Trikebot Java Programming Primer

A complete step-by-step guide to creating a simple example program.

Here's a step-by-step example to help you familiarize yourself with the environment we will be working with and the Java language. Doing this example all the way through should banish most of your questions on programming.

This guide will describe precisely the steps needed to make a really useless Java program using the code base we have prepared and command-line Java tools (which you should already have installed). Try this example, and then you can work from there to create programs that actually do something!

1. First, obtain the source.tar.gz file either by downloading it from the web site or copying it off the CD. Unzip it using WinZip. Open up the folder and note all the .java files inside. We'll only be modifying two of them, UserWindow.java and UserThread.java. UserWindow.java is responsible for creating the window where you will be able to add your own custom commands, and UserThread.java does the actual work of those commands, running them in a separate thread, so you can still use the interface while running your own commands.
2. Open the text editor. I recommend using Jext; it is an easy-to-use editor written in Java and designed especially for working with Java code. If you prefer, you can use WordPad, a primitive text editor that comes with Windows, or you can use XEmacs, a port of a Unix editor which has many potentially powerful features but which can also be confusing and frustrating.
3. Open UserWindow.java and UserThread.java.
4. The program itself is actually a simple shell that opens up all the windows that do interesting things. It creates a bunch of windows, known as frames in the Java code. Most of the windows contain parts of the graphical interface for controlling your robot by hand from your computer and for viewing what the robot's sensors are viewing. One of them contains empty text fields and a bunch of buttons that do nothing. It's here for us to add the features that we want, without having to worry about how to create a GUI. For now, we'll be building a really simple calculator program that adds two numbers and gives us the result.
5. Bring up a command-line window by selecting "Run" from the start menu and then typing "`cmd`"; this brings up a command-line window. Make use of the `cd` command to get to the directory with the code in it; this is most easily done by typing "`cd `" (note the space) and then dragging over to the command line window the folder with the code in it, which will enter the full path name to that folder on the command line. Once at the source folder, type "`javac *.java`" to compile all the Java files. This will produce lots of .class files containing the executable code. Now type "`java Shell`" to launch the interface. Lots of windows should come up; this is how we will control the robot. Quit the program, and now type "`java UserWindow`"; this will launch just the window that we'll need to modify for this exercise. Note that there are buttons and text fields labeled "Action 1" through "Action 10" and "Data 01" through "Data 20",

respectively. Click on the buttons; notice how none of them do anything. Click in one of text fields; notice how you can type anything you want there. We can also place values there via Java code.

6. Go back to the editor and do a search in UserWindow.java for "CHANGE BUTTON TEXT HERE", without the quotes. You should a find section near the beginning of the file. After this text comes the code that sets the label for each of the buttons. By changing the text in the quotes, you can rename the buttons to something more appropriate. Let's rename the first button to "Add"; change text "Action 1" to "Add".
7. Right below that previous section is very similar code that sets the labels for the twenty text fields. Again, we'll just be changing the names inside the quotes. Let's change "Data 01" into "Addend 1", "Data 02" into "Addend 2", and "Data 03" into "Sum". Go ahead and compile and run the program again, to see what effect these changes have had.
8. Now, we're going to make it so that clicking the Add button actually adds the two addends and displays the result in the Sum field. Go back to the source code, and do a search in UserThread.java for "CHANGE BUTTONS TO ACTUALLY DO THINGS HERE", again without the quotes. You should find a section partway into the file. In that section are ten functions with named Action1 to Action10, and they get called when you click that button. Don't change their names; that would keep them from working. We'll be creating new functions that actually carry out the behaviors we want, and then just adding one line of code inside the Action functions to call our new functions. Since the button that we renamed "Add" is button 1, we'll be modifying Action1 for starters. We'll make it so that Action1 calls Add, which is a function we will now write:

```
private void Action1()
{
      Add();
}
```

9. The numbers that we enter will go into the textfields, and thus they will be text as far as the computer is concerned. We can't add text, so we have to convert them to numbers first. To read the contents of each text field, we have to access the window containing it, which has already been conveniently assigned to the variable `theWindow`, then give its name, which will be of the form `TextField1`, and call the `getText()` method to get the current contents. But, this gives us the contents as a `String`. We'll go ahead and store them in a `String` Object, like so:

```
String string1;
string1 = theWindow.TextField1.getText();
```

We can do likewise for the second input field.

```
String string2;
string2 = theWindow.TextField2.getText();
```

10. To get these strings into a form we can add, we will use the `parseInt(String)` method of the `Integer` class. `Integer` (with a capital "I") is an Object that wraps a value of type `int` (with a lower-case "i") which is just a naked number. Only Objects and Classes can have methods, and Java allows neither global variables nor global functions, so all functionality has to be in a method. Functions which deal with Objects of certain types but which do not act on an already existing Object are best implemented as static methods, meaning that we call them on an entire Class instead of a single Object. `parseInt()` is a static method; it returns a new Object of type `Integer` containing the number expressed in the `String` we passed it. We'll store this in a plain old `int` that we declare. Putting this all together, we have the following:

```
int add1;
add1 = Integer.parseInt(string1);
```

and

```
int add2;
add2 = Integer.parseInt(string2);
```

11. Adding these two numbers is easy:

```
int returnval;
returnval = add1 + add2;
```

12. Now, we just need to convert that number back into a `String` to display it in one of the text fields. Again, we will use a static method, this time of Class String. `String.valueOf(int)` gives us a string containing a textual representation of the number we pass in, like so:

```
String returntext;
returntext = String.valueOf(returnval);
```

13. To output the `String`, we use the `setText(String)` method of the text field that we want to output to:

```
theWindow.TextField3.setText(returntext);
```

14. Putting it all together, we have:

```
private void Add()
{
     int returnval, add1, add2;
     String string1, string2, returntext;
     // Get first value and convert to a number
     string1 = theWindow.TextField1.getText();
     add1 = Integer.parseInt(string1);
     // Get second value and convert to a number
     string2 = theWindow.TextField2.getText();
     add2 = Integer.parseInt(string2);
     // Perform Addition and output result
     returnval = add1 + add2;
     returntext = String.valueOf(returnval);
     theWindow.TextField3.setText(returntext);
}
```

15. Of course, if we want to be more efficient, we can do it all in one statement:

```
private void Add()
{
     theWindow.TextField3.setText(String.valueOf(
          Integer.parseInt(theWindow.TextField1.getText()) +
          Integer.parseInt(theWindow.TextField2.getText())));
}
```

In either case, go ahead and compile and run the code to see how it works.

16. Congratulations, you now know how to read parameters, output results, and control when things happen!

The End