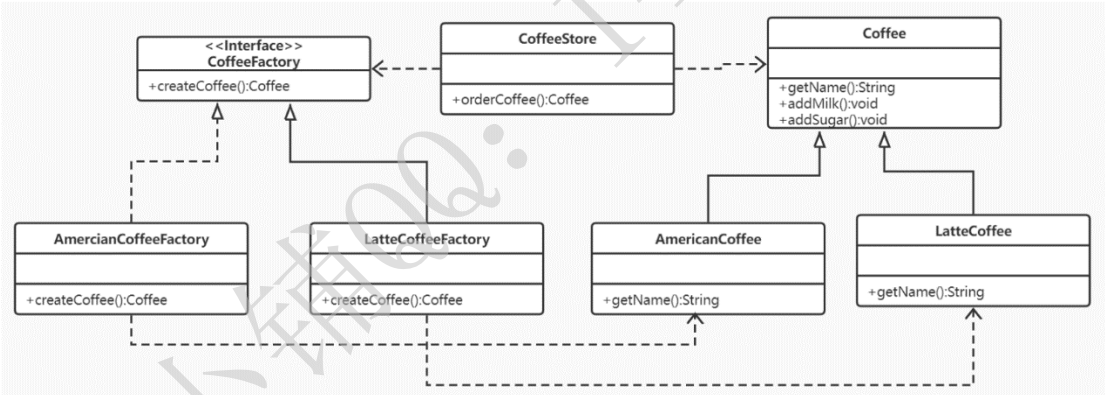
```
    public Author(String name, String email, char gender) {
        this.name = name;
        this.email = email;
        this.gender = gender;
    }
    public String toString() {
        return "Author [name=" + name + ", email=" + email + ", gender=" +
gender + "]";
    }

}
```

（3 分）

（4.2） 使用工厂方法模式设计一盒咖啡店点餐系统：设计一个咖啡类（Coffee），并定义其两个子类（美式咖啡【AmericanCoffee】和拿铁咖啡【LatteCoffee】），同时为每种咖啡类型编写相应的工厂类。；再设计一个咖啡店类（CoffeeStore），咖啡店具有点咖啡的功能

解答：




```
public abstract class Coffee {
    public abstract String getName();
    //加糖
    public void addsugar(){
        System.out.println("加糖");
    }
    //加奶
    public void addMilk(){
        System.out.println("加奶");
    }
}
//美式咖啡
public class AmericanCoffee extends Coffee {

    @Override
```

```
    public String getName() {
        return "美式咖啡";
    }

}

//拿铁咖啡
public class LattaCoffee extends Coffee {
    @Override
    public String getName() {
        return "拿铁咖啡";
    }
}


public interface CoffeeFactory {
    //创建咖啡对象的方法
    Coffee createCoffee();
}

public class AmericanCoffeeFactory implements CoffeeFactory{
    @Override
    public Coffee createCoffee() {
        return new AmericanCoffee();
    }
}

public class LattaCoffeeFactory implements CoffeeFactory{
    @Override
    public Coffee createCoffee() {
        return new LattaCoffee();
    }
}


public class CoffeeStore {
    private CoffeeFactory factory;
    public void setFactory(CoffeeFactory factory){
        this.factory=factory;
    }

    public Coffee orderCoffee(){

        //调用生产咖啡的方法
```

```
            Coffee coffee = factory.createCoffee();
            //添加配料
            coffee.addMilk();
            coffee.addsugar();
            return coffee;
        }
}
public class Test{
        public static void main(String[] args) {
            //创建咖啡店对象
            CoffeeStore store=new CoffeeStore();
            //创建对象
            CoffeeFactory factory=new AmericanCoffeeFactory();
            store.setFactory(factory);
            //点咖啡
            Coffee coffee = store.orderCoffee();
            System.out.println(coffee.getName());
        }
}
```