《Java 基础入门》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、选择题

1、ABCD　　2、C　　3、D　　4、B　　5、B

三、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。

四、编程题

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

第2章 Java 编程基础

一、填空题

1、 class

2、 true 和 false

3、 单行注释、多行注释、文档注释

4、 基本数据类型、引用数据类型

5、 1、2、4、8

6、 & && | ||

7、 0

8、 5

9、 34

10、 56

二、判断题

1、错 2、对 3、错 4、对 5、错

三、选择题

1、AD 2、AD 3、C 4、ABCD 5、C 6 、A 7、AC 8、A 9、B 10、A

四、程序分析题

1、编译不通过。int 值 4 和 b 相加时，由于变量 b 的类型为 byte，取值范围没有 int 类型大，存不下 int 类型的值，因此编译不通过。

2、编译不通过。这是因为 y 是在最里层的代码块中定义的一个变量，只有在那个代码块中才可使用，在使用 y = x;语句时已经超过了 y 变量的作用域，所以编译无法通过。

3、打印结果为：3。

4、打印结果为：

9

8

7

五、问答题

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致。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。

六、编程题

1、参考答案

```
public class Test01 {
    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 Test02 {
    public static void main(String args[]) {
        int y = function(0);
```

```
            System.out.println(y);
        }
        public static int function(int x) {
            int y;
            if (x > 0) {
                y = x + 3;
            } else if (x == 0) {
                y = 0;
            } else {
                y = x * x - 1;
            }
            return y;
        }
    }
```

3、参考答案

```
public class Test03 {
    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、new

3、成员变量、局部变量

4、类、类

5、this

6、finalize()

7、静态变量

8、内部类

9、javadoc

10、private

二、判断题

1、对　2、对　3、错　4、对　5、错

三、选择题

1、B　2、D　3、B　4、ABC　5、ABCD　6、ACD　7、ABCD　8、ABCD　9、D　10、D

四、程序分析题

1、程序不能编译通过，因为在类 A 中的成员变量 secret 用 private 修饰，所以在类 Test1 中无法访问。

2、程序不能编译通过，因为在静态方法 method()中不能访问非静态成员变量 x。

3、程序能够编译通过，运行的结果为“inner”。

五、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、单例模式可以保证在整个程序运行期间针对该类只存在一个实例对象。

六、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

```
class Father {
    private String name = "zhangjun";
    class Child {
        public void introFather() {
            System.out.println("My Father's name is " + name);
        }
    }
}
public class Test02 {
    public static void main(String[] args) {
        Father.Child child = new Father().new Child();
        child.introFather();
    }
}
```

第4章 面向对象（下）

一、填空题

1、继承
2、方法，抽象类
3、import
4、子类、父类、基类
5、Exception
6、final
7、super
8、 Object
9、try、catch
10、jar –cvf，java –jar

二、判断题

1、错　2、对　3、错　4、对　5、对

三、选择题

1、B　2、C　3、ABC　4、 ABCD　5、C　6、AC　7、C　8、D　9、A　10、B

四、程序分析题

1、程序编译能通过，这是因为 int x = 2 / 0; System.out.println(x);这两条语句使用了 try 块，捕获了程序因为除以 0 而产生的异常情况，之后程序会继续向下执行，输出“进入 catch 代码块”，“进入 finally 代码块”。

2、程序编译不通过，这是因为在程序中使用了 final 关键字修饰 Animal 类，使得 Animal 类不能被继承。shout()方法中同样使用了 final 关键字，使得该方法不能被重写。

3、程序编译能通过，输出结果为“动物叫!”和“汪汪……”，因为在程序中调用shout()方法时，首先会通过super.shout()调用父类的方法说出“动物叫!”之后再输出“汪汪……”

4、程序编译不通过，因为接口中定义的方法不能有方法体，所以定义的 eat()方法是错误的。接口中的方法必须在子类中全部实现，由于 run()方法在子类中并没有重新实现，所以这也是错误的。

五、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和抽象方法。由于接口的特殊性，在定义时需要使用 interface 关键字。

六、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));
    }
}
```

3、参考答案

```
class NoThisSongException extends Exception{
    public NoThisSongException(){
        super();
    }
    public NoThisSongException(String message){
        super(message);
    }
}
class Player{
    public void play(int index)throws NoThisSongException{
        if(index>10){
            throw new NoThisSongException("您播放的歌曲不存在");
        }
        System.out.println("正在播放歌曲");
    }
}
public class Test03 {
    public static void main(String[] args) {
        Player player = new Player();
        try {
```

```
                player.play(13);
            } catch (NoThisSongException e) {
                System.out.println("异常信息为： "+e.getMessage());
            }
        }
    }
```

第5章 多线程

一、填空题

1、 线程、通信

2、 Thread、Runnable

3、 就绪

4、 synchronized、对象、this

5、 进程

6、新建状态(New)、就绪状态(Runnable)、运行状态(Running)、阻塞状态(Blocked)、死亡状态(Terminated)

7、 10、1

8、 开启一个新线程、run()方法

9、 wait()、notify()、notifyAll()

10、setDaemon(true)、start()

二、判断题

1、错　2、对　3、对　4、错　5、错

三、选择题

1、B　2、AC　3、ABC　4、BC　5、ABD　6、ABC　7、C　8、D　9、AB　10、ABCD

四、程序分析题

1、程序不能编译通过，因为 RunHandler 类没有实现 Runnable 接口，因此 RunHandler 的实例对象不能作为参数传递给 Thread 的构造方法。

2、程序不能编译通过，因为 Thread 的子类 A 重写的 run()方法的访问级别不能低于父类 run()方法的。访问级别

3、程序不能编译通过，因为同步方法中调用 wait()方法的对象必须为同步锁对象。

4、t.start();

五、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。

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

2、调用 sleep()方法，正在执行的线程主动让出 CPU 去执行其他线程，在 sleep()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()方法并不会释放锁，即使当前线程使用 sleep()方法让出了 CPU，但其它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()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。

六、编程题

1、参考答案

```
public class MyThread extends Thread{
    public MyThread(String name) {
        super(name);
    }
    public void run() {
       System.out.println(this.getName());
    }
    public static void main(String[] args) {
        new MyThread("Thread1").start();
        new MyThread("Thread2").start();
    }
}
```

2、参考答案

```
public class MyRunnable implements Runnable {
    public void run() {
        for (int i = 0; i < 50; i++) {
            System.out.println("new");
        }
    }
    public static void main(String[] args) {
        new Thread(new MyRunnable()).start();
        for (int i = 0; i < 100; i++) {
            System.out.println("main");
        }
    }
}
```

3、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--);
        }
    }
}
```

4、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();
        }
        for (int i = 0; i < 10; i++) {
            threadList[i].join();
        }
```

```
            System.out.println("Sum is : " + sum);
        }
    }
```

第6章 JavaAPI

一、填空题

1、 String、StringBuffer
2、 Date、Calendar、DateFormat
3、 getRuntime()
4、 sqrt()
5、 DateFormat
6、 π、e
7、 Random、java.util
8、 length()
9、 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、D　4、C　5、C　6、B　7、C　8、A　9、A　10、B

四、程序分析题

1、程序编译能通过，输出结果如下

```
5
7.0
-8.0
-5
8.1
-6.1
```

2、程序编译能通过，输出结果如下

```
str.length():15
str.charAt(0):d
lastIndexOf(m):10
substring(2,4):fe
indexOf(g):5
```

五、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。

六、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.text.DateFormat;
import java.util.Calendar;
import java.util.Date;
public class Test02 {
    public static void main(String[] args) {
        Calendar calendar = Calendar.getInstance();
        calendar.add(Calendar.DATE, 100);
        Date date = calendar.getTime();
        DateFormat format = DateFormat.getDateInstance(DateFormat.FULL);
        String string = format.format(date);
        System.out.println(string);
    }
}
```

3、 参考答案

```
import java.util.Random;
public class Test03 {
        public static void main(String[] args) {
              Random rand = new Random();
              int[] num = new int[5];
              for (int i = 0; i < num.length; i++) {
                  num[i] = 20 + rand.nextInt(31);
                  System.out.println(num[i]);
              }
        }
```

```
}
```

第7章 集合类

一、填空题

1、集合

2、Comparator

3、有序、可重复，无序、不可重复

4、hashNext()、next()

5、Collection、Map

6、键、值

7、ListIterator

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

9、put()、get()

10、Collections、Arrays

二、判断题

1、错　2、对　3、对　4、错　5、对

三、选择题

1、BC　2、A　3、D　4、ABD　5、C　6、AB　7、D　8、AB　9、ABC　10、B

四、程序分析题

1、程序可以编译通过，输出结果是“a、b、c”，因为 TreeSet 集合不允许存放重复元素，第 2 次增加的元素 c 会覆盖之前存入的元素 c，所以输出结果是“a、b、c”，而不是“a、b、c、c”。

2、程序不可以编译通过，这是由于向 ArrayList 集合中存入元素时，集合并不能记住元素的类型，因此在取出元素时，只能使用 Object 类型，而不能使用 String 类型。

3、程序可以编译通过，但是什么也没有打印。使用 ListIterator 进行从后向前的遍历集合，可以使用以下两种方法，一是使用 listIterator(int index)方法将索引 index 的值设置为集合元素的数目，也就是 ListIterator it = list.listIterator(3);，二是将程序先从前向后遍历，然后再从后向前遍历。

4、程序编译不通过，由于 Map 集合在遍历的过程中不能使用集合对象本身删除元素，这会导致并发修改异常，若想删除集合中的元素，可以使用 Iterator 的 remove()方法。

五、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。集合框架中包含 3 个接口，分别是 List、Set、Map。

2、List 的特点是元素有序、元素可重复。List 接口的主要实现类有 ArrayList 和 LinkedList。Set 的特点是元素无序、元素不可重复。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。

六、编程题

1、参考答案

```
import java.util.*;
public class Test01 {
    public static void main(String[] args) {
        ArrayList list = new ArrayList();
```

```
        for(int i = 0; i < 10; i++) {
            list.add("A"+i);
        }
        Iterator it = list.iterator();
        while(it.hasNext()) {
            Object obj = it.next();
            System.out.println(obj);
        }
    }
}
```

2、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 other = (Person) obj;
```

```
            return other.name.equals(this.name);
        }
    }
```

3、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8章 IO（输入输出）

一、填空题

1、 字节流、字符流

2、 File、java.io

3、 SequenceInputStream

4、 RandomAccessFile

5、 管道流

6、 LineNumberReader、BufferedReader、setLineNumber(int)、getLineNumber()

7、 字节流、字符流

8、 PrintStream、setOut(PrintStream out)

9、 Unicode

10、newLine()

二、判断题

1、错　2、对　 3、对　4、对　5、错

三、选择题

1、AB　2、C　3、C　4、ABCD　5、AB　6、ACD　7、A　8、A　9、D　10、A

四、程序填空题

1、InputStreamReader、br.readLine()、运行结果为 2

2、FileInputStream(file1)、FileOutputStream、fis.available()

五、问答题

1、Java 程序通过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、管道流分为管道输入流（PipedInputStream）和管道输出流（PipedOutputStream），通常由一个 PipedInputStream 实例对象和一个 PipedOutputStream 实例对象相互连接而实现管道通信，PipedOutputStream 向管道写入数据，PipedInputStream 从管道中读取 PipedOutputStream 写入的数据，管道流主要用来完成线程之间的通信。

六、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9章 GUI（图形用户界面）

一、填空题

1、 GUI、java.awt、javax.swing
2、 适配器
3、 Graphics
4、 事件监听器
5、 JComponent
6、 窗体事件、键盘事件、鼠标事件、动作事件
7、 WindowListener、windowClosing(WindowEvent e)
8、 模态对话框、非模态对话框
9、 setLayout(null)
10、JMenuBar、JMenu、JMenuItem

二、判断题

1、对　2、错　3、对　4、错　5、错

三、选择题

1、D　2、D　3、ABD　4、A　5、D　6、ABCD　7、ABD　8、ABC　9、D　10、C

四、程序分析题

1、参考答案

```
extends
Stdno = new JTextField();
Name = new JTexxtField();
```

2、参考答案

```
card.next(cp);
cp.add(jbt);
```

五、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、参考答案

AWT 是 Abstract Window ToolKit (抽象窗口工具包)的缩写，这个工具包提供了一套与本地图形界面进行交互的接口。AWT 中的图形函数与操作系统所提供的图形函数之间有着一一对应的关系，当我们利用 AWT 来构件图形用户界面的时候，我们实际上是在利用操作系统所提供的图形库。由于不同操作系统的图形库所提供的功能是不一样的，在一个平台上存在的功能在另外一个平台上则可能不存在。为了实现 Java 语言所宣称的"一次编译，到处运行"的概念，AWT 不得不通过牺牲功能来实现其平台无关性，也就是说，AWT 所提供的图形功能是各种通用型操作系统所提供的图形功能的交集。由于 AWT 是依靠本地方法来实现其功能的，我们通常把 AWT 控件称为重量级控件。

Swing 是在 AWT 的基础上构建的一套新的图形界面系统，它提供了 AWT 所能够提供的所有功能，并且用纯粹的 Java 代码对 AWT 的功能进行了大幅度的扩充。由于在 Swing 中没有使用本地方法来实现图形功能，我们通常把 Swing 控件称为轻量级控件。

AWT 和 Swing 之间的基本区别：AWT 是基于本地方法的 C/C++程序，其运行速度比较快；Swing 是基于 AWT 的 Java 程序，其运行速度比较慢。对于一个嵌入式应用来说，目标平台的硬件资源往往非常有限，而应用程序的运行速度又是项目中至关重要的因素。在这种矛盾的情况下，简单而高效的 AWT 当然成了嵌入式 Java 的第一选择。而在普通的基于 PC 或者是工作站的标准 Java 应用中，硬件资源对应用程序所造成的限制往往不是项目中的关键因素，所以在标准版的 Java 中则提倡使用 Swing，也就是通过牺牲速度来实现应用程序的功能。

六、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() == e.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("爱好");
    // 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();
        }
    }
```

3、参考答案

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class MyMenu extends JFrame implements ActionListener {
    JLabel label = new JLabel("请选择菜单", JLabel.CENTER);
    JMenuItem aaMenuItem, baMenuItem;
    MyMenu() {
        JMenuBar menuBar = new JMenuBar();
        JMenu aMenu = new JMenu("菜单 A");
        JMenu bMenu = new JMenu("菜单 B");
        JMenuItem aaMenuItem = new JMenuItem("菜单项 AA");
        JMenuItem abMenuItem = new JMenuItem("菜单项 AB");
        JMenuItem baMenuItem = new JMenuItem("菜单项 BA");
        menuBar.add(aMenu);
        menuBar.add(bMenu);
        aMenu.add(aaMenuItem);
        aMenu.addSeparator();
        aMenu.add(abMenuItem);
        bMenu.add(baMenuItem);
        aaMenuItem.addActionListener(this);
        abMenuItem.addActionListener(this);
        baMenuItem.addActionListener(this);
        setJMenuBar(menuBar);
        getContentPane().add(label, BorderLayout.CENTER);
    }
    public void actionPerformed(ActionEvent e) {
        JMenuItem source = (JMenuItem) (e.getSource());
        label.setText("选择了菜单:" + source.getText());
        label.setHorizontalAlignment(JLabel.CENTER);
    }
```

```
    public static void main(String args[]) {
        JFrame frame = new MyMenu();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 200);
        frame.setVisible(true);
    }
}
```

第10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.A　3、ABD　4、B　5.、A　6、D　7、B　8、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

```
import java.net.InetAddress;
public class Test01 {
    public static void main(String[] args) throws Exception {
        InetAddress localAddress = InetAddress.getLocalHost();
        InetAddress remoteAddress = InetAddress.getByName("www.oracle.com");
```

```
            System.out.println("本机的 IP 地址：" + localAddress.getHostAddress());
            System.out.println("本地的主机名：" + localAddress.getHostName());
            System.out.println("甲骨文的 IP 地址：" + remoteAddress.getHostAddress());

        }
    }
```

2、参考答案

接收端：

```
import java.net.*;
public class Test02 {
    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 Test03 {
    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();
    }
}
```

3、参考答案

服务端

```
import java.io.*;
import java.net.*;
public class Test04 {
    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 Test05 {
        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();
        }
    }
```