



**Onze autoverzekering:
Wij staan dag en nacht voor u klaar.**

Centraal
beheer | achmea

- [Dream.In.Code> Programming Tutorials](#)
- > [Java Tutorials](#)(3 Pages)
- ▼
- 1
- 2
- 3
- ➡

Creating a basic Notepad Application Rate Topic: ★★★★★
5 Votes

gabehabe

Posted 05 October 2008 - 08:17 AM



POPULAR

Creating a basic notepad app

What do I already need to know in order to follow this tutorial?

Not a lot, actually. This stuff is a lot easier than people tend to make it look. My aim is to show you how simple it is to harness the power of Java. Just so long as you've got a basic grasp of the stuff, you should be able to take it in.


So what am I going to learn?

Hopefully, quite a bit! In this tutorial, I'm going to cover the basics of:

- Multiple inheritance (very basic explanation)
- GUI design
- Event handling
- File input/output
- Exception handling

Just to get your mouth watering, here's a screenshot of what we'll be making:

<http://www.dreamincode.net/forums/index.php?act=Attach&type=post&id=8512>

Isn't it sexy? 

So, to get started, we need to import a few things, namely:

```
1 import javax.swing.*; // for the main JFrame
  design
2 import java.awt.*; // for the GUI stuff
3 import java.awt.event.*; // for the event
  handling
4 import java.util.Scanner; // for reading from a
  file
5 import java.io.*; // for writing to a file
```

Next, let's go over multiple inheritance. Our notepad app is going to have the appearance of a JFrame, and the functionality of an ActionListener.

So, we're going to *extend* JFrame, and *implement* ActionListener. Makes sense, right?

```
1 public class Notepad extends JFrame implements
  ActionListener {
```

w00t!

Now we're going to set up our basic stuff. The classes that we'll be using for the design are:

Ask A Question

Join 500,000 **dream.in.code®** Developers

Java trainingen

computrain.nl/Java

Praktijkgerichte Java
programmeer trainingen.
Vraag de studiegids aan

Follow & Share

Dream.In.Code



Follow

+1



3308 readers
BY FEEDBURNER

Java Tutorials

[Phobos - A JavaFX](#)

[Games Engine: Part 2 -](#)

[JavaFX Scene API and
the FSM](#)

[Maven Tutorial 2 -](#)

[Adding Dependencies](#)

[Maven Tutorial 1 -](#)

[Installation and](#)

[Getting Started](#)

[Phobos - A JavaFX](#)

[Games Engine: Part 1 -](#)

[Intro to Threading and
DP](#)

General Discussion

Caffeine Lounge

Corner Cubicle

Student Campus

Software Development

Industry News

Introduce Yourself

Nightmare.In.Code

Programming Help

C and C++

VB.NET

Java

C#

ASP.NET

.NET Framework

VB6

PHP

Ruby

Python

ColdFusion

- TextArea
- MenuBar
- Menu
- MenuItem

So, let's go over it:

The TextArea The most important bit, right? This is the area of text that the user can write in. You'd never guess that, would you?!

So, the constructor for our TextArea is quite big~ it accepts quite a few parameters.

```
1 private TextArea textArea = new TextArea("",
  0,0, TextArea.SCROLLBARS_VERTICAL_ONLY);
```

Let's break it down.

First, we pass "" which is basically an empty string. The constructor is looking for the text to put in the TextArea initially, so you could pass "poo" and the TextArea would say poo when you launch the app.
Next, we have the size. 0,0 (Height/Width)
This might seem weird~ we're setting it to be 0x0 pixels, so it won't be visible, right? Wrong. The basic GUI design in Java will automatically set the TextArea to fill the window. In other words, we don't have to worry about it. TextArea.SCROLLBARS_VERTICAL_ONLY adds WordWrap to our application, to make it look neater. Times where you *shouldn't* have WordWrap include code views (it all belongs on one line)

The MenuBar

Much simpler, the code speaks for itself:

```
1 private MenuBar menuBar = new MenuBar(); //
  first, create a MenuBar item
2 private Menu file = new Menu(); // our File menu
3 // what's going in File? let's see...
4 private MenuItem openFile = new MenuItem(); //
  an open option
5 private MenuItem saveFile = new MenuItem(); // a
  save option
6 private MenuItem close = new MenuItem(); // and
  a close option!
```

On to the constructor!

Personally, I find it easier to go along code when it's together. So, I've added enough comments in to the constructor code for it to make sense:

[Swing to JavaFX](#)

[Swing, Top-Down 2](#)

[Swing, Top-Down \(with GridBagLayout\)](#)

[Basic Java: Types, Variables, Operators](#)

[Simple Regression Library Part 2 - Linear Regression Model](#)

[Simple Regression Library Part 1 - Regression Models](#)

[215 More Java Tutorials...](#)

Reference Sheets



Code Snippets

[C Snippets](#)

[C++ Snippets](#)

[Java Snippets](#)

[Visual Basic Snippets](#)

[C# Snippets](#)

[VB.NET Snippets](#)

[PHP Snippets](#)

[Python Snippets](#)

[Ruby Snippets](#)

[ColdFusion Snippets](#)

[SQL Snippets](#)

[Assembly Snippets](#)

[Functional](#)

[Programming Snippets](#)

[Perl Snippets](#)

[HTML/CSS Snippets](#)

[Javascript Snippets](#)

Databases

Other Languages

Game Development

Mobile Development

52 Weeks Of Code

Web Development

Web Development

HTML & CSS

JavaScript

Graphic Design

Flash & ActionScript

Blogging

SEO & Advertising

Web Servers & Hosting

Site Check

```

01 public Notepad() {
02     this.setSize(500, 300); // set the
    initial size of the window
03     this.setTitle("Java Notepad Tutorial");
    // set the title of the window
04     setDefaultCloseOperation(EXIT_ON_CLOSE);
    // set the default close operation (exit when
    it gets closed)
05     this.textArea.setFont(new Font("Century
    Gothic", Font.BOLD, 12)); // set a default
    font for the TextArea
06     // this is why we didn't have to worry
    about the size of the TextArea!
07     this.getContentPane().setLayout(new
    BorderLayout()); // the BorderLayout bit
    makes it fill it automatically
08     this.getContentPane().add(textArea);
09
10     // add our menu bar into the GUI
11     this.setMenuBar(this.menuBar);
12     this.menuBar.add(this.file); // we'll
    configure this later
13
14     // first off, the design of the menuBar
    itself. Pretty simple, all we need to do
15     // is add a couple of menus, which will
    be populated later on
16     this.file.setLabel("File");
17
18     // now it's time to work with the menu.
    I'm only going to add a basic File menu
19     // but you could add more!
20
21     // now we can start working on the

```

[Flash/ActionScript Snippets](#)

[Other Languages Snippets](#)

[DIC Chatroom](#)

[Join our IRC Chat](#)

[Bye Bye Ads](#)

DIC++

TRADING 212

LEREN HANDELEN

GRATIS DEMO-ACCOUNT MET € 10 000

\$ FOREX

OLIE

GOUD

EFFECTEN

See? That wasn't so bad now, was it? I told you this stuff is easy~ it's just a little repetitive at times.

Listening for Events

Remember how I said earlier that we *implement* ActionListener? Well, now it's time to learn about events. Because we're implementing ActionListener, we can have a function called `ActionPerformed()` which will listen for events for us. Neat, huh? Again, I've added plenty of comments so you can see what's happening along the way.

NOTE: This section will also cover the basics of file I/O, and the use of JFileChooser

Find us on Facebook



Dream.In.Code

Like

58,583 people like Dream.In.Code.



Facebook social plugin

```
01 public void actionPerformed (ActionEvent e) {
02     // if the source of the event was our
    "close" option
03     if (e.getSource() == this.close)
04         this.dispose(); // dispose all
        resources and close the application
05
06     // if the source was the "open" option
07     else if (e.getSource() == this.openFile)
08     {
09         JFileChooser open = new
        JFileChooser(); // open up a file chooser (a
        dialog for the user to browse files to open)
10         int option =
        open.showOpenDialog(this); // get the option
        that the user selected (approve or cancel)
11         // NOTE: because we are OPENing a
        file, we call showOpenDialog~
12         // if the user clicked OK, we have
        "APPROVE_OPTION"
13         // so we want to open the file
        if (option ==
        JFileChooser.APPROVE_OPTION) {
14             this.textArea.setText(""); //
            clear the TextArea before applying the file
            contents
15             try {
16                 // create a scanner to read
                the file (getSelectedFile().getPath() will
                get the path to the file)
17                 Scanner scan = new Scanner(new
                FileReader(open.getSelectedFile().getPath()));
18                 while (scan.hasNext()) {
```

Oh, and don't forget to close off the class! }

So, you should now have a basic Notepad app!

As with all my other tutorials, I'll post the complete code, in case you got lost along the way:

THE FINAL PRODUCT!

```

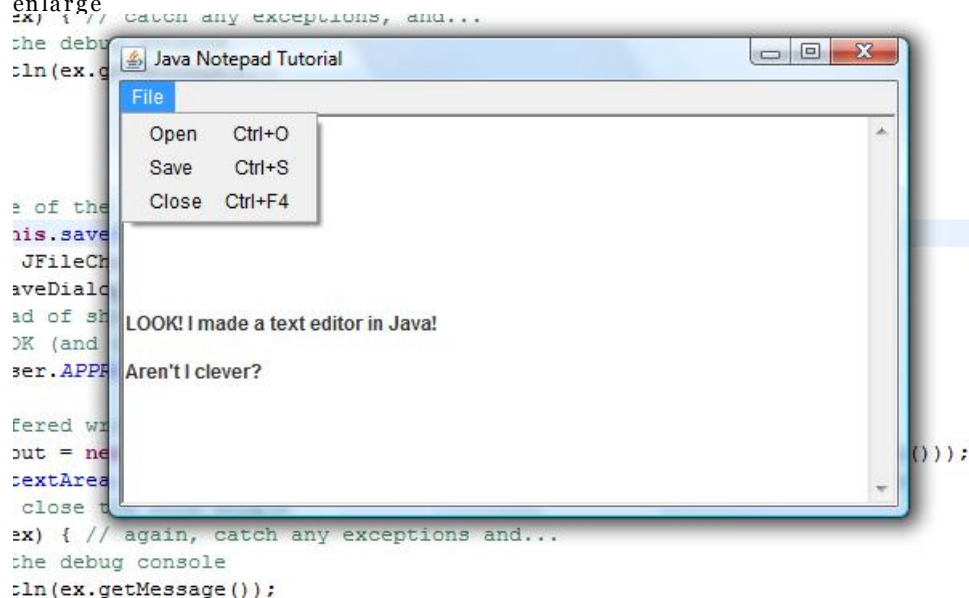
001 import javax.swing.*;
002 import java.awt.*;
003 import java.awt.event.*;
004 import java.util.Scanner;
005 import java.io.*;
006
007 public class Notepad extends JFrame
    implements ActionListener {
008     private TextArea textArea = new
        TextArea("", 0,0,
        TextArea.SCROLLBARS_VERTICAL_ONLY);
009     private MenuBar menuBar = new MenuBar();
        // first, create a MenuBar item
010     private Menu file = new Menu(); // our
        File menu
011     // what's going in File? let's see...
012     private MenuItem openFile = new
        MenuItem(); // an open option
013     private MenuItem saveFile = new
        MenuItem(); // a save option
014     private MenuItem close = new MenuItem();
        // and a close option!
015
016     public Notepad() {
017         this.setSize(500, 300); // set the
        initial size of the window
018         this.setTitle("Java Notepad
        Tutorial"); // set the title of the window
019         setDefaultCloseOperation(EXIT_ON_CLOSE);
        // set the default close operation (exit when
        gets closed)
020         this.textArea.setFont(new
        Font("Century Gothic", Font.BOLD, 12)); //

```

Happy coding!

Attached image(s)

- Resized to 79% (was 640 x 373) - Click image to enlarge





Replies To: Creating a basic Notepad Application

Locke

Posted 05 October 2008 - 08:55 AM

Nice tutorial!



ran123

Posted 02 December 2008 - 11:16 AM

How do I run this notepad program. I tried adding the main method and then creating an instance of it, but nothing happened.

skywalkno8

Posted 07 December 2008 - 11:15 PM

actually i did the same but no action at all..
n how do i place the main method n wat should i put into main method..please!!

gabehabe

Posted 08 December 2008 - 11:25 AM



My bad~!

I musta been busy that day. All you need to do is add a function called "main" into your class, and away you go!



```
1 public static void main(String args[]) {
2     Notepad app = new Notepad();
3     app.setVisible(true);
4 }
```

skywalkno8

Posted 09 December 2008 - 12:30 AM

halo guys.. i do have a questioned here how do i created and set font list on a default application..

Ostralis

Posted 13 December 2008 - 01:29 PM

The tutorial looks good, I'll definitely have a go at it and thanks for sharing! This will help bunches!

sam_a_thebest

Posted 16 December 2008 - 10:18 PM


very very nice



5thWall


Posted 18 December 2008 - 10:14 PM


Thanks! We haven't done anything with Java Swing in any

of my classes so far.  So I've been looking for tutorials to get into it with. One question though? What's with all the this.whatever? Is there a specific reason you do that? I took them all out and my code works fine.

bastones

Posted 13 January 2009 - 05:07 PM

 (<http://www.dreamincode.net/forums/index.php?s=70f9b4be0db5f9cf1ac787a51deec27e&app=forums&module=forums§ion=findpost&pid=493829>)
 5thWall, on 18 Dec, 2008 - 09:14 PM, said:
 Thanks! We haven't done anything with Java Swing in any

of my classes so far.  So I've been looking for tutorials to get into it with. One question though? What's with all the this.whatever? Is there a specific reason you do that? I took them all out and my code works fine.

The this keyword refers to the fields (i.e. the variables at the start of the code) such as:


```
1 private MenuBar menuBar = new MenuBar(); //
  first, create a MenuBar item
2 private Menu file = new Menu(); // our File
  menu
3 // what's going in File? let's see...
4 private MenuItem openFile = new MenuItem();
  // an open option
5 private MenuItem saveFile = new MenuItem();
  // a save option
6 private MenuItem close = new MenuItem(); //
  and a close option!
```

So to refer to these in the constructor we use this.fieldName - variables that are not inside methods or constructors are called **fields**. Because in this tutorial it is extending (inheriting) JFrame the this.setSize, etc., is referring to the setSize method of the JFrame class.

5thWall

Posted 13 January 2009 - 06:48 PM

Thanks bastones, but I know what the this keyword does.


 I've only seen it used when someone names a method argument the same as an object level variable. Like so:


```
01 private int A;
02 private int B;
03
04 //We use the name A twice so we use 'this' to
  tell them apart.
05 public setA(int A) {
06     this.A = A;
07 }
08
09 //The name B is only used once so we don't need
  'this' here.
10 public setB(int _B)/> {
11     B = _B;
12 }
```

So I was wondering if there was a special reason why you would use this elsewhere while writing for Swing. This post has been edited by **5thWall**: 13 January 2009 - 10:41 PM

bastones

Posted 14 January 2009 - 11:21 AM

 (<http://www.dreamincode.net/forums/index.php?s=70f9b4be0db5f9cf1ac787a51deec27e&app=forums&module=forums§ion=findpost&pid=511477>)
 5thWall, on 13 Jan, 2009 - 05:48 PM, said:
 Thanks bastones, but I know what the this keyword does.

 I've only seen it used when someone names a

method argument the same as an object level variable.
Like so:

```

01 private int A;
02 private int B;
03
04 //We use the name A twice so we use 'this' to
    tell them apart.
05 public setA(int A) {
06     this.A = A;
07 }
08
09 //The name B is only used once so we don't need
    'this' here.
10 public setB(int _B)/> {
11     B = _B;
12 }

```

So I was wondering if there was a special reason why you would use this elsewhere while writing for Swing.

Well that's exactly why you'd use the this keyword, incase there are paramater variables that are the same name as your fields names. You can still call your fields via its actual name too under those conditions. Take a look [here](http://java.sun.com/docs/books/tutorial/java/javaOO/thiskey.html) (<http://java.sun.com/docs/books/tutorial/java/javaOO/thiskey.html>) for a quick overview of the keyword because there are a few more uses for it.

gabehabe

Posted 18 January 2009 - 09:34 AM

Sorry about the late response~!

Quote

halo guys.. i do have a questioned here how do i created and set font list on a default application..


I actually have a snippet for a "font dialog" class.


<http://www.dreamincode.net/snippet2571.htm>
(<http://www.dreamincode.net/code/snippet2571.htm>)

Enjoy~! 


joezim007

Posted 23 January 2009 - 09:46 AM

 (<http://www.dreamincode.net/forums/index.php?s=70f9b4be0db5f9cf1ac787a51deec27e&app=forums&module=forums§ion=findpost&pid=511389>) bastones, on 13 Jan, 2009 - 05:07 PM, said:

 (<http://www.dreamincode.net/forums/index.php?s=70f9b4be0db5f9cf1ac787a51deec27e&app=forums&module=forums§ion=findpost&pid=493829>) 5thWall, on 18 Dec, 2008 - 09:14 PM, said:

Thanks! We haven't done anything with Java Swing in any

of my classes so far.  So I've been looking for tutorials to get into it with. One question though? What's with all the this.whatever? Is there a specific reason you do that? I took them all out and my code works fine.

The this keyword refers to the fields (i.e. the variables at the start of the code) such as:


```

1 private MenuBar menuBar = new MenuBar(); //
  first, create a MenuBar item
2 private Menu file = new Menu(); // our File
  menu
3 // what's going in File? let's see...
4 private MenuItem openFile = new MenuItem();
  // an open option
5 private MenuItem saveFile = new MenuItem();
  // a save option
6 private MenuItem close = new MenuItem(); //
  and a close option!

```

So to refer to these in the constructor we use this.fieldName - variables that are not inside methods or constructors are called **fields**. Because in this tutorial it is extending (inheriting) JFrame the this.setSize, etc., is referring to the setSize method of the JFrame class.

I was wondering about this too. Seems kinda pointless and redundant to use "this" all the time.

joezim007

Posted 24 January 2009 - 12:49 PM

I was wondering if you had any reasons for using TextArea, MenuBar, Menu, and MenuItem over the JTextArea, JMenuBar, JMenu, and JMenuItem alternatives? Are there any downsides to the J components?

- (3 Pages)
- ▼
- 1
- 2
- 3
- ➡



Related Java Topics^{beta}

[Basic](#)

[Notepad](#)

[Application](#)

[- Getting](#)

[Error](#)

[Messages](#)

[While](#)

[Trying To](#)

[Run A Basic](#)

[Notepad](#)

[Application](#)

[Error In Gui](#)

[Application](#)

[Full](#)

[Application](#)

[Tutorials](#)

[How To](#)

[Know If](#)

[TextArea](#)

[\(awt\) Is](#)

[Being](#)

[Typed?](#)

[How To](#)

[Encrypt The](#)

[Documents](#)

[That Get](#)

[Saved In](#)

[This](#)

[Notepad](#)

[Like App](#)

[Creating A](#)

[Java](#)

[Notepad](#)

[With File](#)

[I/O And](#)

[Menus](#)

[Notepad](#)

[Application](#)

[- Coding In](#)

[Awt](#)

[Creating A](#)

[Calendar](#)

[Viewer](#)

[Application](#)

[- GUIs,](#)

[Renderers,](#)

[JTable](#)

[Tutorial](#)

Tutorial

[Basic GUI In](#)

[JAVA \(using](#)

[JFrames\) -](#)

[Discussing](#)

[Some](#)

[Aspects In](#)

[Creating](#)

[Graphical](#)

[User](#)

[Interface In](#)

[Java](#)

Tutorial

[Trying To](#)

[Create A](#)

[Java](#)
[Notepad -](#)
[Getting An](#)
[Error](#)
[Concerning](#)
[The](#)
[Constructor](#)

[FAQ](#) | [Team Blog](#) | [Feedback/Support](#) | [Advertising](#)  | [Terms of Use](#) | [Privacy Policy](#) |
[About Us](#)

Copyright 2001-2014 [MediaGroup1 LLC](#), All Rights Reserved
A [MediaGroup1 LLC](#) Production - Version 6.0.2.1.36
Server: secure3