

《Java 程序设计》实验报告 13

学生姓名：李季鸿

班级：2021 级计本（3）班

学号：20213002624

实验地点：9 教 304

指导教师：张春元

实验日期：2023-05-29

共 2 学时

实验环境：Win10+JDK1.8+ IntelliJ IDEA 2022.1.1

1. 实验目的

学习多线程编程和网络编程。

2. 实验内容

（1）用集成化开发工具完成实验教材 P121 实验 4 内容。

（2）用集成化开发工具完成实验教材 P128 实验 2 内容。

3. 实验过程

报告撰写具体要求：截屏显示或直接写出实验 1 至实验 2 的源码和运行结果。

实验内容（1）：

MainClass.java:

```
package _13_.shiyuan1;
```

```
import javax.swing.*;
```

```
import java.awt.*;
```

```
/**
```

```
 * \* Created with IntelliJ IDEA.
```

```
 * \* @ProjectName: java_study_codes
```

```
 * \* @FileName: MainClass
```

```
 * \* @author: li-jihong
```

```
 * \* Date: 2023-05-29 16:23
```

```
 */
```

```
public class MainClass {
```

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

```
        Sky sky = new Sky();           //构造了一个天空(标签对象)
```

```
        JFrame frame = new JFrame();    //构造了一个框架(窗体)
```

```
        frame.add(sky);                 //将天空(标签)置于框架(窗体)里
```

```
        frame.setTitle("月亮绕地球转");
```

```
        frame.setSize(500, 500);
```

```
        frame.setVisible(true);
```

```
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
        frame.getContentPane().setBackground(Color.white);
```

```
    }
```

```
}
```

Earth.java:

```

package _13_.shiyuan1;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * \* Created with IntelliJ IDEA.
 * \* @ProjectName: java_study_codes
 * \* @FileName: Earth
 * \* @author: li-jihong
 * \* Date: 2023-05-29 16:25
 */
public class Earth extends JLabel implements ActionListener {
    JLabel moon; //显示月亮的外观
    Timer timer;
    double pointX[] = new double[360],
           pointY[] = new double[360];
    int w = 200, h = 200, i = 0;

    Earth() {
        setLayout(new FlowLayout());
        setPreferredSize(new Dimension(w, h));
        timer = new Timer(20, this);
        //【代码 1】创建 timer，振铃间隔是 20 毫秒，当前 Earth 对象为其监视器
        setIcon(new ImageIcon("image/地球.png"));
        setHorizontalAlignment(SwingConstants.CENTER);
        moon = new JLabel(new ImageIcon("image/月亮.png"), SwingConstants.CENTER);
        add(moon);
        moon.setPreferredSize(new Dimension(60, 60));
        pointX[0] = 0;
        pointY[0] = h / 2;
        double angle = 1 * Math.PI / 180; //刻度为 1 度
        for (int i = 0; i < 359; i++) { //计算出数组中各个元素的值
            pointX[i + 1] = pointX[i] * Math.cos(angle) - Math.sin(angle) * pointY[i];
            pointY[i + 1] = pointY[i] * Math.cos(angle) + Math.sin(angle) * pointX[i];
        }

        for (int i = 0; i < 360; i++) {
            pointX[i] = 0.8 * pointX[i] + w / 2; //坐标缩放，平移
            pointY[i] = 0.8 * pointY[i] + h / 2;
        }
    }
}

```

```

        timer.start();
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        i = (i + 1) % 360;
        moon.setLocation((int) pointX[i] - 30, (int) pointY[i] - 30);
    }
}

Sky.java:
package _13_.shiyuan1;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * \* Created with IntelliJ IDEA.
 * \* @ProjectName: java_study_codes
 * \* @FileName: Sky
 * \* @author: li-jihong
 * \* Date: 2023-05-29 16:26
 */
public class Sky extends JLabel implements ActionListener {
    Earth earth;
    Timer timer;
    double pointX[] = new double[360],
           pointY[] = new double[360];
    int w = 400, h = 400, i = 0;

    Sky() {
        setLayout(new FlowLayout());
        timer = new Timer(100, this);
        // 【代码 2】创建 timer，振铃间隔是 100 毫秒，当前 Sky 对象为其监视器
        setPreferredSize(new Dimension(w, h));
        earth = new Earth();
        add(earth);
        earth.setPreferredSize(new Dimension(200, 200));
        pointX[0] = 0;
        pointY[0] = h / 2;
        double angle = 1 * Math.PI / 180; //刻度为 1 度
        for (int i = 0; i < 359; i++) { //计算出数组中各个元素的值

```

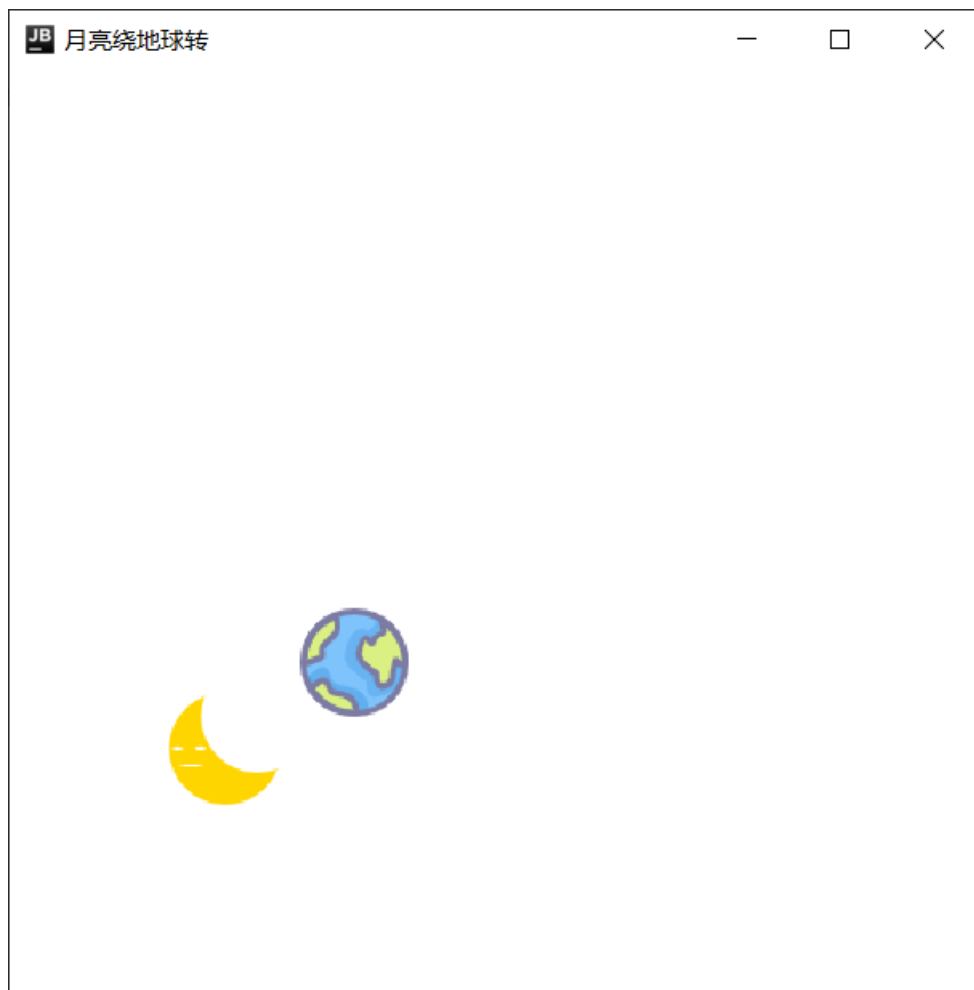
```

        pointX[i + 1] = pointX[i] * Math.cos(angle) - Math.sin(angle) * pointY[i];
        pointY[i + 1] = pointY[i] * Math.cos(angle) + Math.sin(angle) * pointX[i];
    }

    for (int i = 0; i < 360; i++) {
        pointX[i] = 0.5 * pointX[i] + w / 2;    //坐标缩放, 平移
        pointY[i] = 0.5 * pointY[i] + h / 2;
    }
    timer.start();
}

@Override
public void actionPerformed(ActionEvent e) {
    i = (i + 1) % 360;
    earth.setLocation((int) pointX[i] - 100, (int) pointY[i] - 100);
}
}

```



实验内容 (2):

ClientItem.java:

```
package _13_.shiyuan2;
```

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.InetAddress;
import java.net.InetSocketAddress;
import java.net.Socket;
import java.util.Scanner;
```

```
/**
```

```
 * \* Created with IntelliJ IDEA.
 * \* @ProjectName: java_study_codes
 * \* @FileName: ClientItem
 * \* @author: li-jihong
 * \* Date: 2023-05-29 16:51
 */
```

```
public class ClientItem {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Socket clientSocket = null;
        DataInputStream inData = null;
        DataOutputStream outData = null;
        Thread thread;
        Read read = null;
        try {
            clientSocket = new Socket();
            read = new Read();
            thread = new Thread(read);
            System.out.print("输入服务器的 IP:");
            String IP = scanner.nextLine();
            System.out.print("输入端口号:");
            int port = scanner.nextInt();
            String enter = scanner.nextLine();
            if (clientSocket.isConnected()) {
            } else {
                InetAddress address = InetAddress.getByName(IP);
                InetSocketAddress socketAddress = new InetSocketAddress(address, port);
                clientSocket.connect(socketAddress);
                InputStream in = clientSocket.getInputStream();
            }
        }
    }
}
```

```

        OutputStream out = clientSocket.getOutputStream();
        inData = new DataInputStream(in);
        outData = new DataOutputStream(out);
        read.setDataInputStream(inData);
        read.setDataOutputStream(outData);
        thread.start();
    }
} catch (Exception e) {
    System.out.println("服务器已断开" + e);
}
}
}

```

```

class Read implements Runnable {
    Scanner scanner = new Scanner(System.in);
    DataInputStream in;
    DataOutputStream out;

    public void setDataInputStream(DataInputStream in) {
        this.in = in;
    }

    public void setDataOutputStream(DataOutputStream out) {
        this.out = out;
    }

    public void run() {
        System.out.println("输入账单:");
        String content = scanner.nextLine();
        try {
            out.writeUTF("账单" + content);
            String str = in.readUTF();
            System.out.println(str);
            str = in.readUTF();
            System.out.println(str);
            str = in.readUTF();
            System.out.println(str);
        } catch (Exception e) {
        }
    }
}

```

ServerItem.java:

```
package _13_.shiyan2;
```

```

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.InputMismatchException;
import java.util.Scanner;

/**
 * \* Created with IntelliJ IDEA.
 * \* @ProjectName: java_study_codes
 * \* @FileName: ServerItem
 * \* @author: li-jihong
 * \* Date: 2023-05-29 16:51
 */
public class ServerItem {
    public static void main(String args[]) {
        ServerSocket server=null;
        ServerThread thread;
        Socket you=null;
        while(true) {
            try {server=new ServerSocket(4331);
            }
            catch(IOException e1) {
                System.out.println("正在监听");
            }
            try {System.out.println("正在等待客户");
                you=server.accept();
                System.out.println("客户的地址: "+you.getInetAddress());
            }
            catch(IOException e) {
                System.out.println(""+e);
            }
            if(you!=null) {
                new ServerThread(you).start();
            }
        }
    }
}

class ServerThread extends Thread{
    Socket socket;
    DataInputStream in=null;

```

```

DataOutputStream out=null;
ServerThread(Socket t){
    socket=t;
    try {out=new DataOutputStream(socket.getOutputStream());
        in=new DataInputStream(socket.getInputStream());
    }
    catch(IOException e) {}
}
public void run() {
    try {
        String item=in.readUTF();
        Scanner scanner=new Scanner(item);
        scanner.useDelimiter("[^0123456789.]+");
        if(item.startsWith("账单")) {
            double sum=0;
            while(scanner.hasNext()) {
                try {double price =scanner.nextDouble();
                    sum=sum+price;
                    System.out.println(price);
                }
                catch(InputMismatchException exp) {
                    String t=scanner.next();
                }
            }
            out.writeUTF("您的账单: ");
            out.writeUTF(item);
            out.writeUTF("总额: "+sum+"元");
        }
    }
    catch(Exception exp) {}
}
}

```




```
运行 ServerItem x ClientItem x
.m2\repository\org\jetbrains\annotations\24.0.0\annotations-24.0.0.jar
.28\mysql-connector-java-8.0.28.jar _13_.shiyang2.ServerItem
正在等待客户
客户的地址: /192.168.141.86
正在监听
正在等待客户
2189.0
112.9
569.0
832.0
|
```

服务器端



```
运行 ServerItem x ClientItem x
.28\mysql-connector-java-8.0.28.jar _13_.shiyang2.ClientItem
输入服务器的IP:192.168.141.86
输入端口号:4331
输入账单:
房租: 2189元 水费: 112.9元 电费: 569元 物业费: 832元
您的账单:
账单房租: 2189元 水费: 112.9元 电费: 569元 物业费: 832元
总额: 3702.9元
进程已结束,退出代码0
|
```

客户端

课后练习:

ClientItem.java:

import java.io.*;

import java.net.*;

import java.util.*;

public class ClientItem {

public static void main(String args[]) {

Scanner scanner = new Scanner(System.in);

Socket clientSocket=null;

```

DataInputStream inData=null;
DataOutputStream outData=null;
Thread thread ;
Read read=null;
try{ clientSocket=new Socket();
    read = new Read();
    thread = new Thread(read); //负责读取信息的线程
    System.out.print("输入服务器的 IP:");
    String IP = scanner.nextLine();
    System.out.print("输入端口号:");
    int port = scanner.nextInt();
    String enter=scanner.nextLine(); //消耗回车
    if(clientSocket.isConnected()){
    }
    else{
        InetAddress address=InetAddress.getByName(IP);
        InetSocketAddress socketAddress=new InetSocketAddress(address,port);
        clientSocket.connect(socketAddress);
        InputStream in=clientSocket.getInputStream(); //clientSocket 调用 getInputStream()返回
        该套接字的输入流
        OutputStream out=clientSocket.getOutputStream(); //clientSocket 调用 getOutputStream()
        返回该套接字的输出流
        inData =new DataInputStream(in);
        outData = new DataOutputStream(out);
        read.setDataInputStream(inData);
        read.setDataOutputStream(outData);
        thread.start(); //启动负责读信息的线程
    }
}
catch(Exception e) {
    System.out.println("服务器已断开"+e);
}
}
}

class Read implements Runnable {
    Scanner scanner = new Scanner(System.in);
    DataInputStream in;
    DataOutputStream out;
    public void setDataInputStream(DataInputStream in) {
        this.in = in;
    }
    public void setDataOutputStream(DataOutputStream out) {
        this.out = out;
    }
}

```

```

public void run() {
    System.out.println("输入账单:");
    String content = scanner.nextLine();
    try{ out.writeUTF(content);
        String str = in.readUTF();
        System.out.println(str);
        str = in.readUTF();
        System.out.println(str);
        str = in.readUTF();
        System.out.println(str);
    }
    catch(Exception e) {}
}
}

```

ServerItem.java

```

import java.io.*;
import java.net.*;
import java.util.*;
public class ServerItem {
    public static void main(String args[]) {
        ServerSocket server=null;
        ServerThread thread;
        Socket you=null;
        while(true) {
            try{ server=new ServerSocket(4331); //创建在端口 4331 上负责监听的 ServerSocket 对
象
            }
            catch(IOException e1) {
                System.out.println("正在监听");
            }
            try{ System.out.println("正在等待客户");
                you=server.accept(); // server 调用 accept()返回和客户端相连接的 Socket 对象
                System.out.println("客户的地址:"+you.getInetAddress());
            }
            catch (IOException e) {
                System.out.println(""+e);
            }
            if(you!=null) {
                new ServerThread(you).start();
            }
        }
    }
}

```

```

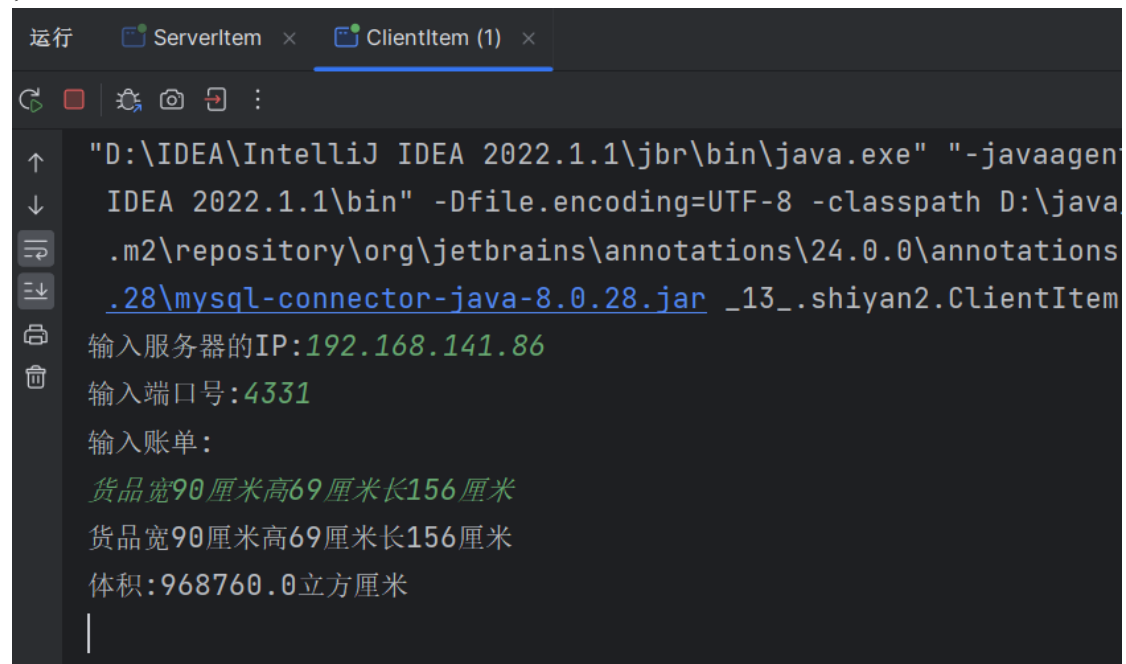
    }
    class ServerThread extends Thread {
        Socket socket;
        DataInputStream in=null;
        DataOutputStream out=null;
        ServerThread(Socket t) {
            socket=t;
            try { out=new DataOutputStream(socket.getOutputStream());
                in=new DataInputStream(socket.getInputStream());
            }
            catch (IOException e) {}
        }
        public void run() {
            try{
                String item = in.readUTF();
                Scanner scanner = new Scanner(item);
                scanner.useDelimiter("[^0123456789.]+");
                if(item.startsWith("账单")) {
                    double sum=0;
                    while(scanner.hasNext()){
                        try{ double price = scanner.nextDouble();
                            sum = sum+price;
                            System.out.println(price);
                        }
                        catch(InputMismatchException exp){
                            String t = scanner.next();
                        }
                    }
                    out.writeUTF("您的账单:");
                    out.writeUTF(item);
                    out.writeUTF("总额:"+sum+"元");
                }
                else if(item.startsWith("货品")) {
                    double volumn=1;
                    while(scanner.hasNext()){
                        try{ double a = scanner.nextDouble();
                            volumn *= a;
                            System.out.println(a);
                        }
                        catch(InputMismatchException exp){
                            String t = scanner.next();
                        }
                    }
                }
            }
        }
    }

```

```

        out.writeUTF(item);
        out.writeUTF("体积:"+volumn+"立方厘米");
    }
}
catch(Exception exp){}
}
}

```



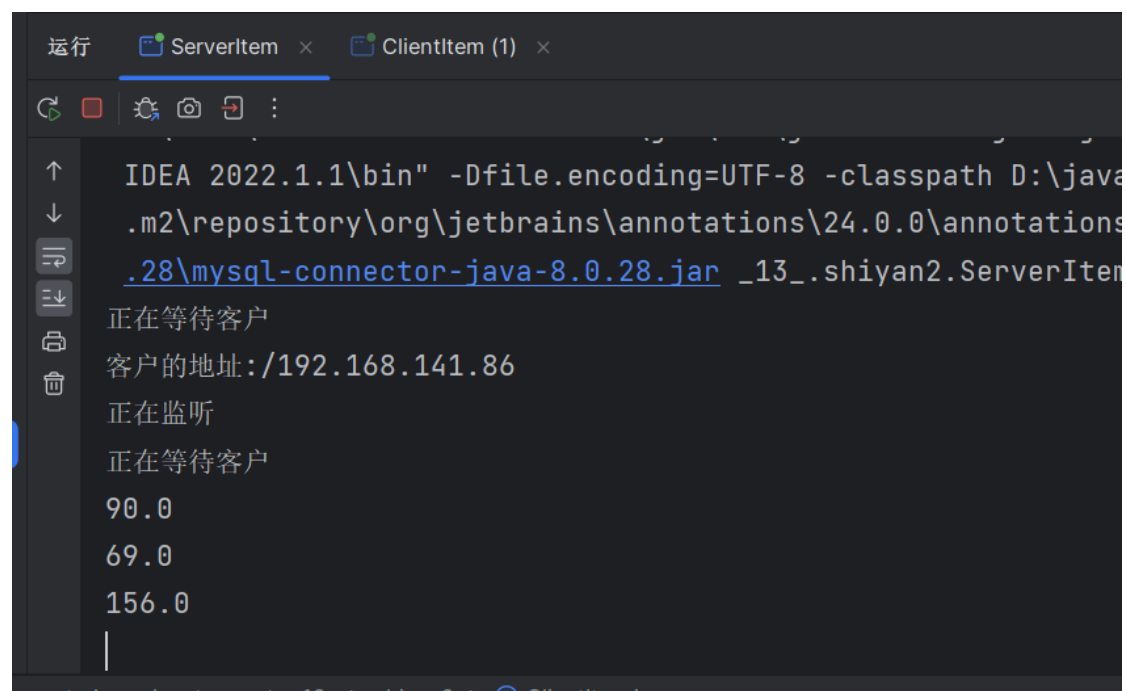
The screenshot shows the IntelliJ IDEA console for the ClientItem (1) class. The output displays the command line used to run the application, followed by user input for server IP, port, and item dimensions, and the calculated volume.

```

"D:\IDEA\IntelliJ IDEA 2022.1.1\jbr\bin\java.exe" "-javaagent
IDEA 2022.1.1\bin" -Dfile.encoding=UTF-8 -classpath D:\java
.m2\repository\org\jetbrains\annotations\24.0.0\annotations
.28\mysql-connector-java-8.0.28.jar _13_.shiyan2.ClientItem
输入服务器的IP:192.168.141.86
输入端口号:4331
输入账单:
货品宽90厘米高69厘米长156厘米
货品宽90厘米高69厘米长156厘米
体积:968760.0立方厘米

```

客户端



The screenshot shows the IntelliJ IDEA console for the ServerItem class. The output displays the command line used to run the application, followed by the server's status and the received dimensions from the client.

```

IDEA 2022.1.1\bin" -Dfile.encoding=UTF-8 -classpath D:\java
.m2\repository\org\jetbrains\annotations\24.0.0\annotations
.28\mysql-connector-java-8.0.28.jar _13_.shiyan2.ServerItem
正在等待客户
客户的地址:/192.168.141.86
正在监听
正在等待客户
90.0
69.0
156.0

```

服务器端

4. 实验总结

写出实验中的心得体会（对第 12 章理论课重点简述）。

一、线程与进程的区别

- 1) 进程是程序的一次运行，是操作系统进行资源分配和调度的一个独立单位。因而进程既是拥有资源的基本单位，同时又是可独立调度分配的基本单位。
- 2) 由于进程是资源的拥有者，其创建、切换和撤销需要花费较大的时空开销。
- 3) 当一个程序中有多个任务需要同时执行时，需要创建多个进程执行。

二、任务

在 Java 中，每个任务都是 `Runnable` 接口的一个实例，也称为可运行对象。一个任务必须在线程中执行。

- 1) 任务类的定义 `class 任务类名 implements Runnable{public void run(){...告诉线程如何完成此任务}}`
- 2) 创建一个任务对象 `任务类名 任务对象名=new 任务类名(...);`

三、线程

在 Java 中，线程可通过 `Thread` 类进行定义。

- 1) 线程的创建 `Thread 线程对象名=new Thread(任务对象名);`
- 2) 线程的启动 `线程对象名.start();` //线程启动后 JVM 将自动执行指定任务中的 `run()` 方法（前提是该线程要能分配到 CPU）

四、线程状态

在 OS 中，通常是多个进程（线程）一起共用一个或多个 CPU，每个进程（线程）只能使用 CPU 一小段时间即要交出 CPU，因而进程（线程）在使用过程中存在多种状态切换。Java 线程的状态有：

- 1) 新建态（只是创建，尚未启动）
- 2) 就绪态（已启动但尚未分配到 CPU 资源）
- 3) 执行态（已获得 CPU 资源，真正执行）
- 4) 阻塞态（因等待某事件发生而主动交出 CPU 停止运行，在该事件发生前即使把处理机分配给该进程，线程也无法运行）
- 5) 完成态（整个任务执行完毕）

五、线程状态切换方法的使用说明

1) `yield()`

当前线程暂停执行，主动交出 CPU 让其它线程执行，其状态由执行态转至就绪态

2) `join()` throws `InterruptedException`

如果当前线程 a 所执行的任务的 `run()` 方法存在另一线程调用 `join()` 方法的语句，当 a 执行到此条语句时，a 即进入阻塞态，直到 b 结束（进入完成态）a 才重新转至就绪态。

3) `sleep(...)` throws `InterruptedException`

当前线程主动交出 CPU 让其它线程执行，其状态由执行态转至阻塞态，待指定时间的休眠期满后自动转至就绪态。

4) `wait(...)` throws `InterruptedException`

当前线程主动交出 CPU 让其它线程执行，其状态使得当前线程由执行态转入阻塞态，`wait` 方法有两种形式：

(1) 允许指定以毫秒为单位的一段时间作为参数，该线程直至被别的线程调用相应的 `notify()` 后或者过了指定的时间才重新由阻塞态转至就绪态；

(2) 没有参数，该线程直至被别的线程调用相应的 `notify()` 后才转至就绪态。

六、为什么要引入线程池

在前面的例子中，我们为每一任务均创建一个线程，当任务数量非常多时，则需创建非常多的线程，会导致程序的性能下降。为此，可创建一个包含一定数量线程的线程缓存池，如果其中某个线程完成了一个任务的执行，可以为其再行分配一个新的任务让其接着执行，从而避免了创建过多的线程。适用场合：通常用于执行相似的任务

七、为什么要使线程互斥

一个例子，例：`AccountWithoutSync1.java` 创建一个拥有 100 个线程的线程池，一个线程每次朝同一帐户存入一个便士，总共存 100 万次，最后一共有多少钱？

问题原因：此程序中帐户的操作语句块是临界区（帐户是临界资源），不能让多个线程同时进入。

线程互斥：当程序中存在临界资源时，为了使程序的执行具有可再现性，任一时间只能让一个线程对该资源进行操作即多个线程必须互斥地进入临界区。