Java 程序考试技巧和常用代码总结

I/O

- 控制台输入

```
public static String getInputLine() {// 获得用户从控制台输入的一行字符串
    BufferedReader br =
        new BufferedReader(new InputStreamReader(System.in));
    String inputLine = br.readLine();
    return inputLine;
}
```

文件 I/O 记住以下三个简单例子的用法基本可以应对程考中所有文件读写方面的问题

- 单行文件读取

```
import java.io.*;

BufferedReader reader =
    new BufferedReader(new FileReader("C:\\file.txt"));
String str = reader.readLine();
reader.close();
```

或者循环读取

```
BufferedReader reader =
    new BufferedReader(new FileReader("C:\\file.txt"));
String str = null;
while((str = reader.readLine())!=null){
    str = reader.readLine();
    //分析str
    //...
}
reader.close();
```

- 文件输出

```
BufferedWriter writer =
    new BufferedWriter(new FileWriter(new File("")));
    writer.write(str);
    writer.close();
```

图形界面

- 界面容器 JFrame JPanel frame.setSize(500, 500)

```
public class MyGUI extends JFrame {
  public MyGUI() { //在构造方法内设置界面的大小，可视性，
        this.setSize(500, 500); //设置界面的大小
        this.setVisible(true); //可视性
        //点击界面右上角小×时候的操作
        this.setDefaultCloseOperation(EXIT_ON_CLOSE);
```

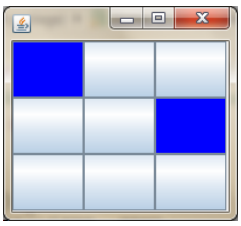
- 控件 JTextField JTextArea JLabel…

```
        JTextField textField = new JTextField();
        textField.getText(); //获取输入框内的文字
        textField.setText("Text to be displayed"); //设置输入框内的文字
        //创建一个JLabel对象，同时设置显示的文字内容
        JLabel label = new JLabel("My Label");
        this.add(textField); //最后别忘了将创建好的界面元素对象放在容器中
        this.add(label);}
```

- 布局 GridLayout BorderLayout

```
        //JButton[][] jbs = new JButton[3][3];//3*3的JButton数组
        this.setLayout(new GridLayout(3, 3));
        for (int i = 0; i < 3; ++i) {
            for (int j = 0; j < 3; ++j) {
                jbs[i][j] = new JButton();//接下来为按钮添加事件监听
                jbs[i][j].addMouseListener(new GameButtonListener());
                this.add(jbs[i][j]);}}
```

运行的效果见下图。用按钮数组来做棋盘非常方便！



- 棋子 == Jbutton。用一个按钮作为棋盘类游戏的一个格子。格子上的事件响应也就变成了一个按钮的事件响应。代码见上一知识点。
- 事件
  - 鼠标按下、松开，获取鼠标坐标 MouseListener addMouseListener

```
class MyMouseListener implements MouseListener {
    @Override
    public void mousePressed(MouseEvent e) {
        sx = e.getX(); //鼠标按下时候的x坐标
        sy = e.getY(); ////鼠标按下时候的y坐标
    }
```

- 按钮点击，同上！如何知道我点击的具体是哪一行哪一列的按钮呢？

```
    public void mousePressed(MouseEvent arg0) {
        JButton but = (JButton) arg0.getSource();
        for (int i = 0; i < 3; ++i) {
            for (int j = 0; j < 3; ++j) {
                if (jbs[i][j].equals(but)) { //
                    //被点击的按钮在数组中的index也就是(I,j)}
```

- 2D 图形绘制，重写 public void paint(Graphics g)，调用 jframe.repaint()来完成绘制
  - 画线

```
public void paint(Graphics g) { //切记，要在一个重写的方法内绘图
    super.paint(g);
    Graphics2D g2d = (Graphics2D) g;
    g2d.drawLine(sx, sy, ex, ey); //从(sx,sy)到(ex,ey)中画一条线
```

  - 画圆（画点） Ellipse2D

```
    Ellipse2D ee2d = new Ellipse2D.Double(ex, ey, 10, 10);
    g2d.setColor(Color.RED); //将颜色设置为红色
    g2d.draw(ee2d);//把圆的边线画出来
    g2d.fill(ee2d); //把圆填充
```

  - 画矩形 Rectangle2D

```
Rectangle2D r2d = new Rectangle2D.Double(sx, sy, ex-sx, ey-sy);
g2d.draw(r2d);
```

  - 画字符串

```
    Graphics2D g2d = (Graphics2D) g;
    g2d.drawString("String", (sx+ex)/2, (sy+ey)/2);
```

  - 颜色设置（按钮、线条、填充）

```
myJButton.setBackground(Color.BLUE); //将按钮设置为蓝色
//设置所画线条、字符串颜色或圆形、矩形等填充颜色
Graphics2D g2d = (Graphics2D) g;
g2d.setColor(Color.RED);
g2d.drawLine(sx, sy, ex, ey); //画出一条红色的线条
```

算法

- 没有特别要求算法的时间复杂度的时候就用最笨的方法（暴力求解）

字符串的处理

- 分割 str.split

```
String string = "a,b,c";
String []str_array = string.split(",");
```

- 出现的指定项 str.indexOf

```
int index = 0;
String string = "aba";
index = string.indexOf("a");    //index==0
index = string.lastIndexOf("a");//index==2
index = string.indexOf("c");    //index==-1
```

- 子字符串 str.subString

```
String str2 = string.substring(beginIndex, endIndex);
```

Warning: subString 函数不会对 str 对象产生任何影响

不要以为执行了一下代码以后 str 的值会变成”1”

String str = “123”

str.subString(0, 1); //这一行 statement 没有对 str 产生任何影响，仅仅是返回一个新串”1”

- trim()

从外部接受到字符串以后 trim 是对于字符串处理最常用的方法，使用之前检查一下是否为 null

- String 转化成数字: Integer.parseInt Float.parseFloat…

X value = X.parseX(string);

X 是要转换的类型，记住这个 coding 的模板。

- 数字转化成 String: ""+12 Integer.toString

- 日期和字符串之间的相互转化

数组

对于大小已知而且没有特殊限制和操作要求的数据用数组比较方便
但是如果涉及删除数据的话会比较麻烦

容器

（掌握增、删、改、查）

- ArrayList
- HashMap（enumeration）
- HashSet (enumeration)

对象的序列化：

虽然直接用上面介绍的文件 I/O 可以进行数据存取，但是对于某些题目使用对象序列化会十分方便，节省时间。

比如：家庭收支管理程序，图书馆的书籍管理程序，etc.

画图程序题还特别强调：要求结果可以存盘，但不必存成图片文件。

就是在暗示使用对象序列化方法简化文件存取部分的工作量。

对于需要序列化的对象需要 implements Serializable 接口,但不需要实现任何序列化的方法，非常方便

```
public class Book implements Serializable {
    //…
}
```

对象的存储

```
ObjectOutputStream out =
    new ObjectOutputStream(new FileOutputStream("c:\\book.ser"));
out.writeObject(books);
out.close();
```

对象的读取

```
ObjectInputStream in =
    new ObjectInputStream(new FileInputStream("c:\\book.ser"));
books = (ArrayList<Book>)in.readObject();
in.close();
```

图形界面使用 NetBeans + Eclipse 快速生成

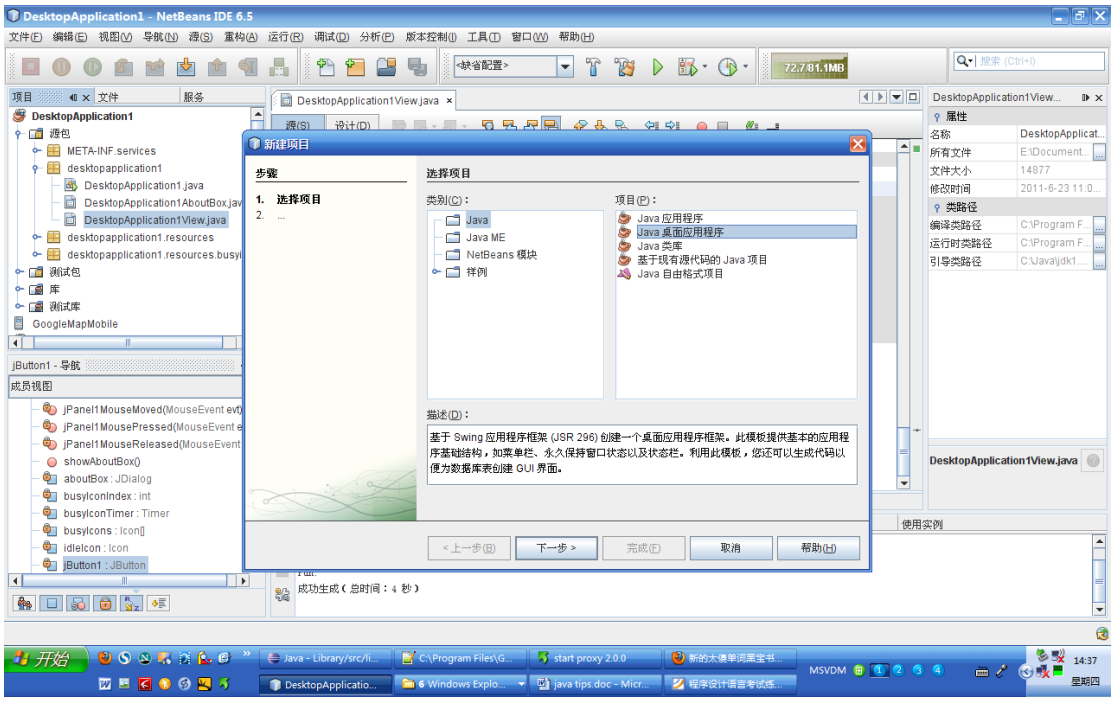
很多人选择 Eclipse 作为 Java 程序开发的 IDE，但是 Eclipse 不提供类似于 Visual Studio 的拖拽图形界面生成工具，所以使用 Eclipse 编写 Java 桌面应用程序需要自己敲图形界面的代码，费时费力，而且往往写出来的界面很难看。

NetBeans 提供方便图形界面拖拽功能，但是其外观让很多人不习惯。使用 NetBeans 写代码总感觉很别扭。

这里介绍利用 NetBeans 进行快速图形界面生成，并转换成 Eclipse 项目的方法，该方法适合图形界面需要许多图形控件的题目，可以短时间内生成好图形界面代码框架，节省宝贵的考试时间。

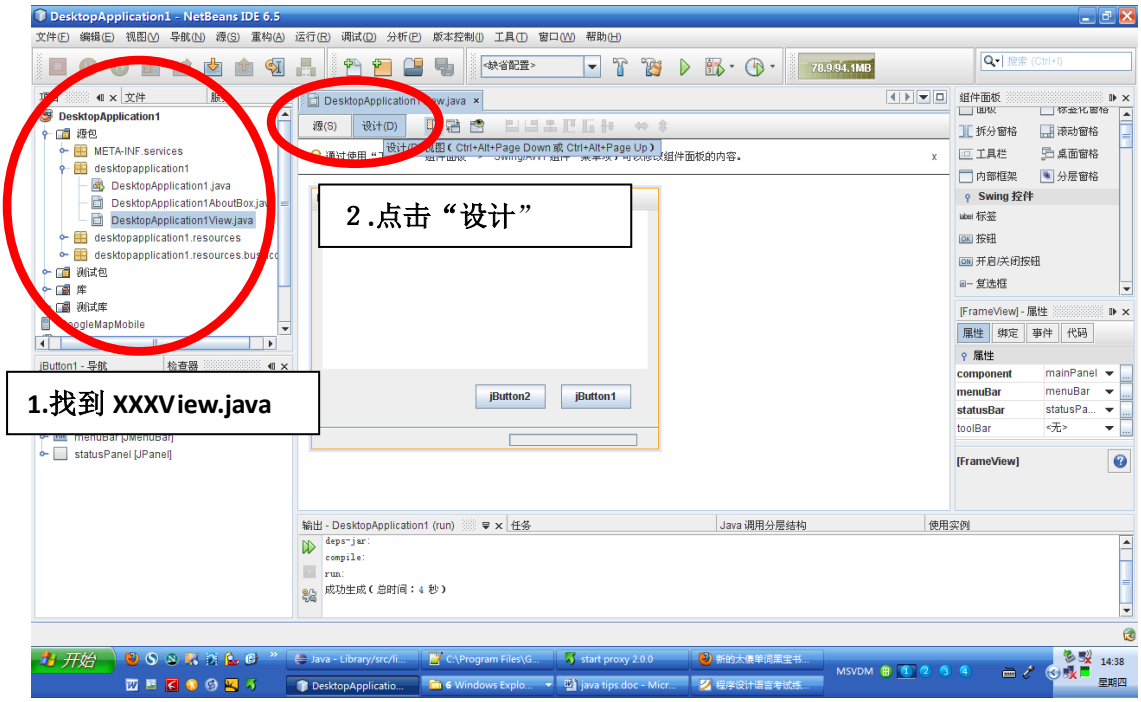
步骤：

1.打开 NetBeans 新建一个 Java 桌面应用程序

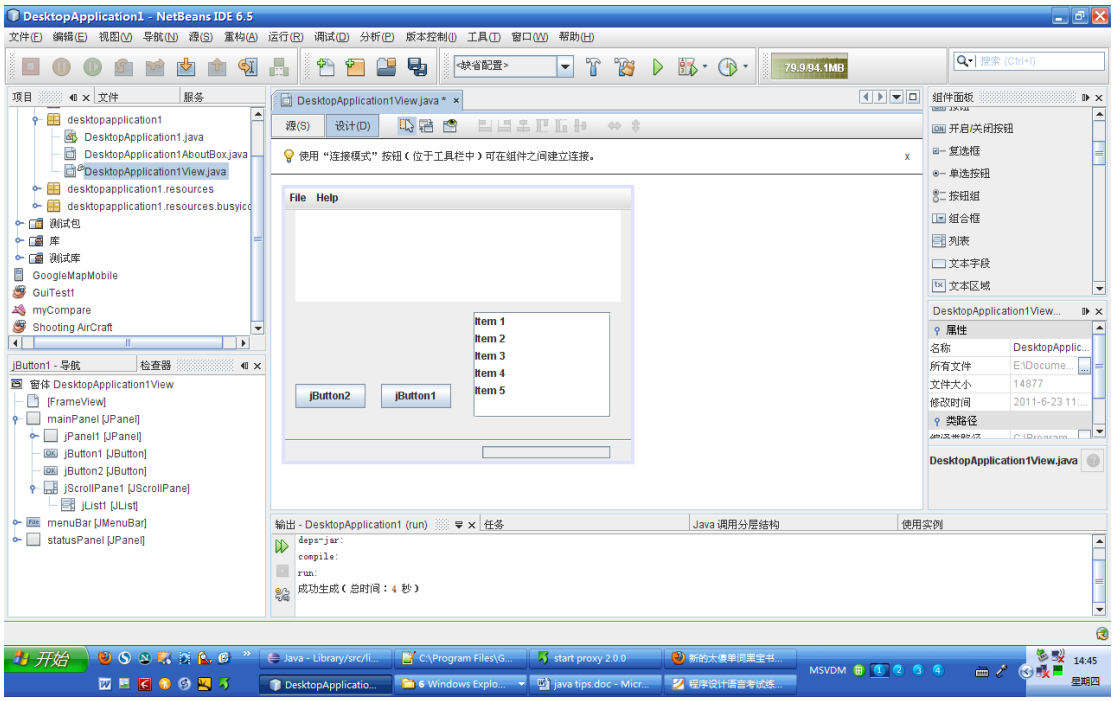


2.找到 *XXXX*View.java 文件双击打开，并点击“设计”，切换到图形设计界面

注: *XXXX* 是新建的项目名称

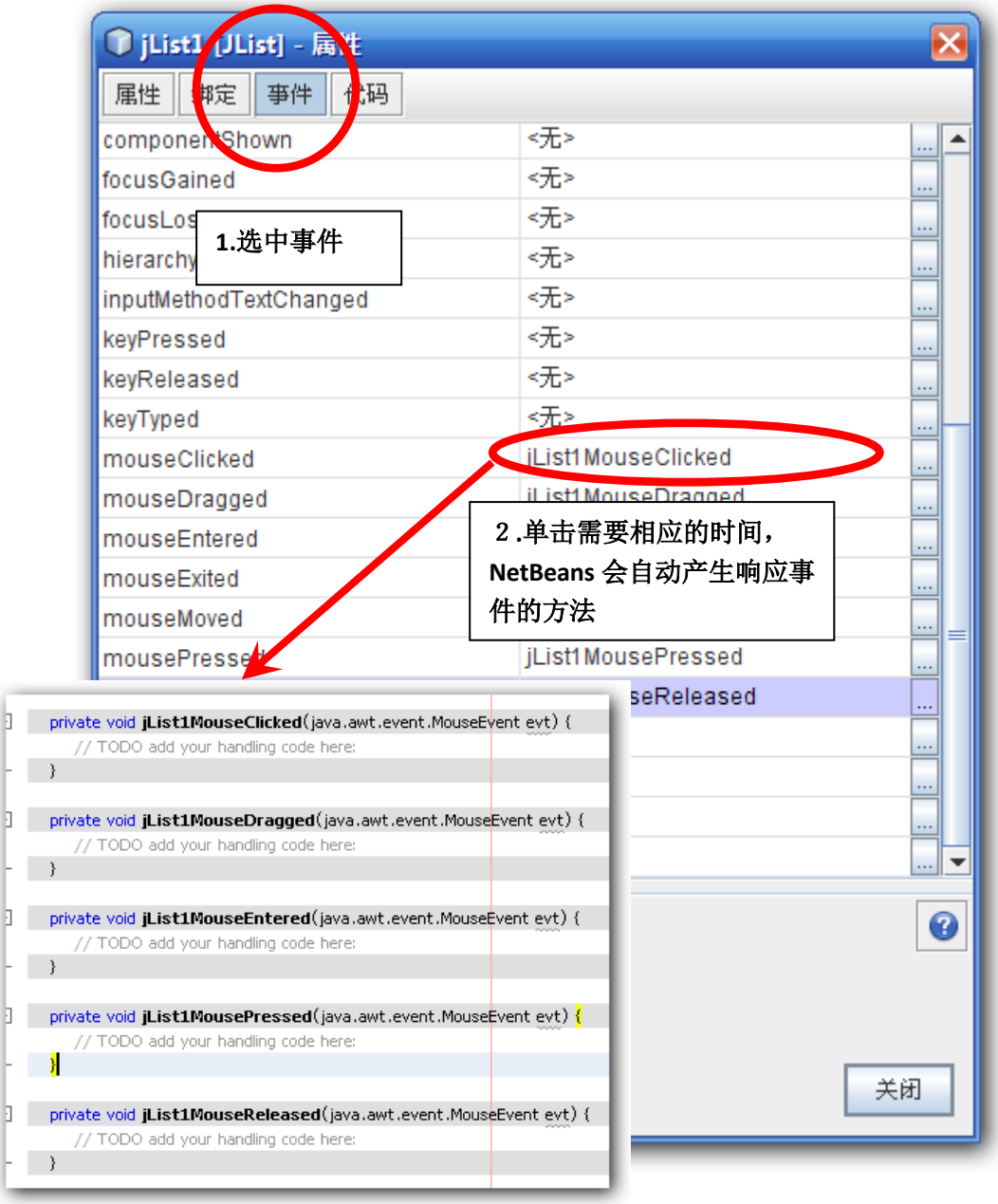


3.拖拽添加需要的所有控件

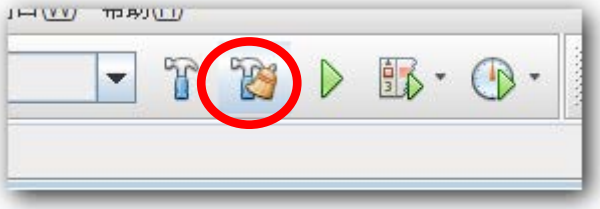


4 .对需要响应事件的控件：

点击鼠标右键->”属性”->上方选择”事件”



5.点击生成项目，产生必要的 jar 文件（不可忽略）

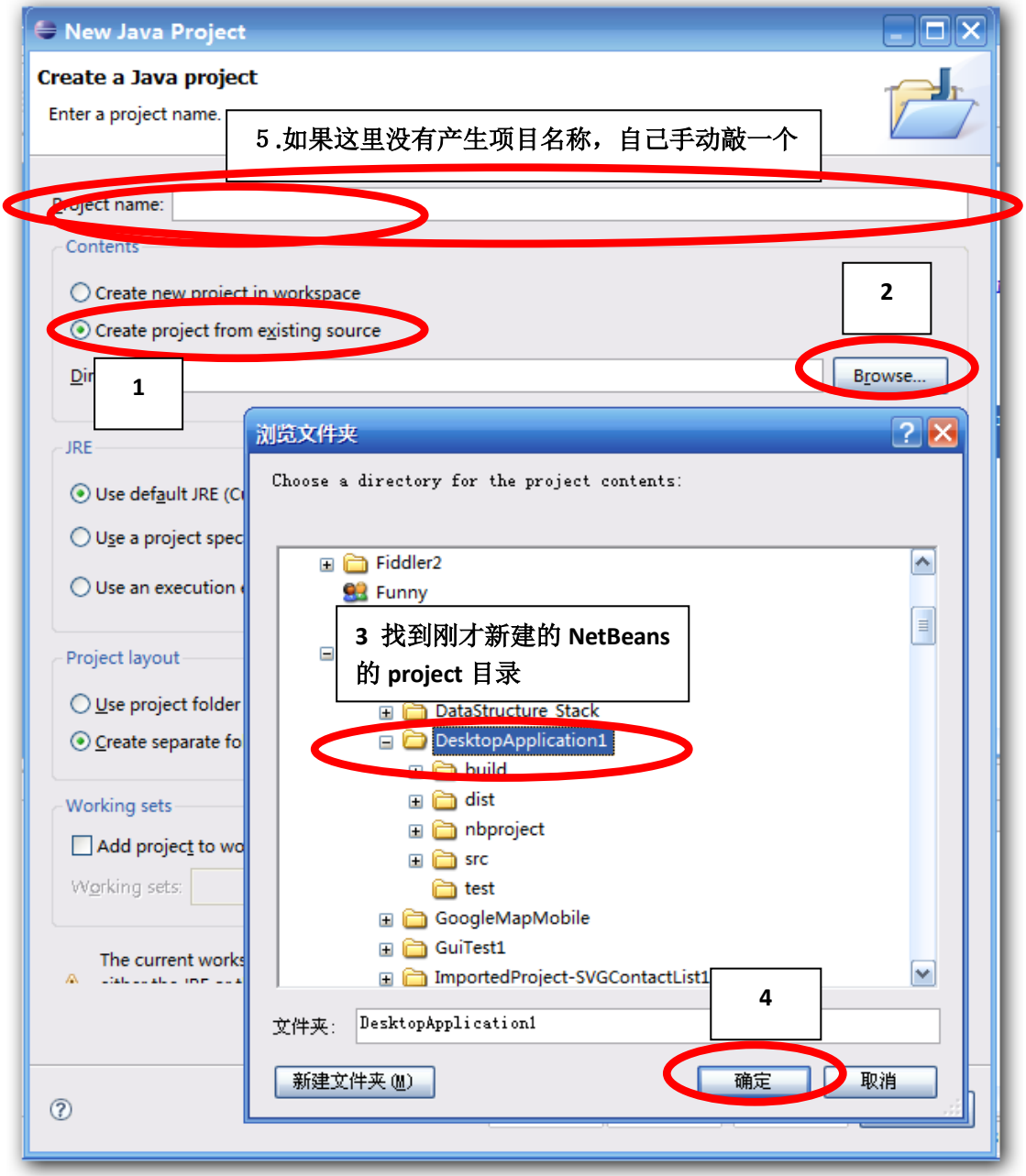


Warning: 这一步非常重要，用于产生 project 图形界面框架需要用到的 jar 包
如果不执行此步骤则无法在 Eclipse 环境下 build project

6.NetBeans 上面的工作做完了，下面换到 Eclipse IDE

在 Eclipse 中，”File”->”New”->”Java Project”

出现如下的对话框，选中”Create project from existing source”



7. 大功告成，打开 *XXXX*View.java 文件添加相应事件的代码，项目可以在 Eclipse IDE 中运行或者 debug

与代码无关的 Tips

- 利用开考前的热身时间，将一些肯定会用到的代码（例如从控制台读取输入、文件IO）先写下来。到正式开考的时候就可以直接复制使用了。
- 善用快捷键，加快代码编写速度。 代码补全快捷键：Alt + /，整行删除：ctrl + D，导入缺少的包：ctrl+shift+o
- 代码格式美化快捷键：ctrl+shift+f