《Java 基础入门》课后习题答案

第 1 章　Java 开发入门

一、填空题

1、 Java EE、Java SE、Java ME

2、 JRE

3、 javac

4、 bin

5、 path、classpath

二、判断题

1、对　2、错　3、对　4、对　5、错

三、选择题

1、ABCD　2、C　3、D　4、B　5、B

四、简答题

1、 面向对象、跨平台性、健壮性、安全性、可移植性、多线程性、动态性等。

2、 JRE（Java Runtime Environment，Java 运行时环境），它相当于操作系统部分，提供了 Java 程序运行时所需要的基本条件和许多 Java 基础类，例如，IO 类、GUI 控件类、网络类等。JRE 是提供给普通用户使用的，如果你只想运行别人开发好的 Java 程序，那么，你的计算机上必须且只需安装 JRE。

JDK（Java Development Kit，Java 开发工具包），它包含编译工具、解释工具、文档制作工具、打包工具多种与开发相关的工具，是提供给 Java 开发人员使用的。初学者学习和使用 Java 语言时，首先必须下载和安装 JDK。JDK 中已经包含了 JRE 部分，初学者安装 JDK 后不必再去下载和安装 JRE 了。

3、 Java 程序运行时，必须经过编译和运行两个步骤。首先将后缀名为.java 的源文件进行编译，生成后缀名为.class 的字节码文件，然后 Java 虚拟机将字节码文件进行解释执行，并将结果显示出来。

五、编程题

1、参考答案

HelloWorld.java

```
1  public class HelloWorld {
2      public static void main(String[] args) {
3          System.out.println("这是第一个 Java 程序!");
4      }
5  }
```

第 2 章　Java 编程基础

一、填空题

1、 true 和 false

2、 基本数据类型、引用数据类型

3、 & && | ||

4、 5

5、 56

二、判断题

1、错 2、对 3、错 4、对 5、错

三、选择题

1、AD 2、C 3、C 4、B 5、A

四、简答题

1、Java 语言的八种基本数据类型有：byte 字节型，占一个字节。short 短整型，占两个字节。int 整型，占 4 个字节。long 长整型，占 8 个字节。float 单精度浮点型，占 4 个字节。double 双精度浮点型，占 8 个字节。char 字符型，占两个字节。boolean 型，表示逻辑值，有 true 和 false 两个值，分别占一个字节。

2、如果使用“&”在表达式之间进行连接，那么无论任何情况，“&”两边的表达式都会参与计算。如果使用“&&”进行连接，当“&&”左边的表达式为 false，则不会执行其右边的表达式。例如定义 int x = 2,y = 0; boolean b = x < y & x / 2 > 0 表达是会发生被 0 除异常，因为 x / y 的表达式执行了。而 boolean b = x < y & x / 2 > 0 是不会出现这种异常的，因为 x < y 为 false，表达式 x / y 不会执行。

3、方法重载指的是在一个类中可以声明多个同名的方法，而方法中参数的个数或者数据类型不一样。调用这些同名的方法时，JVM 会根据实际参数的不同绑定到不同的方法。

五、编程题

1、参考答案

```
public class getSum {
    public static void main(String[] args) {
        int sum = 0;
        for (int i = 1; i < 100; i++) {
            if (i % 2 != 0)
                sum += i;
        }
        System.out.println(sum);
    }
}
```

2、参考答案

```
public class ArraySort {
    public static void main(String[] args) {
        int[] arr = { 25, 24, 12, 76, 101, 96, 28 };
        for (int i = 0; i < arr.length - 1; i++) {
            // 定义内层循环
            for (int j = 0; j < arr.length - i - 1; j++) {
                if (arr[j] > arr[j + 1]) { // 比较相邻元素
                    // 下面的三行代码用于交换两个元素
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
```

```
                }
            }
            for (int i = 0; i < arr.length; i++) {
                System.out.print(arr[i] + " "); // 打印元素和空格
            }
        }
    }
```

第 3 章　面向对象（上）

一、填空题

1、封装、继承、多态

2、this

3、private、default、protected、public

4、静态变量

5、private

二、判断题

1、对　2、对　3、对　4、对　5、错

三、选择题

1、D　2、ABC　3、A　4、D　5、BD

四、简答题

1、构造方法是类的一个特殊成员，它会在类实例化对象时被自动调用。而普通方法只有在使用的时候才会被调用。在定义构造方法时要求方法名与类名相同、在方法名的前面没有返回值类型的声明、在方法中不能使用 return 语句返回一个值。

2、（1）方法名与类名相同；（2）在方法名的前面没有返回值类型的声明；（3）在方法中不能使用 return 语句返回一个值，但是可以单独写 return 语句来作为方法的结束。

3、Java 面向对象有三大特性，封装是将对象的属性和行为封装起来，不需要让外界知道具体实现细节；继承是可以在无需重新编写原有类的情况下，对原有类的功能进行扩展；多态指的是在一个类中定义的属性和功能被其他类继承后，当把子类对象直接赋值给父类引用变量时，相同引用类型的变量调用同一个方法所呈现出的多种不同行为特性。

五、编程题

1、参考答案

```
class Student {
    private String name;
    private double grade;
    public Student() {
    }
    public Student(String name, double grade) {
        this.name = name;
        this.grade = grade;
    }
    public String getName() {
        return name;
    }
```

```
    public void setName(String name) {
        this.name = name;
    }
    public double getGrade() {
        return grade;
    }
    public void setGrade(double grade) {
        this.grade = grade;
    }
}
public class Test01 {
    public static void main(String[] args) {
        Student stu1 = new Student();
        stu1.setName("zhangsan");
        stu1.setGrade(99);
        Student stu2 = new Student("lisi", 100);
    }
}
```

2、参考答案

```
public class Test {
    public static void main(String[] args) {
        int n = 8;
        int num = getNum(n);
        System.out.println("第"+n+"个数的数值是："+num);
    }
   public static int getNum(int n) {
      if (n == 1){
       return 0;
      } else if (n == 2 || n == 3) {
       return 1;
      } else {
       return getNum(n - 1) + getNum(n - 2);
      }
   }
}
```

第 4 章　面向对象（下）

一、填空题

1、方法，抽象类

2、子类、父类、基类

3、final

4、Object

5、参数列表、“->”、表达式主体

二、判断题

1、错　2、对　3、错　4、对　5、错

三、选择题

1、B　2、ABC　3、C　4、C　5、A

四、简答题

1、在继承关系中，子类的方法与父类的某一方法具有相同的方法名、返回类型和参数列表，则称子类的该方法重写(覆盖)父类的方法。

2、多态意味着一个对象有着多种形态，可以在特定的情况下，表现不同的状态，从而对应着不同的属性和方法。简单的说，多态就是使用父类类型的变量引用子类对象，根据被引用子类对象的特性，程序会得到不同的运行效果。

3、在 Java 中，使用 abstract 关键字修饰的类称之为抽象类。抽象类是不能被实例化的，通常需要写一个子类来继承抽象类，同时实例化子类来获得该类的对象。抽象类通常用于表示一种抽象的概念。接口可以说是一种特殊的抽象类，接口中只能定义常量、抽象方法、静态方法和默认方法。由于接口的特殊性，在定义时需要使用 interface 关键字。

五、编程题

1、参考答案

```
class Student {
    public String name;
    public int age;
    public Student(String name,int age){
        this.name=name;
        this.age=age;
    }
    public void show(){
        System.out.println("name: "+name+" age: "+age);
    }
}
class UnderGraduate extends Student{
    public String degree;
    public UnderGraduate(String name,int age,String degree){
        super(name, age);
        this.degree=degree;
    }
    public void show(){
        System.out.println("name: "+name+" age: "+age+" degree: "+degree);
    }
}
public class Test01{
    public static void main(String[] args) {
        Student student = new Student("zhangsan", 16);
        student.show();
        UnderGraduate underGraduate = new UnderGraduate("lisi", 20, "bechalor");
        underGraduate.show();
    }
```

```
}
```

2、参考答案

```
interface Shape {
    double area(double givenValue);
}
class Square implements Shape{
    public double area(double sideLength) {
     return sideLength*sideLength;
    }
}
class Circle implements Shape{
    public double area(double r) {
      return Math.PI*r*r;
    }
}
public class Test02 {
    public static void main(String[] args) {
        Shape square = new Square();
        Shape circle = new Circle();
        System.out.println(square.area(2));
        System.out.println(circle.area(3));
    }
}
```

第 5 章　Java 中的常用类

一、填空题

1、 String、StringBuffer

2、 Date、Calendar、DateFormat

3、 DateFormat

4、 静态

10、edcba

二、判断题

1、错　2、错　3、对　4、错　5、对

三、选择题

1、C　　2、C　　3、B　　4、A　　5、B

四、简答题

1、String 类是不可变类，即字符串值一旦初始化后就不可能改变。StringBuffer 是可变字符串类，类似 String 的缓冲区，可以修改字符串的值。

2、Date 类用来表示某个特定的瞬间，能够精确到毫秒。而在实际应用中，往往需要把一个日期中的年、月、日等信息单独返回进行显示或处理，这个类中的大部分方法都已被标记过时。Calender 类基本取代了 Date 类，该类中定义了一系列用于完成日期和时间字段操作的方法。

Calendar 的 getTime()方法，getTime()返回一个表示 Calendar 时间值的 Date 对象，同时 Calendar 有

一个 setTime(Date date)方法，setTime()方法接收一个 Date 对象，将 Date 对象表示的时间值设置给 Calendar 对象，通过这两个方法就可以完成 Date 和 Calendar 对象之间的转换。

3、自动装箱是指将基本数据类型的变量赋给对应的包装类变量，反之，拆箱是指将包装类对象类型直接赋给一个对应的基本数据类型变量。

五、编程题

1、 参考答案

```
public class Test01 {
    public static void main(String[] args) {
        String str = "HelloWorld";
        // 字符串转成 char 数组
        char[] ch = str.toCharArray();
        StringBuffer buffer = new StringBuffer();
        for (int i = str.length() - 1; i >= 0; i--) {
            if (ch[i] >= 'A' && ch[i] <= 'Z') {
                buffer.append(String.valueOf(ch[i]).toLowerCase());
            } else if (ch[i] >= 'a' && ch[i] <= 'z') {
                buffer.append(String.valueOf(ch[i]).toUpperCase());
            }
        }
        System.out.println(buffer.toString());
    }
}
```

2、 参考答案

```
import java.util.Random;
public class Test {
    public static void main(String[] args) {
        Random rand = new Random();
        for (int i = 0; i < 5; i++) {
            int num = 20 + rand.nextInt(11);
            System.out.println(num);
        }
    }
}
```

第 6 章　集合

一、填空题

1、Comparator

2、hashNext()、next()

3、键、值

4、ArrayList、LinkedList，HashSet、TreeSet，HashMap、TreeMap

5、forEach(Consumer action)

二、判断题

1、错　2、对　3、对　4、错　5、对

三、选择题

1、BC　　2、D　　3、C　　4、D　　5、ABC

四、简答题

1、为了使程序能方便的存储和操作数目不固定的一组数据，JDK 提供了一套类库，这些类都位于 java.util 包中，统称为集合。集合框架中常用的接口和类有，List、Set、ArrayList、HashSet、Map、HashMap、TreeMap。

2、List 的特点是元素有序、可重复。List 接口的主要实现类有 ArrayList 和 LinkedList。Set 的特点是元素无序、不可重复。Set 接口的主要实现类有 HashSet 和 TreeSet。Map 的特点是存储的元素是键(Key)、值(Value)映射关系，元素都是成对出现的。Map 接口的主要实现类有 HashMap 和 TreeMap。

3、Collection 是一个单例集合接口。它提供了对集合对象进行基本操作的通用方法。Collections 是一个工具类。它包含各种有关集合操作的方法。

五、编程题

1、参考答案

```
import java.util.*;
public class Test02 {
    public static void main(String[] args) {
        HashSet hashSet = new HashSet();
        Person p1 = new Person("Jack",25);
        Person p2 = new Person("Rose",23);
        Person p3 = new Person("Jack",27);
        hashSet.add(p1);
        hashSet.add(p2);
        hashSet.add(p3);
        for(Object obj:hashSet){
            Person p=(Person)obj;
            System.out.println(p.name+":"+p.age);
        }
    }
}
class Person{
    String name;
    int age;
    public Person(String name, int age) {
        super();
        this.name = name;
        this.age = age;
    }
    public int hashCode() {
       return name.hashCode();
    }
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
```

```
            return false;
        Person p = (Person) obj;
        return p.name.equals(this.name);
    }
}
```

2、参考答案

```
import java.util.*;
public class Test03 {
    public static void main(String[] args) {
        TreeMap map = new TreeMap(new MyComparator());
        map.put("1", "Lucy");
        map.put("2", "Lucy");
        map.put("3", "John");
        map.put("4", "Smith");
        map.put("5", "Amanda");
        for (Object key : map.keySet()) {
            System.out.println(key + ":" + map.get(key));
        }
    }
}
class MyComparator implements Comparator {
    public int compare(Object obj1, Object obj2) {
        String ele1 = (String) obj1;
        String ele2 = (String) obj2;
        return ele2.compareTo(ele1);
    }
}
```

第 7 章　IO（输入输出）

一、填空题

1、 字节流、字符流

2、 Channel、Buffer

3、 InputStreamReader、OutputStreamWriter

4、 Buffer、Channel、Selector

5、RandomAccesseFile

二、判断题

1、 错　2、对　 3、对　4、对　5、错

三、选择题

1、AB　 2、C　 3、AB　　4、A　5、B

四、简答题

1、Java 程序通过 I/O 流来完成输入和输出，流是输入或输出信息的抽象。流通过 Java 的输入/输出系统与外设连接进行数据通信。流是抽象的对象，具体实现代码在 java.io 包中。

2、字节流的两个基类是 InputStream 和 OutputStream，字符流的两个基类是 Reader 和 Writer，它们都是 Object 类的直接子类，字节流是处理以 8 位字节为基本单位的字节流类；Reader 和 Writer 类是专门处理 16 位字节的字符流类。

3、Java 中的 NIO 是为替代传统标准的 I/O 而出现的。与标准的 IO 相比，Java NIO 提供了一种与 I/O 不同的工作方式。NIO 采用内存映射文件的方式来处理输入/输出，它将文件或文件的一段区域映射到内存中，这样就可以像访问内存一样来访问文件了。

在标准 IO 中，使用的是字节流和字符流，而在 NIO 中，使用的是通道（Channel）和缓冲区（Buffer）。数据总是从通道读入缓冲区，或从缓冲区写入通道。

NIO 主要有三大核心部分：Buffer、Channel 和 Selector。其中 Buffer 可以被看成是一个容器，其本质是一个数组缓冲区，读入或写出到 Channel 中的所有对象都会先放在 Buffer 中；Channel 是对传统的输入/输出的模拟，在 NIO 中，所有的数据都需要通过通道流的形式传输；Selector（选择器）用于监听多个通道的事件（例如：连接打开、数据到达等），主要用于多线程处理。

五、编程题

1、参考答案

```
import java.io.*;
public class Test01 {
    public static void main(String[] args) throws Exception {
        // 字节流拷贝
        FileInputStream in = new FileInputStream("E:/src.txt");
        FileOutputStream out = new FileOutputStream("E:/des1.txt");
        byte[] buf = new byte[1024];
        int len;
        while ((len = in.read(buf)) != -1) {
            out.write(buf, 0, len);
        }
        in.close();
        out.close();
        // 字符流拷贝
        BufferedReader bf = new BufferedReader(new FileReader("E:/src.txt"));
        BufferedWriter bw = new BufferedWriter(new FileWriter("E:/des2.txt"));
        String str;
        while ((str = bf.readLine()) != null) {
            bw.write(str);
            bw.newLine();
        }
        bf.close();
        bw.close();
    }
}
```

2、参考答案

```
import java.io.*;
public class Test02 {
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
```

```
            String password = "";
            boolean b = false;
            for (int i = 0; i < 5; i++) {
                System.out.println("请输入密码:");
                password = br.readLine();
                if (password.equals("123456")) {
                    System.out.println("恭喜你进入游戏");
                    b = true;
                    break;
                }
            }
            if (!b) {
                System.out.println("密码错误，游戏结束");
                System.exit(0);
            }
        }
    }
```

第 8 章　GUI（图形用户界面）

一、填空题

1、 GUI

2、 事件监听器

3、Swing

4、 WindowListener、windowClosing(WindowEvent e)

5、 JMenuBar、JMenu、JMenuItem

二、判断题

1、对　2、错　3、对　4、错　5、错

三、选择题

1、B　2、ABD　3、D　4、ABD　5、C

四、简答题

1、参考答案

- 通过实现 XxxListener 接口或者继承 XxxAdapter 类实现一个事件监听器类，并对处理监听动作的方法进行重写
- 创建事件源对象和事件监听器对象
- 调用事件源的 addXxxLisntener()方法，为事件源注册事件监听器对象

2、Swing 工具在 AWT 的基础上提供了 8 种布局管理器，分别为 BorderLayout（边界布局管理器）、BoxLayout（箱式布局管理器）、CardLayout（卡片布局管理器）、FlowLayout（流式布局管理器）、GridBagLayout（网格包布局管理器）、GridLayout（网格布局管理器）、GroupLayout（分组布局管理器）和 SpringLayout（弹性布局管理器）。

3、参考答案

- 事件源（Event Source）：事件发生的场所，通常就是产生事件的组件，例如窗口、按钮、菜单等。
- 事件对象（Event）：封装了 GUI 组件上发生的特定事件（通常就是用户的一次操作）。
- 监听器（Listener）：负责监听事件源上发生的事件，并对各种事件做出相应处理的对象（对象中

包含事件处理器）。

五、编程题

1、参考答案

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class MyMouseHandler extends JFrame {
    public MyMouseHandler() {
        final JLabel label = new JLabel("此处显示鼠标右键点击的坐标");
        label.setOpaque(true);
        label.setBackground(Color.PINK);
        this.add(label, BorderLayout.NORTH);
        this.setSize(300, 200);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent e) {
                if (e.getButton() == MouseEvent.BUTTON1) {
                    int x = e.getX();
                    int y = e.getY();
                    String banner = "鼠标当前点击位置的坐标是" + x + "," + y;
                    label.setText(banner);
                }
            }
        });
        this.setVisible(true);
    }
    public static void main(String[] args) {
        new MyMouseHandler();
    }
}
```

2、参考答案

```
import java.awt.*;
import java.util.*;
import javax.swing.*;
import java.awt.event.*;
public class Information extends JFrame {
    // 窗口 NORTH 部的 JPanel 面板
    private JPanel panel = new JPanel();
    // 爱好标签
    private JLabel lb1 = new JLabel("兴趣");
    // 三个表示爱好的 JCheckBox 复选框
    private JCheckBox cb1 = new JCheckBox("羽毛球");
    private JCheckBox cb2 = new JCheckBox("乒乓球");
```

```
    private JCheckBox cb3 = new JCheckBox("唱歌");
    // 性别标签
    private JLabel lb2 = new JLabel("性别");
    // 表示性别的 JRadioButton 单选框
    private JRadioButton rb1 = new JRadioButton("男");
    private JRadioButton rb2 = new JRadioButton("女");
    // ButtonGroup 添加 JRadioButton,实现单选功能
    private ButtonGroup bg = new ButtonGroup();
    // 文本域组件
    private JTextArea area = new JTextArea();
    // 窗口 CENTER 部的 JScrollPane 面板，其中放置 area 文本域
    private JScrollPane pane = new JScrollPane(area);
    // Set 集合存放选中的兴趣
    private Set<String> hobbies = new HashSet<String>();
    // gender 选中的性别
    private String gender = "";
    // JCheckBox 复选框的事件监听器
    private ActionListener listener1 = new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            JCheckBox cb = (JCheckBox) e.getSource();
            // 选中的复选框把文本添加到 Set 集合中
            if (cb.isSelected()) {
                hobbies.add(cb.getText());
            //  反之从集合中移除
            } else {
                hobbies.remove(cb.getText());
            }
            print();
        }
    };
    // JRadioButton 单选框的事件监听器
    private ActionListener listener2 = new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            JRadioButton jb = (JRadioButton) e.getSource();
            gender = jb.getText();
            print();
        }
    };
    // 打印方法
    private void print() {
        // 清空文本域
        area.setText("");
        // 如果 Set 集合中有元素，打印兴趣
        if (hobbies.size() > 0)
```

```
            area.append("你的兴趣爱好有：");
        Iterator<String> it = hobbies.iterator();
        while (it.hasNext()) {
            area.append(it.next() + " ");
        }
        // 如果 gender 不为空字符串，打印性别
        if (!"".equals(gender))
            area.append("你的性别为： " + gender);
    }
    public Information() {
        //添加标签、单选和复选按钮
        panel.add(lb1);
        panel.add(cb1);
        panel.add(cb2);
        panel.add(cb3);
        panel.add(lb2);
        panel.add(rb1);
        panel.add(rb2);
        bg.add(rb1);
        bg.add(rb2);
        // 为单选和复选按钮添加事件监听器
        cb1.addActionListener(listener1);
        cb2.addActionListener(listener1);
        cb3.addActionListener(listener1);
        rb1.addActionListener(listener2);
        rb2.addActionListener(listener2);
        // 将 JPanel 面板和 JScrollPane 面板添加到 JFrame 容器中
        Container container = this.getContentPane();
        container.add(panel, BorderLayout.NORTH);
        container.add(pane, BorderLayout.CENTER);
        this.pack();
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setVisible(true);
    }
    public static void main(String[] args) {
        new Information();
    }
}
```

第 9 章　JDBC

一、填空题

1、 Java Database Connectivity

2、 java.sql.*

3、 classpath

4、 registerDriver()

5、 next()

二、判断题

1、对　2、错　3、对　4、错　5、错

三、选择题

1、A　2、C　3、A　4、D　5、AC

四、简答题

1、JDBC 是一套用于执行 SQL 语句的 Java API。应用程序可通过这套 API 连接到关系型数据库，并使用 SQL 语句来完成对数据库中数据的查询、新增、更新和删除等操作。

2、

（1）加载数据库驱动

（2）通过 DriverManager 获取数据库连接

（3）通过 Connection 对象获取 Statement 对象

（4）使用 Statement 执行 SQL 语句

（5）操作 ResultSet 结果集

（6）关闭连接，释放资源

3、

（1）代码的可读性和可维护性

（2）PreparedStatement 尽最大可能提高性能，因为预编译语句有可能被重复调用,所以语句在被 DB 的编译器编译后的执行代码被缓存下来,那么下次调用时只要是相同的预编译语句就不需要编译,只要将参数直接传入编译过的语句执行代码中（相当于一个函数）就会得到执行。

（3）极大地提高了安全性传。递给 PreparedStatement 对象的参数可以被强制进行类型转换，使开发人员可以确保在插入或查询数据时与底层的数据库格式匹配；在公共 Web 站点环境下，防止 Sq 注入问题。

五、编程题

1、参考答案

```
public static void main(String[] args) throws Exception {
        // int i = insert();
        // int i = delete();
        // int i = update();
        select();
        // System.out.println(i);
    }

    // 获取连接对象
    private static Connection getConn() {
        String driver = "com.mysql.jdbc.Driver";
        String url = "jdbc:mysql://localhost:3306/javatest";
        String username = "root";
        String password = "1234";
        Connection conn = null;
        try {
            Class.forName(driver); // classLoader,加载对应驱动
            conn = DriverManager.getConnection(url, username, password);
```

```
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return conn;
    }

    // 插入操作
    private static int insert() {
        Connection conn = getConn();
        int i = 0;
        String sql = "insert into tb_user(name,sex,email,birthday) values(?,?,?,?)";
        PreparedStatement pstmt;
        try {
            pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, "itcast");
            pstmt.setString(2, "男");
            pstmt.setString(3, "itcast@126.com");
            pstmt.setString(4, "2000-01-01");
            i = pstmt.executeUpdate();
            pstmt.close();
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return i;
    }

    // 删除操作
    private static int delete() {
        Connection conn = getConn();
        int i = 0;
        String sql = "delete from tb_user where name=?";
        PreparedStatement pstmt;
        try {
            pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, "itcast");
            i = pstmt.executeUpdate();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return i;
    }
```

```
        // 更新操作
        private static int update() {
            Connection conn = getConn();
            int i = 0;
            String sql = "update tb_user set name=? where name =?";
            PreparedStatement pstmt;
            try {
                pstmt = conn.prepareStatement(sql);
                pstmt.setString(1, "itcast");
                pstmt.setString(2, "abc");
                i = pstmt.executeUpdate();
                System.out.println("resutl: " + i);
                pstmt.close();
                conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
            return i;
        }

        // 查询操作
        private static void select() {
            Connection conn = getConn();
            String sql = "select * from tb_user";
            PreparedStatement pstmt;
            try {
                pstmt = conn.prepareStatement(sql);
                ResultSet rs = pstmt.executeQuery();
                while (rs.next()) {
                    int id = rs.getInt("id"); // 通过列名获取指定字段的值
                    String name = rs.getString("name");
                    String sex = rs.getString("sex");
                    String email = rs.getString("email");
                    Date birthday = rs.getDate("birthday");
                    System.out.println(id + "   |   " + name + "    |   " + sex + " |   "  +
email + "   |   " + birthday);
                }
                pstmt.close();
                conn.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
```

2、参考答案

```
import java.sql.*;
public class Test {
    public static void main(String[] args) throws Exception {
        transferAccounts(2, 1, 100);
    }
    public static void transferAccounts(int fromid,int toid,int transferMoney) throws
                                                                    Exception{
        Class.forName("com.mysql.jdbc.Driver");
        String url = "jdbc:mysql://localhost:3306/jdbc";
        Connection con = DriverManager.getConnection(url,"root", "root");
        int fromMoney = getMoney(fromid);
        // 转出账户
        String sql = "update tb_count set money=? where id =?";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setInt(1, fromMoney-transferMoney);
        ps.setInt(2, fromid);
        // 转入账户
        int toMoney = getMoney(toid);
        String sql2 = "update tb_count set money=? where id =?";
        PreparedStatement ps2 = con.prepareStatement(sql2);
        ps2.setInt(1, toMoney+transferMoney);
        ps2.setInt(2, toid);
        ps.executeUpdate();
        ps2.executeUpdate();
        ps.close();
        ps2.close();
        con.close();
    }
    // 获取当前账户 id 的账户余额
    public static int getMoney(int id) throws Exception{
        Class.forName("com.mysql.jdbc.Driver");
        String url = "jdbc:mysql://localhost:3306/jdbc";
        Connection con = DriverManager.getConnection(url,"root", "root");
        String sql = "select money from tb_count where id =?";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setInt(1, id);
        ResultSet resultSet = ps.executeQuery();
        int money = 0;
        while(resultSet.next()){
            money = resultSet.getInt(1);
        }
        ps.close();
        con.close();
```

```
            return money;
        }
    }
```

第 10 章 多线程

一、填空题

1、Thread、Runnable、Callable

2、synchronized、this

3、NEW（新建状态）、RUNNABLE（可运行状态）、BLOCKED（阻塞状态）、WAITING（等待状态）、TIMED_WAITING（定时等待状态）、TERMINATED（终止状态）

4、开启一个新线程、run()方法

5、setDaemon(true)、start()

二、判断题

1、错　2、对　3、对　4、错　5、对

三、选择题

1、AC　2、BC　3、ABC　4、C　5、ABCD

四、简答题

1、一种是继承 java.lang 包下的 Thread 类，覆写 Thread 类的 run()方法，在 run()方法中实现运行在线程上的代码。

```
new Thread() {
    public void run(){}
}.start();
```

另一种就是实现 java.lang.Runnable 接口，同样是在 run()方法中实现运行在线程上的代码。

```
class MyThread implements Runnable{
    public void run(){}
}
```

另一种就是实现 java.util.concurrent.Callable 接口，同样是在 call()方法中实现运行在线程上的代码。

```
class MyThread implements Callable<Object>{
    public Object call() throws Exception{}
}
```

2、 调用 sleep(long millis)方法，正在执行的线程主动让出 CPU 去执行其他线程，在 sleep(long millis)方法指定的时间过后，CPU 才会回到这个线程上继续往下执行，如果当前线程进入了同步锁，sleep(long millis)方法并不会释放锁，即使当前线程使用 sleep(long millis)方法让出了 CPU，但其他被同步锁挡住了的线程也无法得到执行。wait()在一个已经进入了同步锁的线程内进行调用，让当前线程暂时让出同步锁，以便其它正在等待此锁的线程可以得到同步锁并运行。当其它线程调用了 notify()或 notifyAll()方法后，调用 wait()方法的线程就会解除 wait 状态，当再次获得同步锁后，程序可以继续向下执行。

3、 单线程的程序都是从 main()方法入口开始执行到程序结束，整个过程只能顺序执行，如果程序在某个地方出现问题，那么整个程序就会崩溃，所以这就说明了单线程在某些方面的脆弱性和局限性。。

五、编程题

1、参考答案

```
public class Test01 {
    public static void main(String[] args) {
```

```
        Teacher t = new Teacher();
        new Thread(t, "陈老师").start();
        new Thread(t, "高老师").start();
        new Thread(t, "李老师").start();
    }
}
class Teacher implements Runnable {
    private int notes = 80;
    public void run() {
        while (true) {
            dispatchNotes(); // 调用售票方法
            if (notes <= 0) {
                break;
            }
        }
    }
    private synchronized void dispatchNotes() {
        if (notes > 0) {
            try {
                Thread.sleep(10); // 经过的线程休眠 10 毫秒
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            System.out.println(Thread.currentThread().getName() + "---发出的笔记"
                    + notes--);
        }
    }
}
```

2、参考答案

```
public class Accumulator extends Thread {
    private int stratNum;
    public static int sum;
    public Accumulator(int startNum) {
        this.stratNum = startNum;
    }
    public static synchronized void add(int num) {
        sum += num;
    }
    public void run() {
        int sum = 0;
        for (int i = 0; i < 10; i++) {
            sum += stratNum + i;
        }
        add(sum);
```

```
    }
    public static void main(String[] args) throws Exception {
        Thread[] threadList = new Thread[10];
        for (int i = 0; i < 10; i++) {
            threadList[i] = new Accumulator(10 * i + 1);
            threadList[i].start();
        }
        System.out.println("Sum is : " + sum);
    }
}
```

第 11 章　网络编程

一、填空题

1、面向连接、客户端、服务器端

2、2、0-65535

3、链路层、网络层、运输层、应用层

4、InetAddress

5、DatagramPacket、DatagramSocket

二、判断题

1、错　2、对　3、对　4、错　5、对

三、选择题

1、C　2、ABD　3.、A　4、B　5、C

四、简答题

1、在 Internet 中传输数据都需要遵守一定的规则，这种规则通常被称作网络通信协议。网络通信协议对数据传输格式、传输速率、传输步骤等作了统一规定，通信双方必须共同遵守这个规定才能完成数据的交互。到目前为止，网络通信协议已经有很多种，其中 TCP/IP 协议在世界范围内应用最为广泛。

2、UDP 协议是无连接通信协议，所谓的无连接就是指数据的发送端和接收端不建立逻辑连接。由于 UDP 协议消耗资源小，通信效率高，通常都会用于音频、视频和普通数据的传输。UDP 协议在传输数据时不能保证数据的完整性，因此在传输重要数据时不建议使用 UDP 协议。

TCP 协议是面向连接的通信协议，即在传输数据前先在发送端和接收端建立逻辑连接，然后再传输数据，它提供了两台计算机之间可靠无差错的数据传输。在 TCP 连接中必须要明确客户端与服务器端，由客户端向服务端发出连接请求，每次连接的创建都需要经过“三次握手”。

3、ServerSocket 类用于创建服务端程序，通过调用 ServerSocket 对象的 accept()方法，接收来自客户端的请求。

Socket 类用于创建客户端程序，当客户端和服务端的两个 Socket 建立了专线连接后，连接的一端既能向另一端连续写入字节，也能从另一端读取字节。Socket 类中定义了 getInputStream()方法返回 Socket 的输入流对象，定义了 getOutputStream()方法返回 Socket 的输出流对象。只要连接的一端向该输出流对象写入了数据，连接的另一端就能从其输入流对象中读取到。

五、编程题

1、参考答案

接收端：

```
import java.net.*;
```

```
public class Receiver {
    public static void main(String[] args) throws Exception {
        byte[] buf = new byte[1024];
        DatagramSocket ds = new DatagramSocket(8001);
        DatagramPacket dp = new DatagramPacket(buf, 1024);
        ds.receive(dp);
        String str = new String(dp.getData(),0, dp.getLength());
        System.out.println(str);
        ds.close();
    }
}
```

发送端

```
import java.net.*;
public class Send {
    public static void main(String[] args) throws Exception {
        DatagramSocket ds = new DatagramSocket(3000);
        String str = "hello world";
        DatagramPacket dp = new DatagramPacket(str.getBytes(), str.length(),
                InetAddress.getByName("localhost"), 8001);
        ds.send(dp);
        ds.close();
    }
}
```

2、参考答案

服务端

```
import java.io.*;
import java.net.*;
public class Server {
    public static void main(String[] args) throws Exception {
        new TCPServer().listen();
    }
}
class TCPServer {
    public void listen() throws Exception {
        ServerSocket serverSocket = new ServerSocket(8002);
        Socket client = serverSocket.accept();
        OutputStream os = client.getOutputStream();
        os.write(("hello world").getBytes());
        Thread.sleep(5000);
        os.close();
        client.close();
    }
}
```

客户端

```
import java.io.*;
import java.net.*;
public class Client {
    public static void main(String[] args) throws Exception {
        new TCPClient().connect();
    }
}
class TCPClient {
    public void connect() throws Exception {
        Socket client = new Socket(InetAddress.getLocalHost(), 8002);
        InputStream is = client.getInputStream();
        byte[] buf = new byte[1024];
        int len = is.read(buf);
        System.out.println(new String(buf, 0, len));
        client.close();
    }
}
```