Hello Emanuel,

very straight forward solution.  I noticed that you introduced a public method to verify the user’s input using regular expressions. Nevertheless at the end you are actually not using it.  
  
I assume that you have opted to parse the input using Double.parseDouble after testing and reviewing  your Regular Expression approach?  
To my understanding Regular expressions will only help to avoid parser problems as you can „only“ make sure that the input to the parser is valid. Since the relevant parsers such as from the NumberFormat or Double class will „stumble“ over the problem anyhow, using Regular Expressions seems to kill the cow twice.  
Would you mind sharing the reason why you came away from your solution based on regular expressions?

My second thought is in respect to your approach to identify the button where the event originated from:  
From what I understand you are identifying the button by its label such as „Cancel“ since you are not setting an action command explicitly in the event.  
From what I recall, using getActionCommand is beneficial once a button “toggles” something for instance such as “show/hide” making the “common” getSource() approach  useless as the object will be the same regardless its current label.  
Nevertheless I think in a system supporting various languages, getActionCommand may become dangerous once button labels are translated.  
What do you think?

Best regards,  
Daniel

Hi Daniel,

Thank you for looking at my program. The isInputValid method was included by mistake. This was actually intended for the DQ1 assignment. Had I used it here, it may have presented a problem since the regular expression is checking for the format of (any number of numeric charcters).(2 numeric characters), which is dollar format.

Regarding the getAction command, I’m not sure I understand what you mean. Could you elaborate further? I used the example in our text to help me on handling JButton events (Deitel and Deitel, 2011, p. 572). Good point on the danger of translating buttons labels; I wasn’t anticipating supporting systems with different languages.

Regards,

Emanuel

Reference:

Deitel, P. & Deitel H. (2011) *Java How to Program* 9th Edition p. 572. Pearson HE, Inc.. [Kindle Edition].

Hi Emanuel,  
Thanks for sharing your solution with us!

well-format and clearly OO design, but I have one question that why you don't consider split the two buttion action into two different method separately in <actionPerformed>, which could make the code more readable. How do you think so ?

e.g  
private class ButtonHandler implements ActionListener {  
 // process button events public void actionPerformed(ActionEvent event) {

      if (event.getActionCommand().contentEquals("Convert to GBP")) {

         //DoConversion();

       }

      else if (event.getActionCommand().contentEquals("Cancel")) {

  //Quite();

        }  
  }

}  
  
private void doConversion(){

//...

}  
private void Quite(){

 //...

}  
  
Regards, Liu

Hi Liu,

Thank you for looking at my code and the comment regarding readability. I think we you say is a good idea. I’m assuming that when you said “Quite”, you meant to say “Quit”, but since that process is one line (System.exit(0)), I would not give it its own method. Looking back, I would have probably broken it up like this:

public void actionPerformed(Action event) {

if (event.getActionCommand().contentEquals("Convert to GBP")) {

doConversion();

} else if (event.getActionCommand().contentEquals("Cancel")) {

System.exit(0);

}

}

public void doConversion() {

//

}

Regards,

Emanuel

Hi Delisa,

I tried out your code, and it works. I must say, you chose an interesting way of solving the problem, which I am not quite sure, is object-oriented. I just have a couple of recommendations:

1. Validate that the input received is limited to only 2 decimal places.
2. Adding comments to some of your closing curly braces. Since tabs are inconsistent among editors, formatting might not appear the way you saw it, and I had a hard time being able to tell which curly brace ended a block.

Regards,

Emanuel

Hi Fatai,

I tried to compile your program on my Mac, and I got the following errors:

EMANUELs-MacBook-Pro:Week4DQ2M3 langite$ javac \*.java

SquareNumber.java:83: <identifier> expected

catch (NullPointerException | NumberFormatException e)

^

SquareNumber.java:83: '{' expected

catch (NullPointerException | NumberFormatException e)

^

SquareNumber.java:83: not a statement

catch (NullPointerException | NumberFormatException e)

^

SquareNumber.java:83: ';' expected

catch (NullPointerException | NumberFormatException e)

^

SquareNumber.java:89: illegal start of expression

public double doSquare(double number)

^

SquareNumber.java:89: ';' expected

public double doSquare(double number)

^

SquareNumber.java:89: ';' expected

public double doSquare(double number)

^

SquareNumber.java:94: reached end of file while parsing

}

^

8 errors

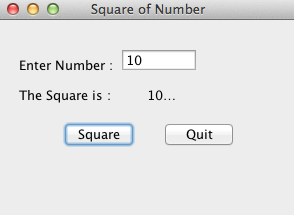
I edited line from:

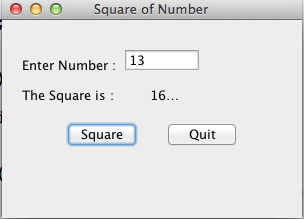
catch (NullPointerException | NumberFormatException e)

to:

catch (NumberFormatException e)

and then it compiled. It worked for small numbers, but when I tried to use a number whose square was more than 2 digits, it would print out only the first two digits, and then “…”:





Hello Emanuel,  
I apologize for not being clear what I meant. I came across the “actionCommand” example (Deitel & Deitel 2012, Fig. 14.15) as well and immediately thought about internationalizing an application.  
This is certainly due to the fact that I am working for a company where English and French are in concurrence to be the primary language.

If you are looking at Figure 14.9 (Deitel & Deitel 2012, p.562) you will see that the authors are using the getSource method from the ActionEvent class to determine which component (or better user interaction with the component) triggered the event.  
From what I understand this is the most straight forward approach (you will find it in numerous online Java tutorials as well).  
Nevertheless the ActionEvent class also offers a convenient way of customizing event handling by introducing an “action-command” String along with the object.  
This becomes handy for the JToggleButton for instance where the object triggering the event will be the same, but the action-listener has to determine further actions from the current state of the object associated to the visual component.   
To my understanding Java implicitly sets the action command to equal the label/caption of a button if the action-command is not explicitly set.  
At this point I actually see the “risk” coming from translating button labels to the user’s language since your conditional statement in the ActionListener object will no longer evaluate to true.  
The JButton class (as an implementation of the AbstractButton class) for instance offers a method setActionCommand to specify the action command that is sent along with the event (Oracle 2011).

Once the action-command is explicitly set to something unique such as “ACTION\_CLOSE” it becomes independent from the button’s actual label/caption and will work even if the visual part of the button is change due to translation.  
The way I see it, using getActionCommand() to determine what to do is certainly a way to handle events, but only if the action-command is explicitly set up-front.  
JButton test = new JButton(“Test”);  
test.setActionCommand(“PERFORM\_TEST”);  
test.addActionListener( new ActionListener() {  
public void actionPerformed(ActionEvent e) {  
if (e.getActionCommand().equals(“PERFORM\_TEST”)) {  
…..  
}  
}  
});  
What do you think?  
With best regards,  
Daniel  
References:  
Deitel P. & Deitel H. (2012) ‘Java How to program’. Prentice Hall 9th Edition (eBook). ISBN-13: 978-0-13-257566-9  
Oracle (2011) ‘Class AbstractButton’. [Online]. Available from [http://docs.oracle.com/javase/6/docs/api/javax/swing/AbstractButton.html#setActionCommand(java.lang.String](http://docs.oracle.com/javase/6/docs/api/javax/swing/AbstractButton.html#setActionCommand%28java.lang.String)) [Accessed July 30th 2013]

Hi Daniel,

For some reason, your replies weren’t showing up in my thread. It must be the new forum we have.Thank you for the advice. I actually used your suggested method in this week’s assignment, just to give it a try. In the end, It looks like the same amount of work though, when compared to using to contentEquals method.

Regards,

Emanuel

Hi Emanuel,

Thanks for sharing your code with us.

I like your error handling routine and the pattern matching line of code:

if (inputGiven.matches("\\d\*(\\.\\d{2})?")) {

I noticed that you imported individual swing component that you used in your application.

I think a single import  statement would have sufficed e.g

import javax.swing.\*;

What do you think?

Best regards,

Ifeanyi.

Hi Ifeanyi,

Thank you for looking at my code. The import statement you suggest would work, but the reason I did individual swing components was really just to be efficient resource-wise, in that I only wanted to import what I will use. If I were to import all swing components, then there could be various components that I am importing that will be unused.

Regards,

Emanuel