考试科目名称 计算与软件工程Ⅰ

考试方式： 闭卷 考试日期____年___月___日 教师_____

系（专业）_____ 年级_____ 班级_____

学号_____ 姓名_____ 成绩_____

题号	一	二	三	四	五	六	七	八	九	十
分数										

得分	

一、 判断题（本题满分 15 分）

1、一个 **Object** 类型的变量只能引用实例，不能引用数组。 （**F**）

2、**Java** 的源代码中定义几个类，编译结果就生成几个以**.class** 为后缀的字节码文件。（**T**）

3、如果父类对象引用指向的实际是一个子类对象，则这个父类对象的引用可以用强制类型转换转化成子类对象的引用。 （**T**）

5、**Java** 的字符类型采用的是 **ASCII** 编码。 （**F**）

6、**Java** 的各种数据类型占用固定长度，与具体的软硬件平台环境无关。 （**T**）

7、方法的覆盖与域的隐藏一样，父类的同名方法在子类中仍然占有自己独立的内存空间。 （**F**）

8、在 **Java** 中变量和对象实例所占用的资源都只是内存资源。 （**F**）

9、**Java** 中的 **String** 类的对象既可以是字符串常量，也可以是字符串变量。（**F**）

10、**Java** 的屏幕坐标是以像素为单位，容器的左上角被确定为坐标的起点。（**T**）

11、**Java** 中，并非每个事件类都只对应一个事件。 （**T**）

12、一个类只能有一个父类，但一个接口可以有一个以上的父接口。 （**T**）

13、**Java** 中对象实例只能通过引用它的变量对其进行操作。 （**T**）

14、无论在一个类中有没有定义构造函数，该类均存在一个不带任何参数的默认构造函数。 （**F**）

15、**Java** 中输入输出流序列中的数据既可以是未经加工的原始二进制数据，也可以是

经一定编码处理后符合某种格式规定的特定数据。 （T）

得分	

二、 简答题（本题满分 15 分）

1. 什么是 Java 中的接口？接口和抽象类有哪些相同与不同之处？

接口（interface）是一种与类相似的结构，只包含常量和抽象方法。多继承性可通过实现这样的接口而获得。（2 分）

抽象类和接口都可以用于模拟共同特征。抽象类可以用于描述如父子关系之类的强是关系；而接口用于描述对象拥有某种属性这样的弱是关系；

抽象类与接口都用于抽象，但是抽象类可以有自己的部分实现，而接口则完全是一个标识(同时有多重继承的功能)，在接口中，数据必须是常量，而抽象类可以有非常量的数据域；

所有类包括抽象类共享同一个根 Object 类，但接口没有共同的根。（3 分）

2. 简述 Error 与 Exception 的区别。

Error 表示系统级的错误和程序不必处理的异常， 举例 （3 分）

Exception 表示需要捕捉或者需要程序进行处理的异常，举例 （2 分）

3. 简述 Applet 与 Application 的区别 ，并分别写程序举例。

	Application	Applet （3 分）
运行程序	Java 解释器调用	web 浏览器运行网页调用其上的 applet
参数	可以从命令行获得	从 HTML 页面获得
是否有 main()	有	无，有 init()，start()等
安全运行限制	无	有

举例（2 分）

得分	

三、阅读程序，并写出结果（本题满分 16 分）

（1）

```
class Value{
    public int i = 15;   }
public class Test{
    public static void main(String argv[]){
       Test t = new Test();
       t.first();     }
    public void first(){
       int i = 5;
```

```
        Value v = new Value();
        v.i = 25;
        second(v, i);
       System.out.println(v.i);     }
     public void second(Value v, int i){
        i = 0;
        v.i = 20;
        Value val = new Value();
        v = val;
        System.out.println(v.i + " " + i);  }
}
```

输出结果： 15 0 20

（2）

```
public class HidingDemo {
   public static void main(String[] args) {
      A x = new B();
      System.out.println("(1) x.i is " + x.i);
      System.out.println("(2) (B)x.i is " + ((B)x).i);
      System.out.println("(3) x.j is " + x.j);
      System.out.println("(4) ((B)x).j is " + ((B)x).j);
      System.out.println("(5) x.m1() is " + x.m1());
      System.out.println("(6) ((B)x).m1() is " + ((B)x).m1());
      System.out.println("(7) x.m2() is " + x.m2());
      System.out.println("(8) x.m3() is " + x.m3());     }
}
class A {
   public int i = 1;
   public static int j = 11;
   public static String m1() {       return "A's static m1";     }
   public String m2() {     return "A's instance m2";       }
   public String m3() {     return "A's instance m3";    }
}
class B extends A {
   public int i = 2;
   public static int j = 12;
   public static String m1() {    return "B's static m1";    }
   public String m2() {     return "B's instance m2";    }
}
```

输出结果： (1) x.1 is 1; (2)(B)x.i is 2; (3) x.j is 11; (4)((B)x).j is 12;
(5) x.m1() is A's static m1; (6) ((B)x).m1 is B's static m1;
(7) x.m2() is B's instance m2; (8) x.m3() is instance m3

（3）

```
public class E{
        int l = 2;
        public class Inner{
            int l = 1;
            public void printl(int l){
                System.out.println(l);
                System.out.println(this.l);
                System.out.println(E.this.l);
            }
        }

        public static void main(String[] args) {
            E e = new E();
            Inner in = e.new Inner();
            in.printl(0);
        }
    }
```

输出结果：0 1 2

（4）

```
class MyParent {
        int x, y;
        MyParent(int x, int y) { this.x = x;   this.y = y; }
        public int addMe(int x, int y) { return this.x + x + y + this.y; }
        public int addMe(MyParent myPar){
            return addMe(myPar.x, myPar.y); }
    }
    class MyChild extends MyParent{
        int z;
        MyChild (int x, int y, int z){ super(x,y); this.z = z; }
        public int addMe(int x, int y, int z) {
          return this.x + x + this.y + y + this.z + z; }
        public int addMe(MyChild myChi){
          return addMe(myChi.x, myChi.y, myChi.z); }
        public int addMe(int x, int y) {
          return this.x + x + this.y + y; }
    }
    public class MySomeOne{
        public static void main(String args[]){
            MyChild myChi = new MyChild(10, 20, 30);
            MyParent myPar = new MyParent(10, 20);
            int x = myChi.addMe(10, 20, 30);
            int y = myChi.addMe(myChi);
            int z = myPar.addMe(myPar);
            System.out.println(x +","+ y +","+z);}
```

```
}
```

输出结果：__120，120，60__

得分	

四、编程填空题（每空格 2 分，共 20 分）

1. 编程计算斐波那挈数,下面给出了该程序的主要部分，请填写其中的空缺。

```
import java.io.*;
public class Fibonacci {
    static int value=1;
    static int Fibonacci (int n){
        int temp;
        if (n==0 || n==1)
            __return n____;
        else {
            temp=_ temp = f(n - 1) + f(n - 2)___;
            if (temp>value) {
                System.out.print(value+" ");
                value=temp;   }
            return temp;   }
    }
}
```

2. 实现有自定义异常类的处理程序

```
public Class example {
public static void main ( String args [ ]) {
  Worker dobj =   new Worker( );
  for   (int   i = 0; i <3; i++) {
  __try___ {
       dobj.f1(i );
       System.out.println("No exception," + i ); }
  __catch__ (_Exception__ e){
       System.out.println(e.toString() + " i = " + i ); }
  _finally__ {
      System.out.println("Finally code " ); }
  }
 }
}

Class Worker {
  public void f1(int val) __throws___ Except1 {
       System.out.println("The number is" + i );
       _throw___      __new__ Except1 ("Gotcha!");}
}

Class Except1 extends _Except_
__   {
```

```
    private String theMessage_;
    public Except1(String aMsg) {theMessage_ = aMsg;}
    public String toString() {return “Except type 1” + theMessage_;}
}
```

得分	

五、阅读程序，回答问题（共 20 分）：

1. 回答程序后问题

```
1:  public class Output1 {
2:      public static void main(String arge[]) {
3:          int i=0;
4:          for ( char ch = 97; ch<113; ch++,i++) {
5:              if( i % 8 == 0 )
6:                  System.out.println(" ");
7:              System.out.print("\t" +ch);
8:          }
9:      }
10: }
```

(1)程序第 5、6 行的 if 语句的功能是什么？ (2 分)

每打印 8 个字符，则换行

(2)程序输出的结果有几行？(3 分)

输出的结果有 2 行

2. 回答程序后问题

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class KeyExample extends JFrame {
   private KeyboardPanel keyboardPanel = new KeyboardPanel();
      public KeyExample() {   add(keyboardPanel);
                              keyboardPanel.setFocusable(true);   }

   public static void main(String[] args) {
      KeyExample frame = new KeyExample();
      frame.setTitle("KeyExample");
      frame.setLocationRelativeTo(null);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setSize(300, 300);
      frame.setVisible(true);      }

  static class KeyboardPanel extends JPanel {
      private int x = 100;
      private int y = 100;
```

```
    private char keyChar = 'A';
    public KeyboardPanel() {
      addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
          switch (e.getKeyCode()) {
            case KeyEvent.VK_DOWN: y += 10; break;
            case KeyEvent.VK_UP: y -= 10; break;
            case KeyEvent.VK_LEFT: x -= 10; break;
            case KeyEvent.VK_RIGHT: x += 10; break;
            default: keyChar = e.getKeyChar();       }
          repaint();     }
      });
    }
   protected void paintComponent(Graphics g) {
      super.paintComponent(g);
      g.setFont(new Font("TimesRoman", Font.PLAIN, 24));
      g.drawString(String.valueOf(keyChar), x, y);      }
  }
}
```

(1)解释函数 **paintComponent**（）的作用是什么？**(2** 分**)**

Swing 组件使用该方法绘制图形，当组件第一次显示或需要显示时当该方法被自动调用来绘制图形

(2)该程序的功能是什么？**(3** 分**)**

显示用户输入的字符，用户可以使用箭头上下左右移动字母。

3. 解释程序中语句的含义，并按要求写在后面的横线上：

纯文本文件 **f1.txt** 中的内容是 **abcd**。下面的程序将 **f1.txt** 文件中的内容写到 **f2.txt** 文件中和屏幕上

```
import java.io.*;
public class filecopy {
  public static void main(String[] args) {
    try { StringBuffer str=new StringBuffer();建立字符串缓冲流
        FileInputStream  fin=new  FileInputStream("f1.txt");建立以 f1.txt 为数据源的输入流
        FileOutputStream fout=new FileOutputStream("f2.txt");建立以 f2.txt 为数据宿的输出流
        int c;
        while((c=fin.read())!=-1)循环条件是文件内容没有读完
        {   fout.write(c);  把从 f1.txt 读入的内容写到输出流中
```

```
            str.append((char)c); 把读入的内容转变成字符添加到字符缓冲区中   }
          fin.close();关闭输入流
          fout.close();   关闭输出流
          String str2=str.toString();把字符缓冲区的内容转换成字符串
          System.out.println(str2); 显示的结果是 abcd
      }catch(Exception c)   { System.out.println(c);}
    }
}
```

得分	

六、**编程题**（共 **14** 分）：

编程实现 **Employee** 类（有属性 **name[**姓名**], salary[**工资**], hireday[**雇佣日期**]**和相应的存取方法），实现如下排序接口 **Sortable**：

interface Sortable { int compareTo(Sortable a); }

可按工资对员工进行排序。接受一组用户输入员工信息，显示员工排序结果并写入文件 **data.txt** 中。

```
import java.util.*;
public class Employee implements Sortable {          （1 分）
   String name;
   int salary;
   Date hireday;
   Public String getName() { … };                    （3 分）
   Public int getSalary() {…};
   Public Date get Hireday(){…};
   Public static int compareTo(Sortable a) {         （2 分）
     return   (this. getSalary() - a. getSalary()); }
   public static void sort(Object[] list) {          （4 分）
     Object currentMax;
     Int currentMaxIndex;
     For (int I = list.length -1; I >= 1; i--) {
      currentMax = list[i];
      currentMaxIndex =1;
      …//find the maximum in the list[0..i];
      …//swap list[i] with list[currentMaxIndex] if necessary;}
   Public static void main(String[] args) {          （4 分）
      …//input employees into list[];
      …//sort salary;
      …//write to data.txt
     }
```