



Programmers Guide for TIE

TurnItEasier

João Lourenço



Index

Introduction	2
Organization of this Document	2
Basic Organization of the API	2
Tutorials/Explanation	3
Console	3
Dynamic Console	6
Window	6

Introduction

This project was created to allow people that are new to Java to be able to create their own programs easier without requiring a Medium level of knowledge in Java.

It contains basic functions to make write and reads easier on console as well as some more complex functions to help the programmer and avoid tons of work to create runnables or even frames.

I'm going to use Eclipse on windows for the tutorials.

Organization of this Document

This document will work as a tutorial and documentation for this API.

First I'm going to have tutorials where I'm going to explain how to create basic console programs and how to use them that will include examples, then the same thing for Dynamic Console and finally the windows.

After the tutorials there will be a section with the documentation for this API with all the functions on the API.

Basic Organization of the API

This API is organized by two main accessible classes with in total three Display Modes.

The Display Modes are Console, Dynamic Console and Window and they are used to define the way the program will run.

Console is the basic console that your Operating System as and where you can run Java applications.

Dynamic Console is a console that I created in order to give the programmer a console with a good platform to get keyboard listeners, mouse listeners, colors, fonts and so on.

Window, as the name by itself tells us, it is a window, in Java also called frame, where the programmer as full access to it but it makes easier to the programmer to create since with only one line of code it created.

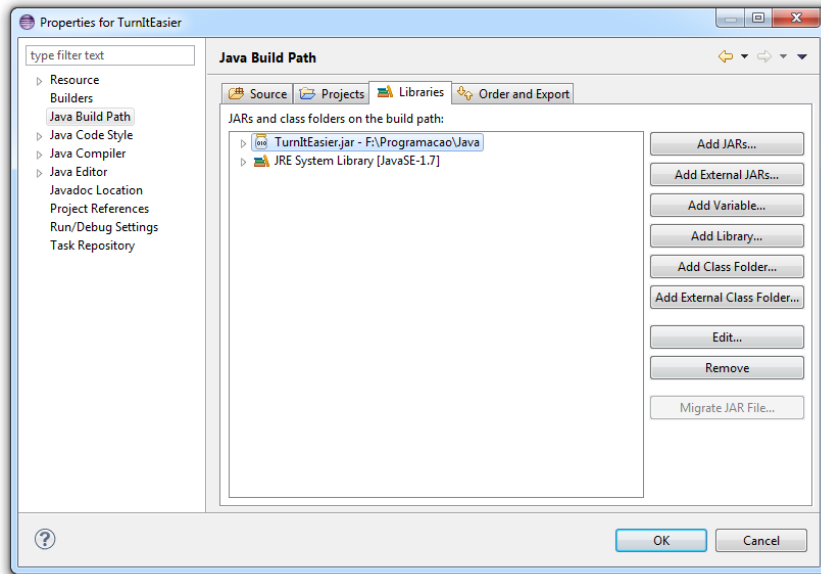
About the classes, the API as two main classes, one is TurnItEasier which contains the basic console commands, and the second is the Display class which contains Methods for the Dynamic Console and Window.

Tutorials/Explanation

Console

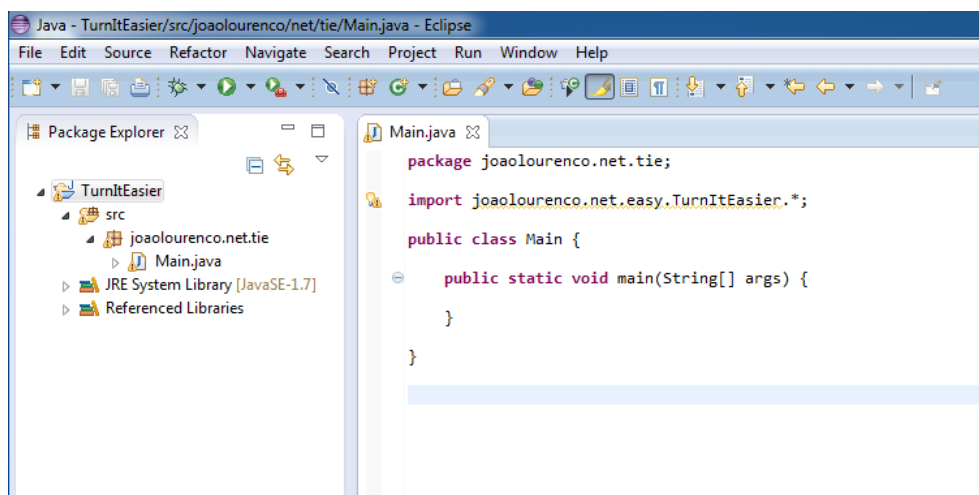
To startup we are going to use the easiest mode on the API, Console.

Create a new Java project at your own description and import the TurnItEasier.jar.



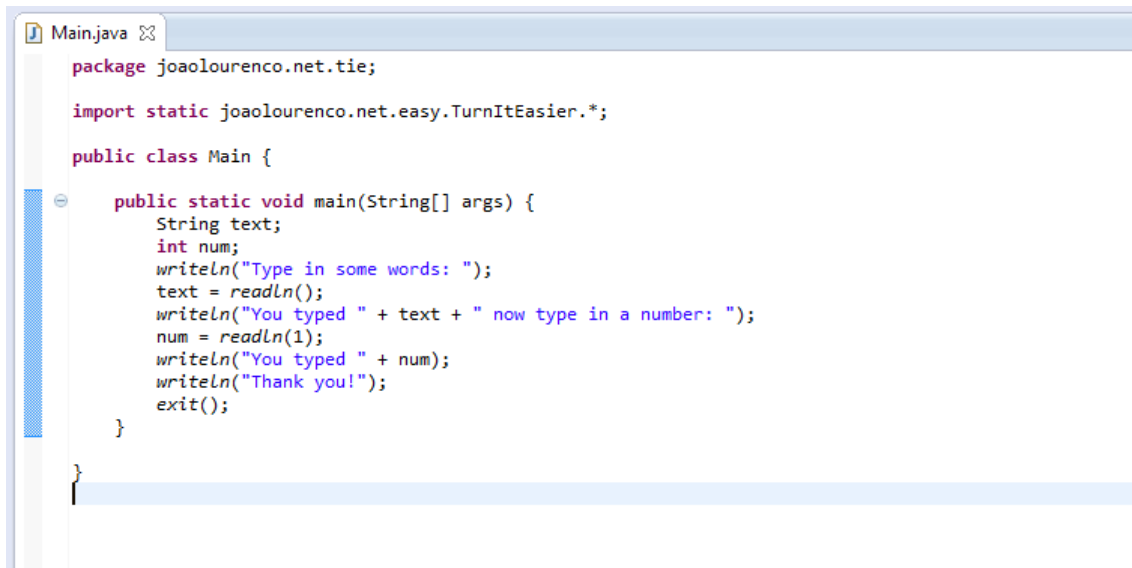
Right-Click on your project -> Properties -> Java Build Path -> Add External JAR's (find the TurnItEasier.jar) -> Open -> OK

Create your Main class and Method, and import as static the package `joaolourenco.net.easy.TurItEasier.*; .`



The import is used to give you access to the API.

Now let's start creating something with the API itself.



```

Main.java
package joaolourenco.net.tie;

import static joaolourenco.net.easy.TurnItEasier.*;

public class Main {

    public static void main(String[] args) {
        String text;
        int num;
        writeln("Type in some words: ");
        text = readln();
        writeln("You typed " + text + " now type in a number: ");
        num = readln(1);
        writeln("You typed " + num);
        writeln("Thank you!");
        exit();
    }
}

```

Explanation:

```
String text;
int num;
```

Here we started by creating two local variables, one as a String to keep some input text from the user and one integer to keep a number input from the user.

```
writeln("Type in some words: ");
text = readln();
```

The writeln is a method from the API that is going to print a new line to the console. The readln() is also a method by the TIE API that returns the user input. It is going to wait until the user types something.

```
writeln("You typed " + text + " now type in a number: ");
num = readln(1);
```

On this bit of code we are going to write in a new line like the text that we wrote and ask the user to write a number. This number is going to be stored on the num variable given by the readln(1). In this case we typed a 1 inside the brackets to inform the API that we want the read to be a number.

```
writeln("You typed " + num);
writeln("Thank you!");
exit();
```

To finish the program we tell the user what number was received thank him for the using our program and exit() the program. This exit() method is also from the API and it is recommended to use but not mandatory to use at the end, this will close the scanners and avoid some process to keep running after the application has been closed.

In this case we didn't create a `DisplayType` instead we used some console type Methods, this tells the API that we are going to use the default console as `DisplayType`, this is used to avoid people that don't know much of this yet to give more attention to the Java itself and less to setting up the environment.

On the console `DisplayType` there aren't much more to learn because of the low quality of the console given by the Java, and because of that we are going to have only one last example of code before moving into the next `DisplayType`, `DynamicConsole`.



```

package joaolourenco.net.tie;

import static joaolourenco.net.easy.TurnItEasier.*;

public class Main {

    public static void main(String[] args) {
        String user;
        String pw;
        write("User: ");
        user = readLn();
        write("Password: ");
        pw = readPW();
        if (user.equalsIgnoreCase("Joao") && pw.equalsIgnoreCase("123")) {
            writeln("5");
            delay(1000);
            writeln("4");
            delay(1000);
            writeln("3");
            delay(1000);
            writeln("2");
            delay(1000);
            writeln("1");
            delay(1000);
            writeln("Your in!");
        } else writeln("You have typed a wrong user/password.");
        exit();
    }
}

```

Explanation:

In this very basic example we created a simple login system password. If the user is logged in it is going to count from 5 to 1 until it says "You're in!", if you fail it is going to say that you have failed the login.

```

String user;
String pw;
write("User: ");
user = readLn();
write("Password: ");
pw = readPW();

```

This, as before is going to read the user and password and it is basically the same as it was before except some small changes: instead of using `writeln` we used `write` which is going to allow the user to type the user and password in the same line as the words "User:" and "Password:". And the last change we have done is instead of using `readln()` we used `readPW()` this will prevent the user to see the actual password on the console when its typed so if there is someone looking to the screen can't tell how many characters or what characters the password as.

```
deLay(1000);
```

The delay method is also from the TIE API and it is used to add a delay in miliseconds.

Attention: The `readPW()` function might throw an `ImpossibleActionException`, if it does it is because the console you are using does not have password support. (If running on eclipse to use the `readPW` you gave to export the project first and run it outside the eclipse)

Dynamic Console

Window

This is the best, most complex and the biggest area of this API, there is a vast amount of functions to make you able of doing anything you want.

My goal with this API wasn't forbid the users of it from using the normal Java and for that reason there are a ton of ways of doing certain things and there are ways to get the frame that the API is using. I'm not able to cover the all methods and functions available on the API so I'm going to cover only the main ones and expect the user will try and find out.