Tinkerforge GUI – Beginner Tutorial

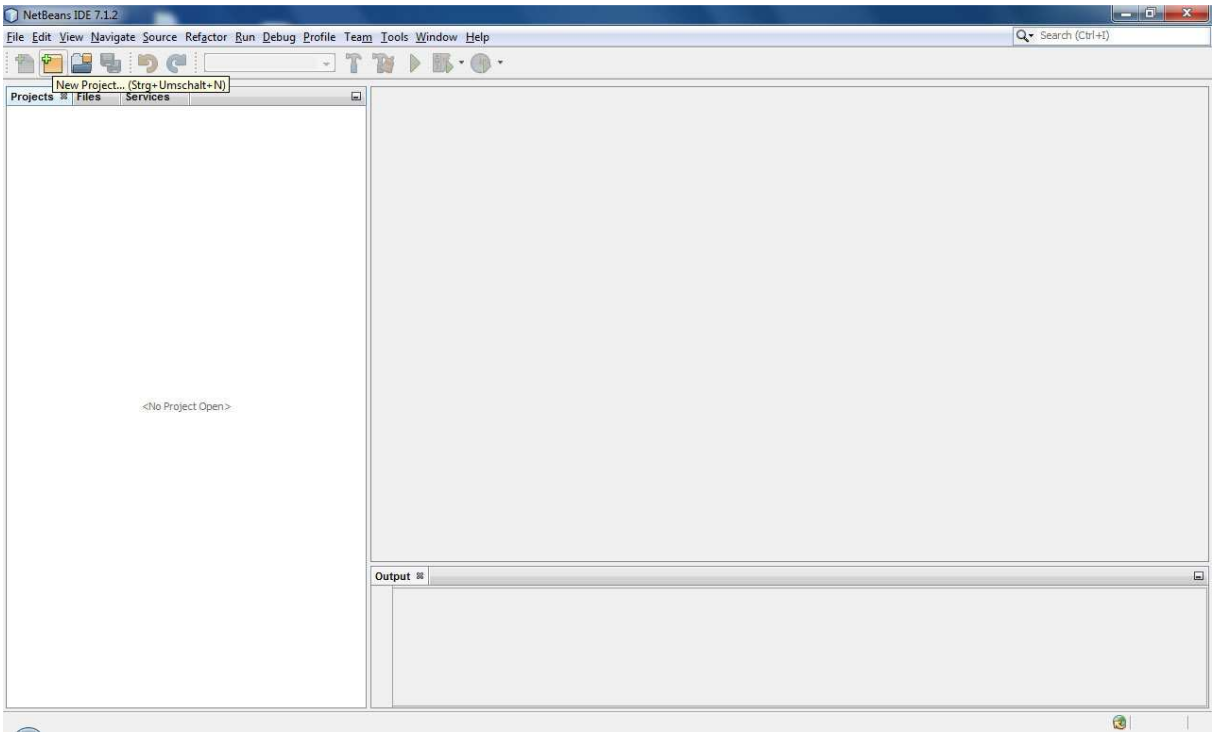
By M4ST3R

Voraussetzungen:

- Netbeans (7.1 aber funktioniert mit allen anderen auch) von Oracle installiert www.netbeans.org
- Java Bindings von TF runtergeladen
- BrickD installiert
- Masterbrick und JoystickBricklet vorhanden

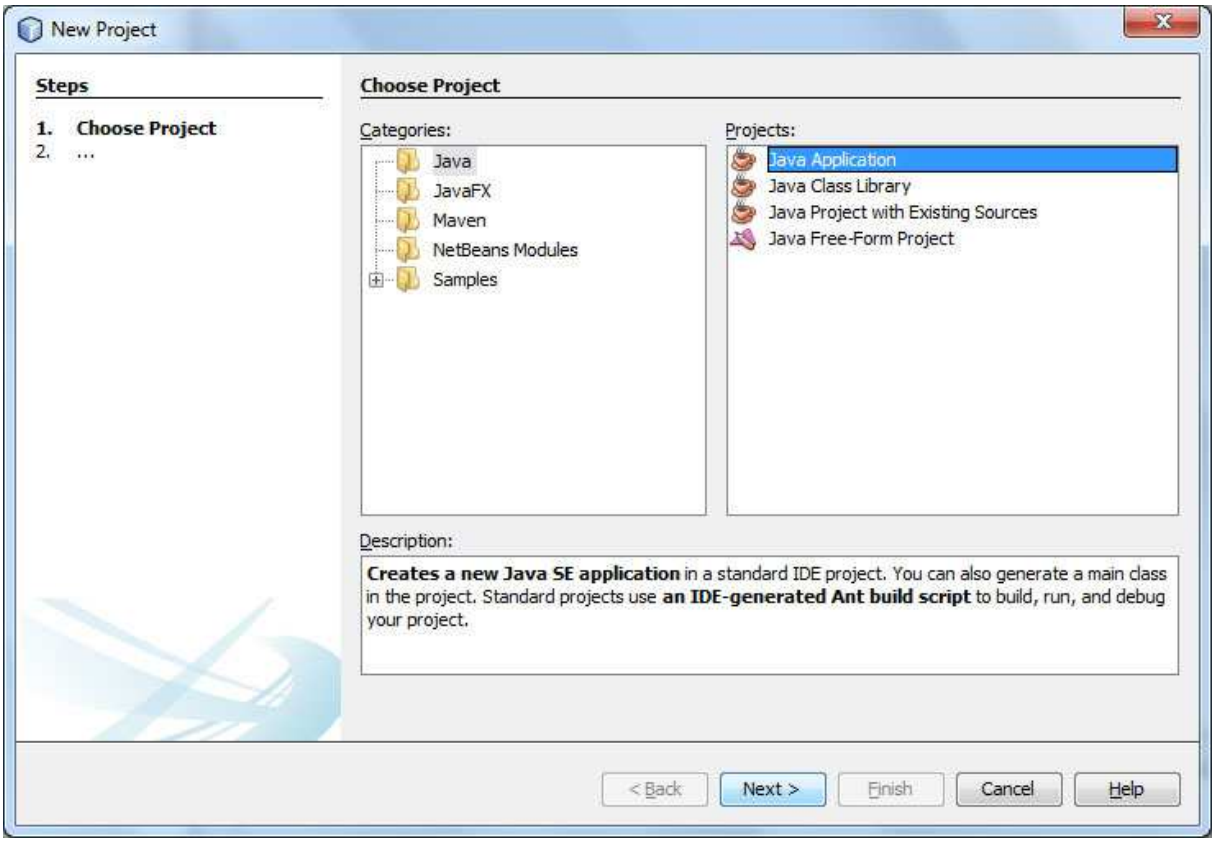
Schritt 1)

Als erstes starten wir Netbeans und sollten folgende Ansicht bekommen:

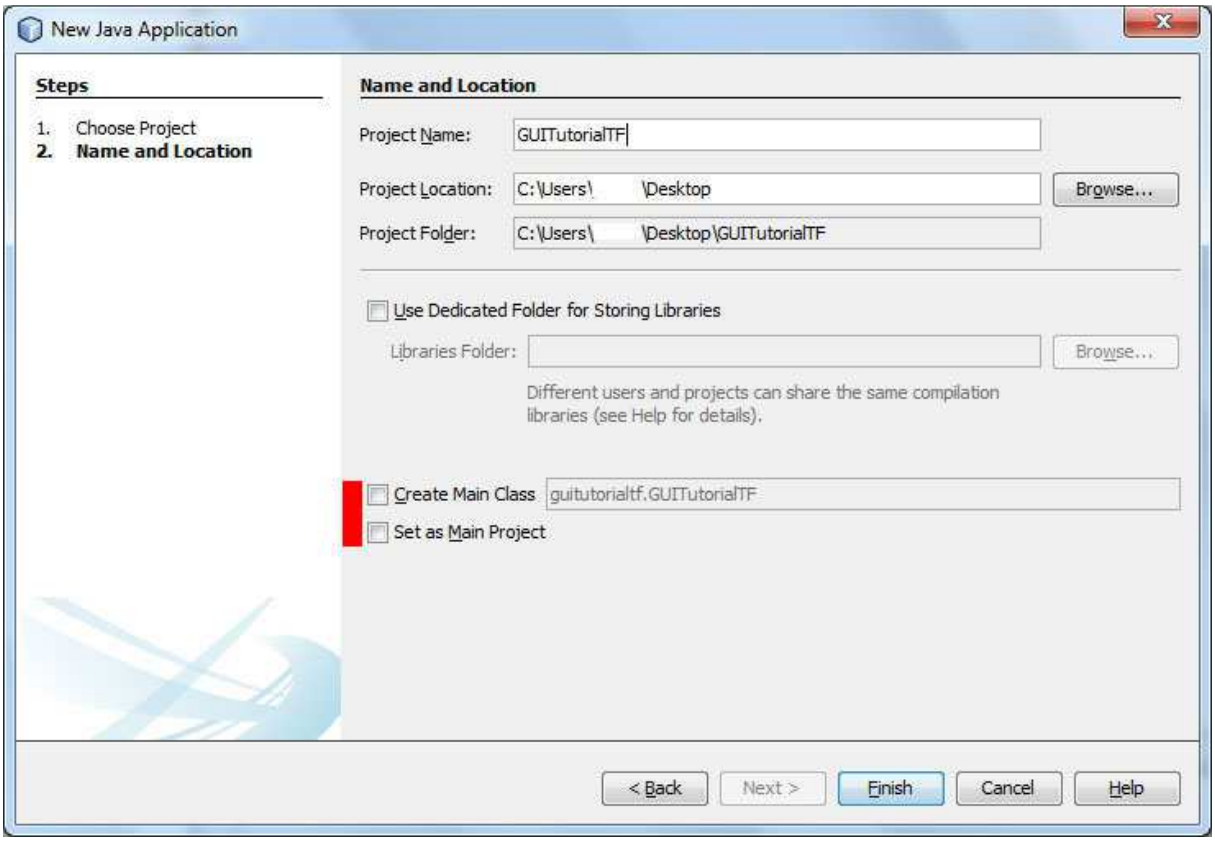


Um ein neues Projekt zu öffnen, klickt man auf das entsprechende Icon (siehe Bild)

Dann sollte sich folgender Dialog öffnen:



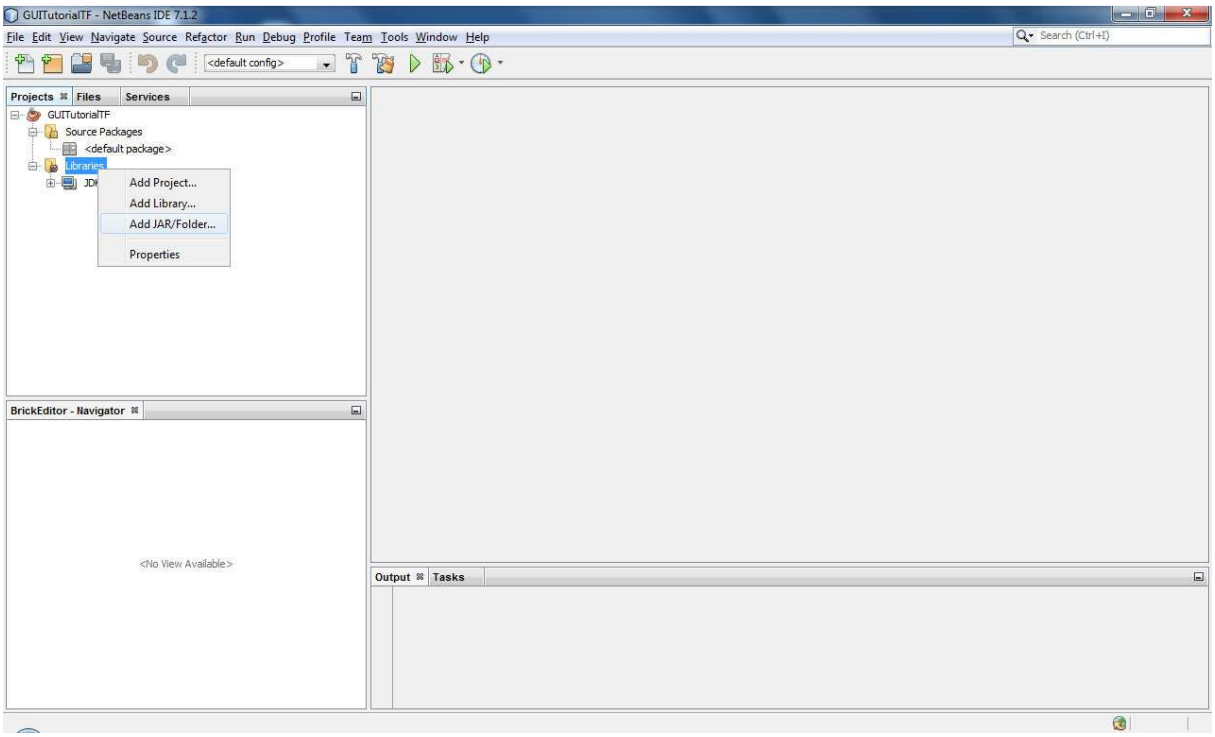
Man wählt in diesem Fall, da wir nur ein einfaches Testprojekt erstellen wollen, Java Application aus.



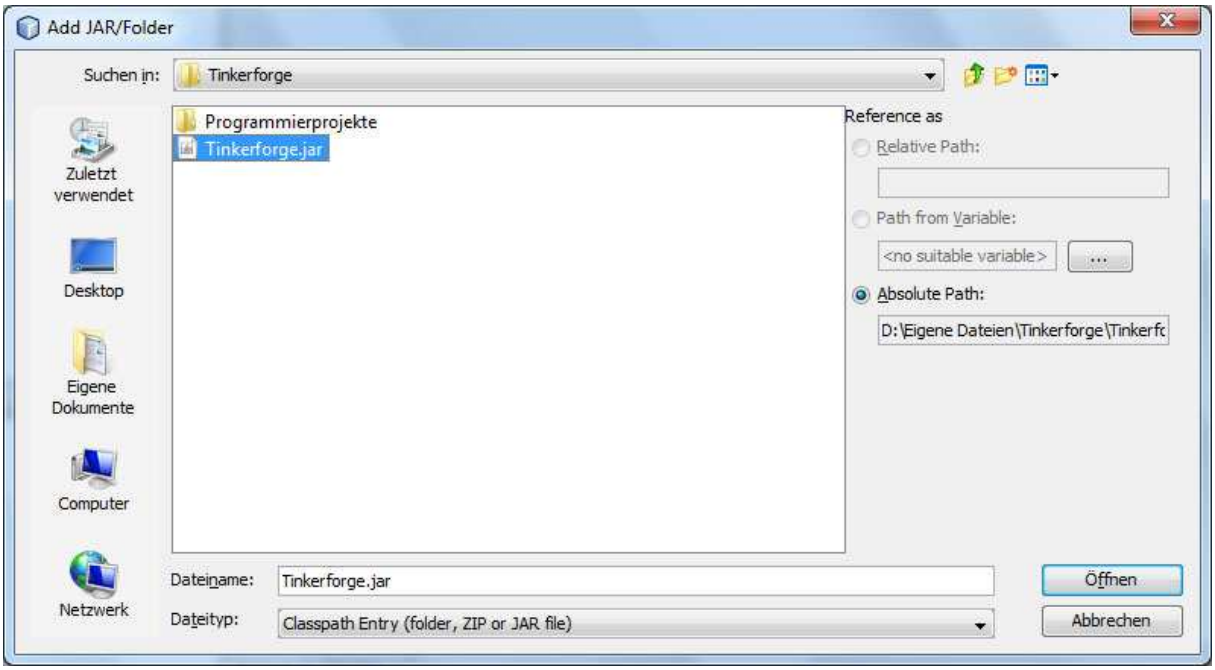
Nun müssen wir dem Projekt noch einen Namen geben. Dieser kann frei gewählt werden. Ebenso der Speicherort kann frei gewählt werden. Auf jeden Fall sollte man die beiden Haken unten entfernen! Anschließend auf Finish klicken.

Schritt 2

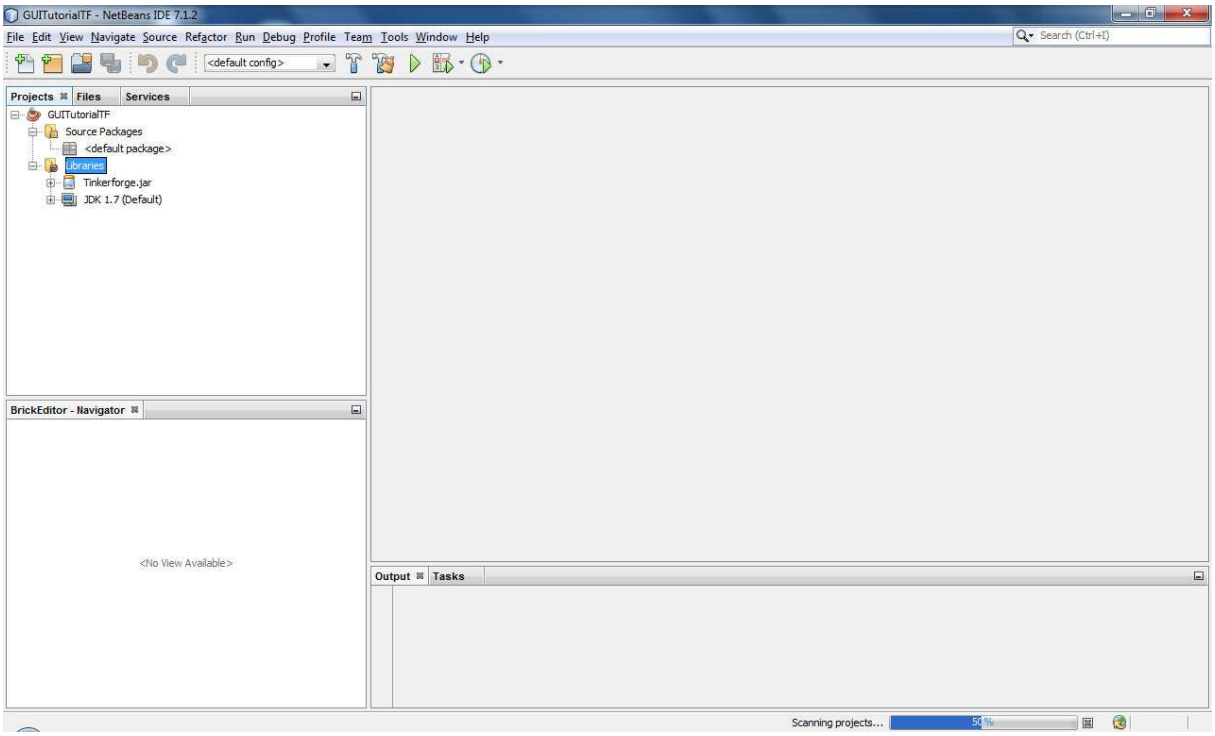
Als nächstes müssen wir die von TF bereitgestellten Bindings hinzufügen, das geht folgendermaßen.



Wir klicken mit der rechten Maustaste auf „Libraries“ und anschließend auf Add Jar/Folder.



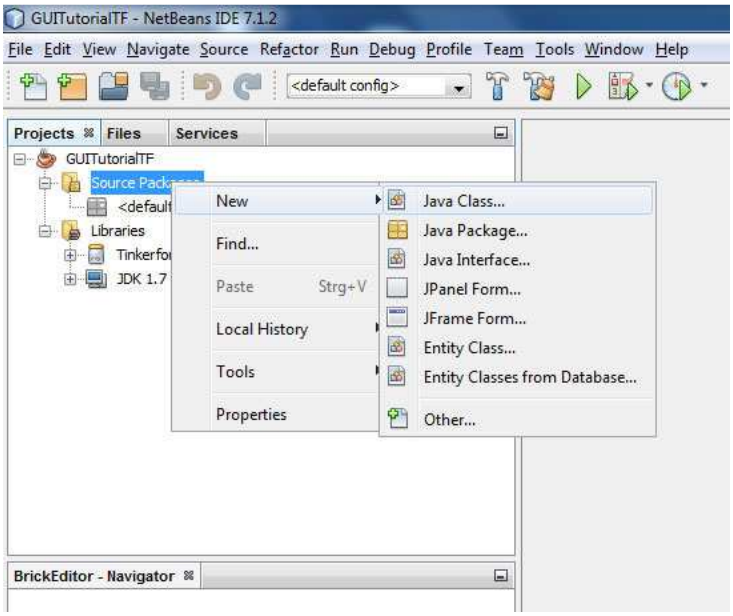
Jetzt wählen wir die Tinkerforge.jar aus und klicken auf öffnen.



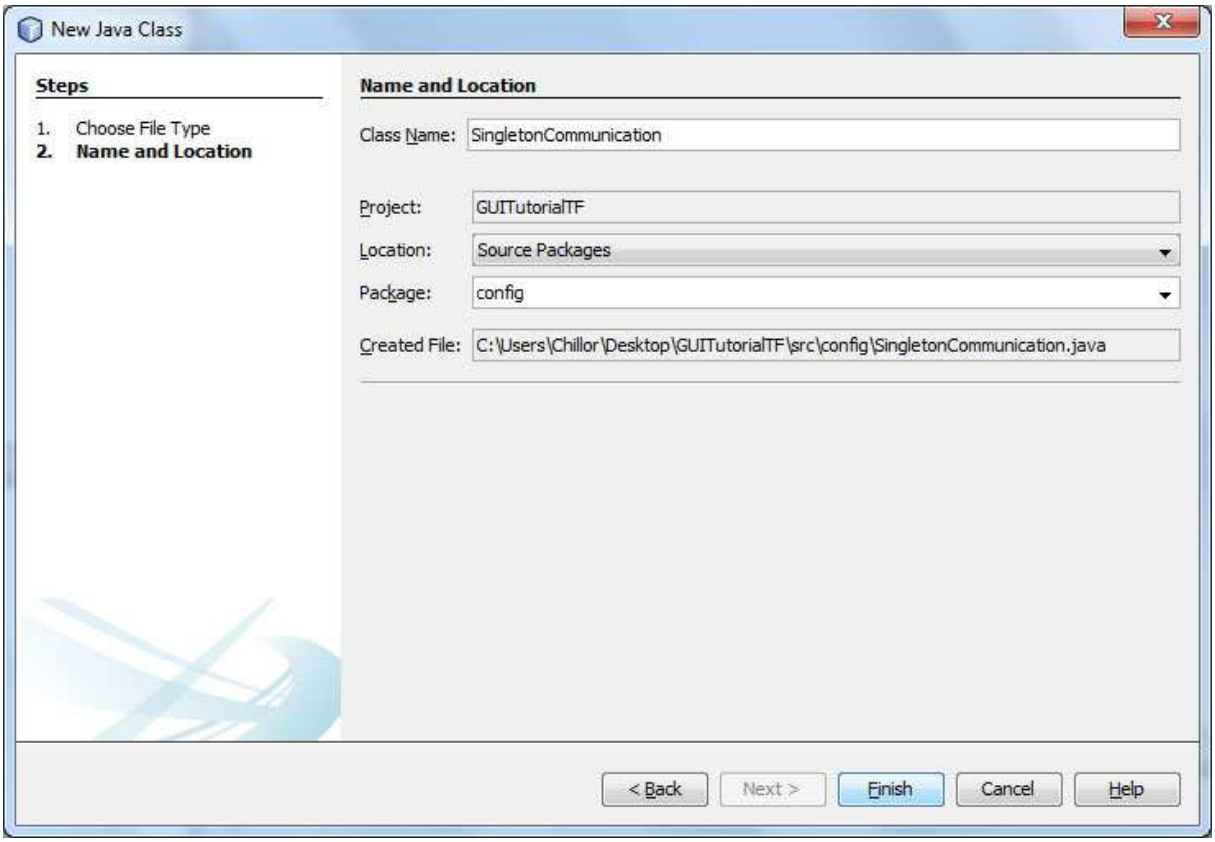
Wenn alles gut gegangen ist sollte unser Projekt jetzt so aussehen.

Schritt 3)

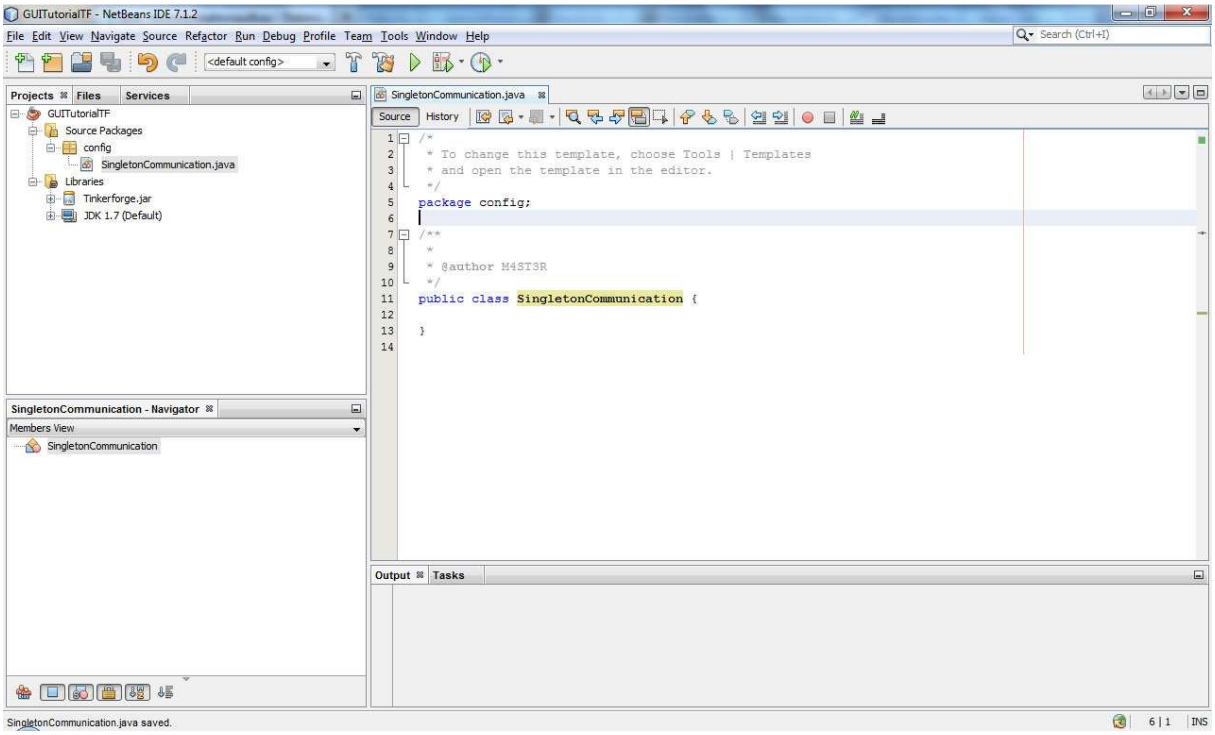
Unsere erste eigene Klasse. Dies wird zudem eine spezielle Klasse, da sie die Verbindung zu den Bricks aufbauen wird.



Dazu klicken wir mit der rechten Maustaste auf den Ordner „Source Package“ und wählen Neu→ Java Class aus.



Der Klasse geben wir den Namen „singletonCommunication“ (habe leider hier das „S“ großgeschrieben, aber ihr solltet es klein schreiben, da es im Wiki auch klein ist). Als Package habe ich „config“ angegeben.



So sieht unsere leere Klasse aus. Wir könnten das Programm jetzt starten, aber das würde nichts bringen, weil wir nicht sehen und noch keine „Hauptklasse“ benannt haben.

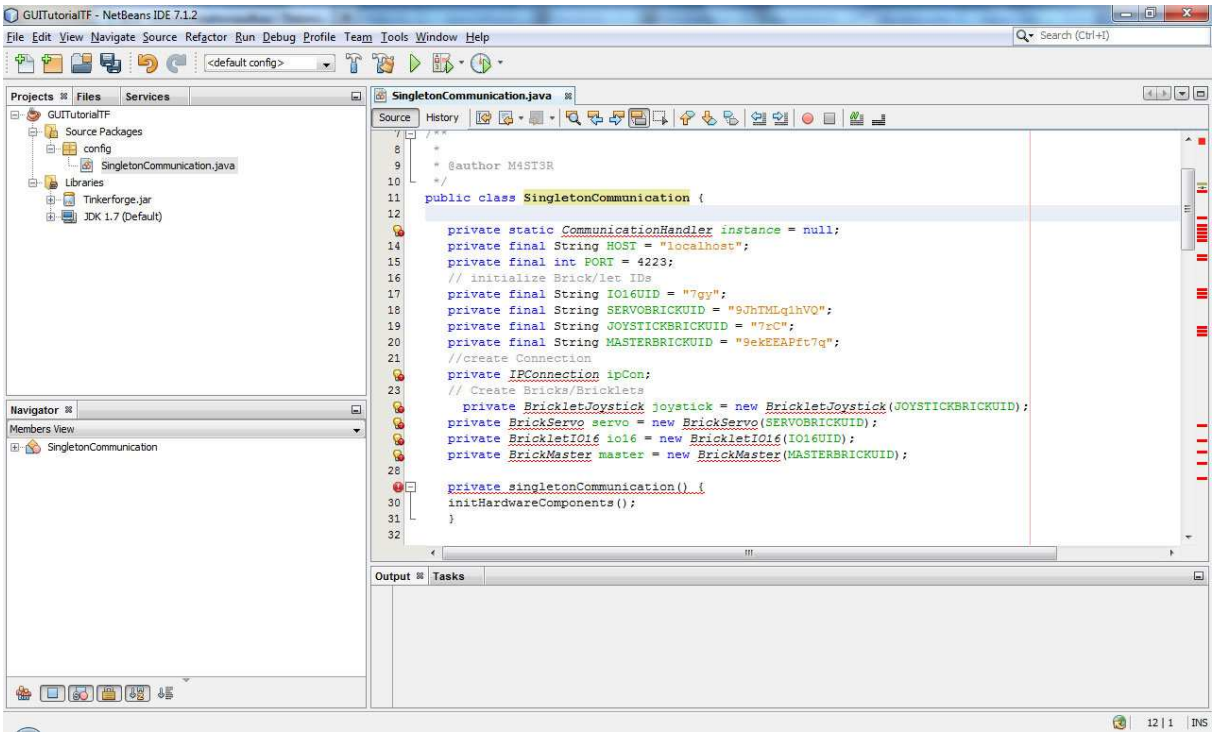
Der Quellcode für die Klasse ist im Wiki Dokumentiert. Ich kopiere ihn für euch aber auch hier rein.

Denkt daran, dass ihr die IDs der Bricks auf eure anpasst. Alle Bricks bzw. Bricklets, die wir nicht benötigen, was so ziemlich alle sind außer das Joystickbricklet und das Masterbrick, könnt ihr aus dem Code entfernen bzw. auskommentieren.

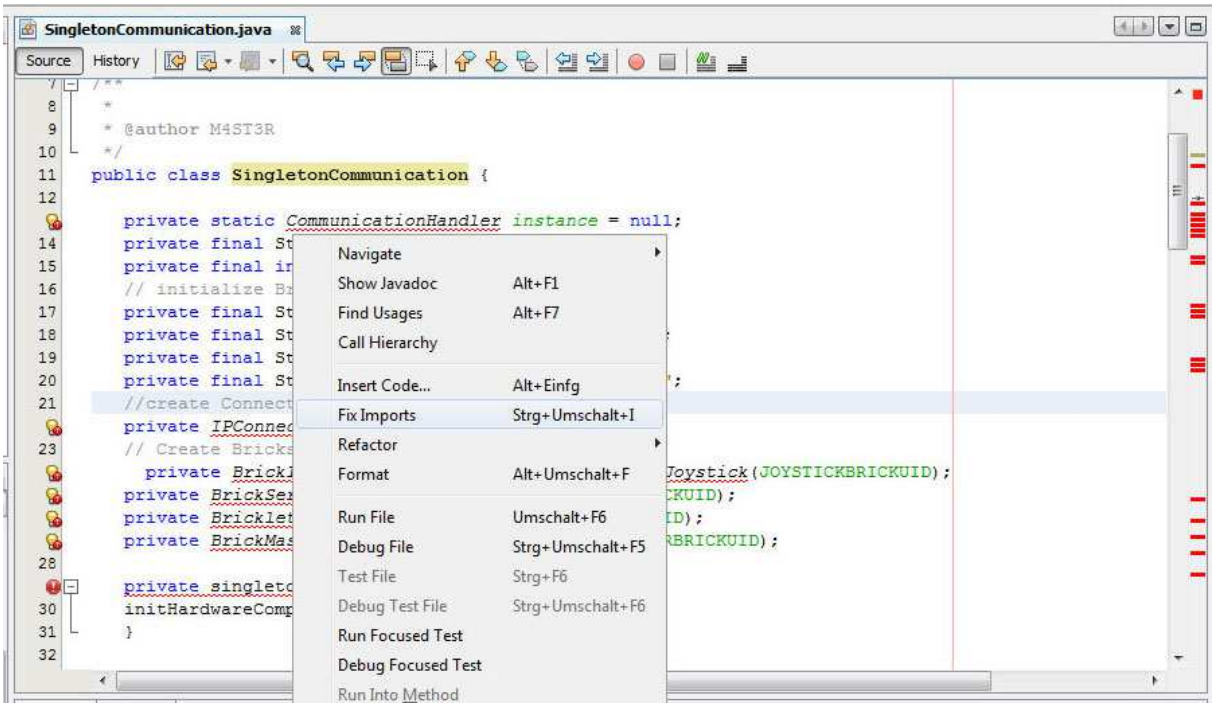
```
public class singletonCommunication {
  private static singletonCommunication instance = null;
  private final String HOST = "localhost";
  private final int PORT = 4223;
  // initialize Brick/let IDs
//   private final String IO16UID = "7gy";
//   private final String SERVOBRICKUID = "9JhTMLq1hVQ";
  private final String JOYSTICKBRICKUID = "7rC";
  private final String MASTERBRICKUID = "9ekEEAPft7q";
  //create Connection
  private IPConnection ipCon;
  // Create Bricks/Bricklets
    private BrickletJoystick joystick = new BrickletJoystick(JOYSTICKBRICKUID);
//   private BrickServo servo = new BrickServo(SERVOBRICKUID);
//   private BrickletIO16 io16 = new BrickletIO16(IO16UID);
  private BrickMaster master = new BrickMaster(MASTERBRICKUID);

  private singletonCommunication() {
  initHardwareComponents();
  }
  public synchronized static singletonCommunication getInstance() {
     if (instance == null) {
        instance = new singletonCommunication();
     }
     return instance;
  }
  private void initHardwareComponents() {
     // Create new Connection with host and port
     try {
        ipCon = new IPConnection(HOST, PORT);
     } catch (IOException ex) {
        Logger.getLogger(singletonCommunication.class.getName()).log(Level.SEVERE, null, ex);
        System.out.println("Hardware initialization IO error in " +
singletonCommunication.class.getName());
     }
     // add devices to the connection
     try {
        ipCon.addDevice(master);
//         ipCon.addDevice(servo);
//         ipCon.addDevice(io16);
        ipCon.addDevice(joystick);
```

```
      } catch (Exception ex) {
         Logger.getLogger(singletonCommunication.class.getName()).log(Level.SEVERE, null, ex);
         System.out.println("Hardware initialization timeout error in " +
singletonCommunication.class.getName());
      }
  }
  public String getHOST() {
     return HOST;
  }
  public String getJOYSTICKBRICKUID() {
     return JOYSTICKBRICKUID;
  }
  public String getMASTERBRICKUID() {
     return MASTERBRICKUID;
  }
  public int getPORT() {
     return PORT;
  }
  public IPConnection getIpCon() {
     return ipCon;
  }
  public BrickletJoystick getJoystick() {
     return joystick;
  }
  public BrickMaster getMaster() {
     return master;
  }
}
```

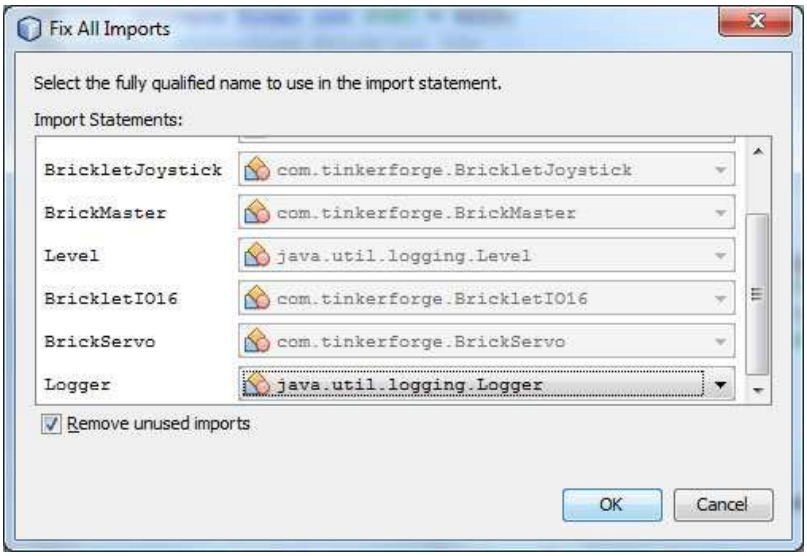
Einfach alles kopieren und einfügen. Nicht wundern es gibt einige Fehlermeldungen, aber das ist erst mal ok!
Mit der rechten Maustaste könnt ihr euren Code formatieren lassen. Das ist ganz hilfreich für die Übersicht.



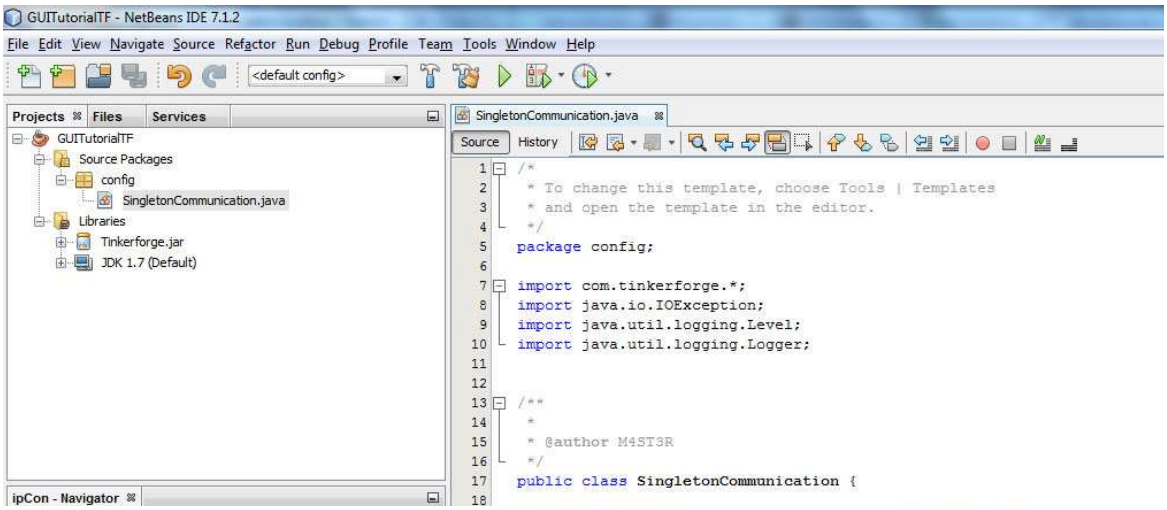
So sollte das Ganze aussehen.



Mit der rechten Maustaste ins Feld klicken und dann einfach „Fix Imports“ anklicken. Dann erscheint folgendes Fenster:



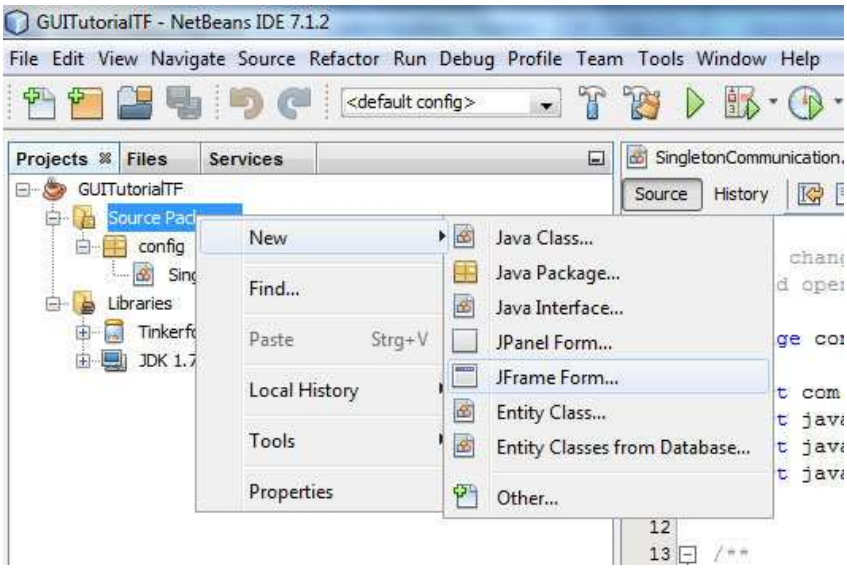
Das könnt ihr einfach bestätigen. Danach die Klasse speichern und die Fehlermeldungen sind weg. Was ist hier passiert? Die Netbeans IDE hat an den Anfang der Klasse Imports geschrieben:



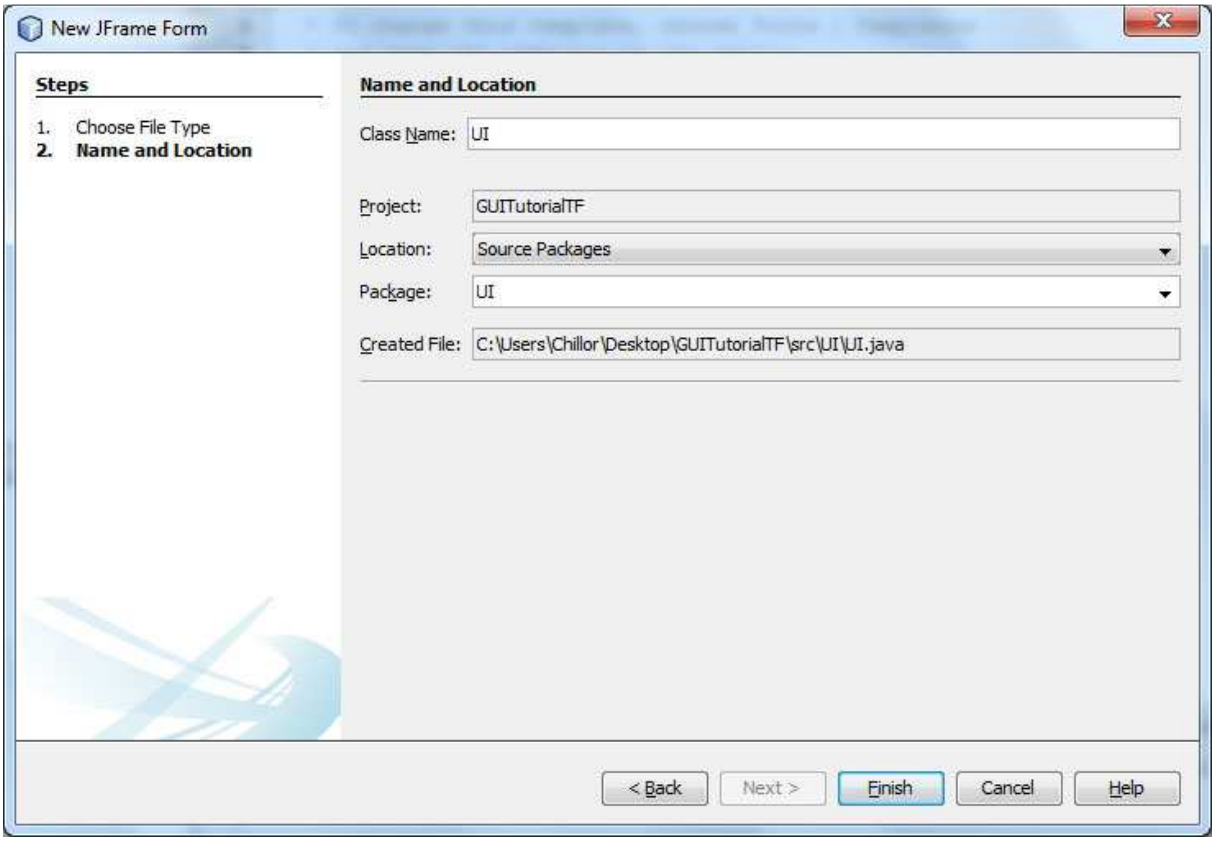
Dadurch werden die Fehler behoben.

Schritt 4)

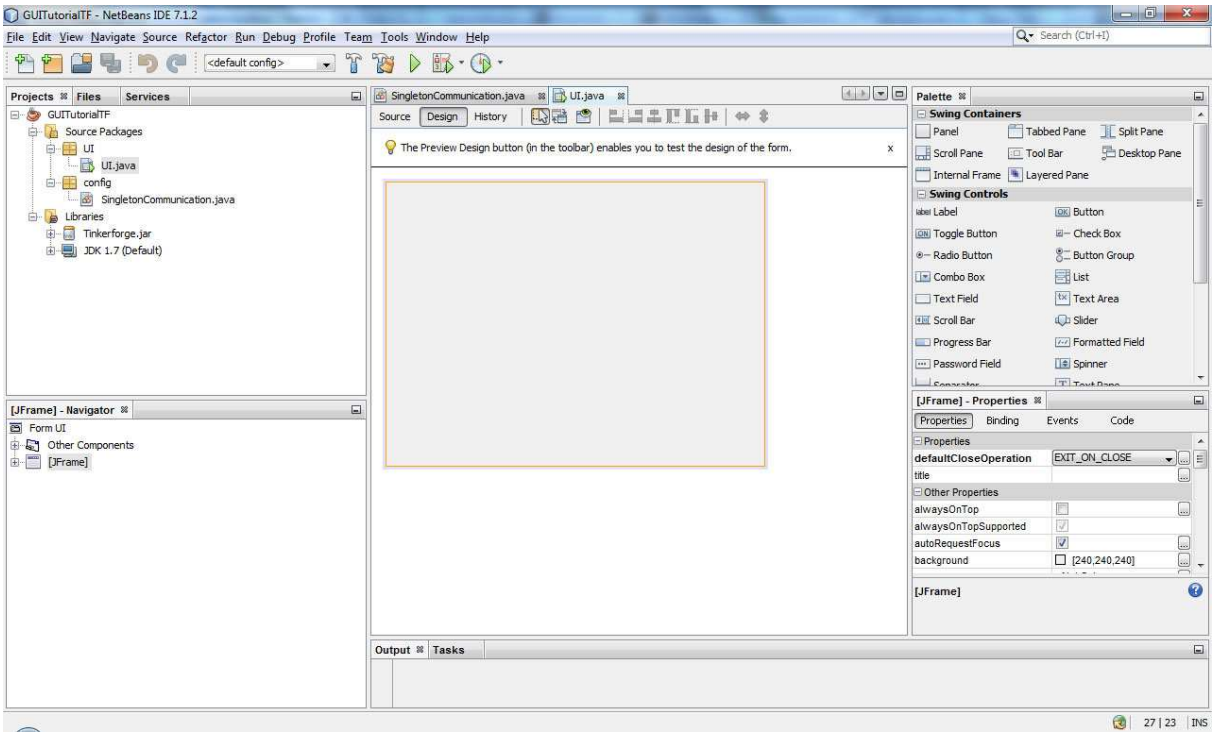
Als nächstes werden wir die Oberfläche gestalten. Dies wird für das Tutorial hier nur eine sehr einfache sein, um die Funktionsweise zu erklären.



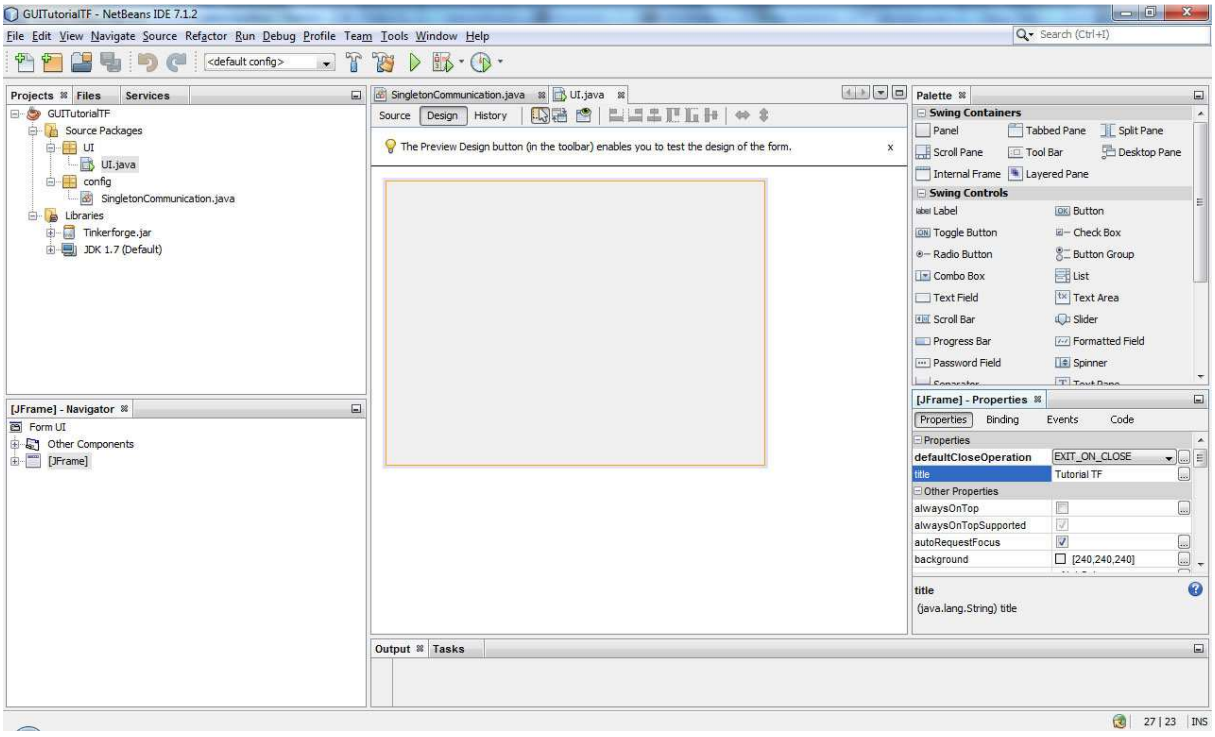
Dazu klicken wir wieder auf „Source …“ und wählen New→ JFrame Form



Nennen die Klasse UI und packen sie in das Package „UI“. Mit Finish wird alles erzeugt

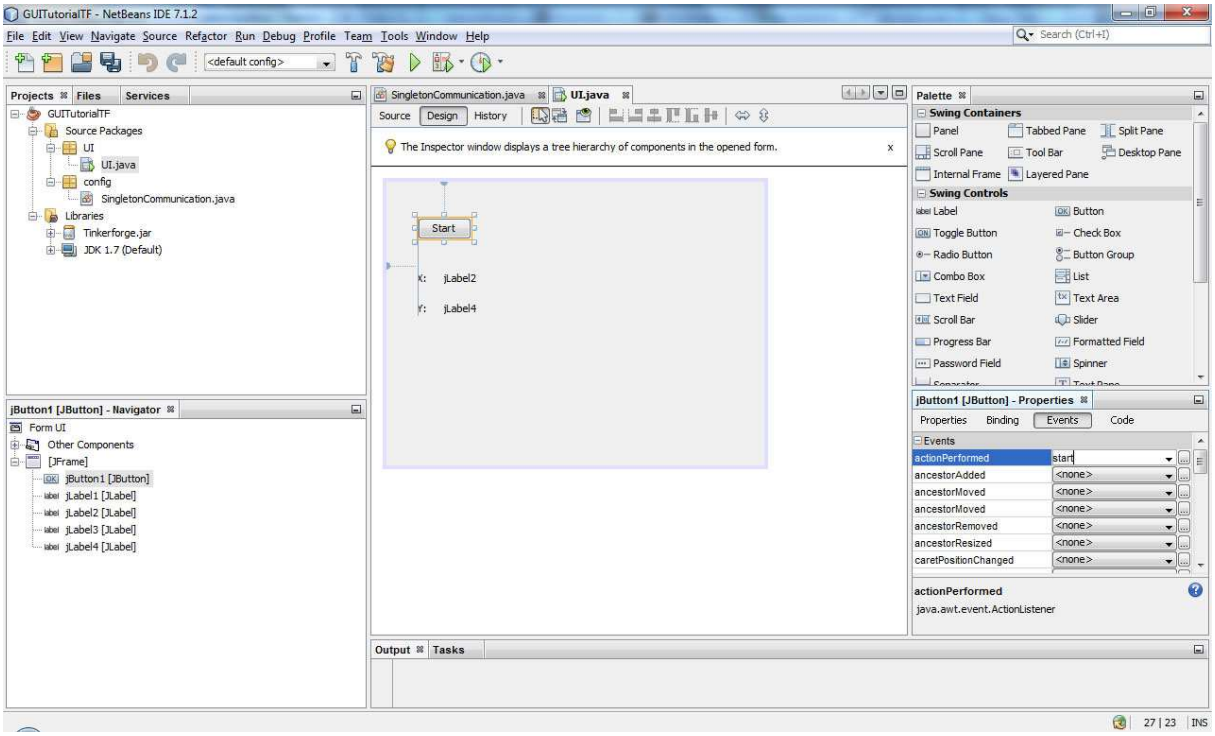


Ihr habt gemerkt, das Netbeans in einen anderen Modus gesprungen ist. Da wir nun Oberflächen bearbeiten möchten, werden entsprechende Bilder rechts am Rand angezeigt, aber dazu kommen wir gleich.

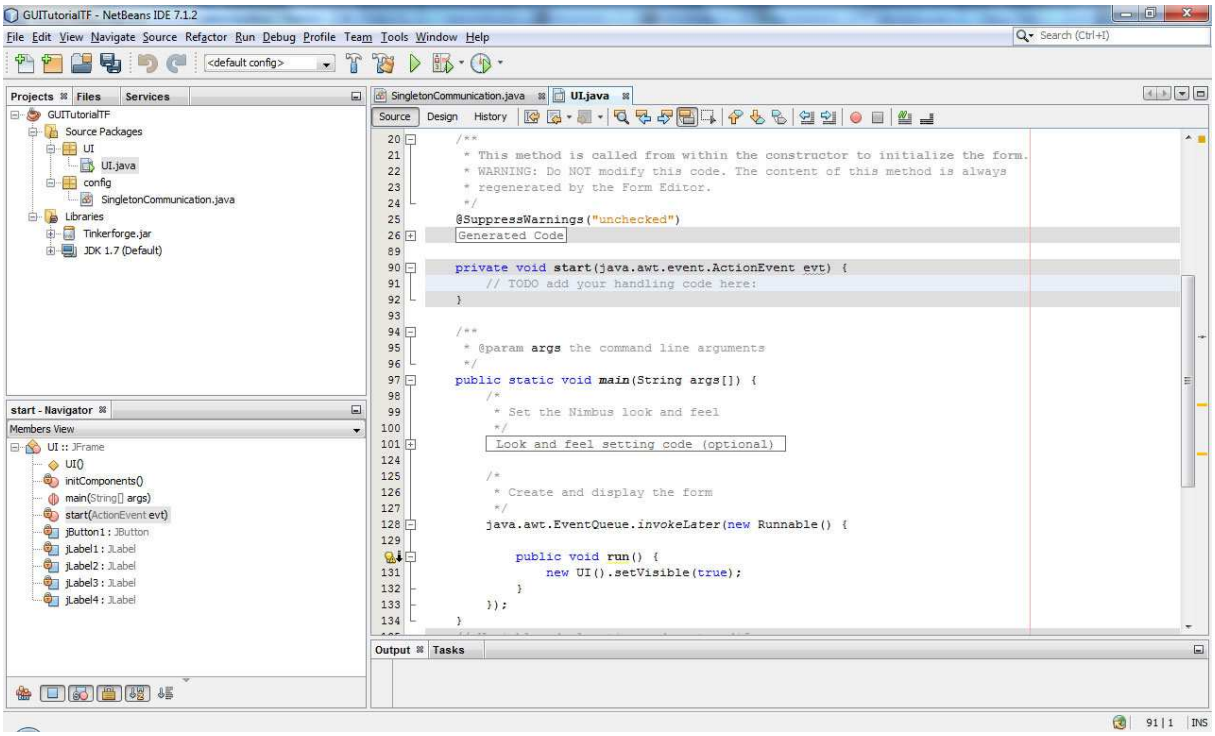


Als erstes ändern wir den Namen unserer Applikation. Dazu klicken wir Links auf JFrame und könnten dann rechts unter den Properties den Title ändern. Der Name ist dabei völlig egal.

Jetzt können wir endlich unseren Button in die GUI ziehen. Dazu können wir einfach per Drag and Drop die einzelnen Elemente auf die Oberfläche ziehen. Zusätzlich ziehen wir noch 4 Labels auf die Oberfläche in die wir unsere Werte gleich schreiben wollen. Mit der rechten Maustaste auf den Steuerelementen, kann man den angezeigten Text verändern. Den Button nennen wir Start und die jeweils linken Labels (siehe folgendes Bild) nennen wir X: bzw. Y:



So sollte unsere Oberfläche jetzt aussehen. Um dem Button nun eine Funktion hinzuzufügen, klicken wir diesen einmal an und wählen rechts „Events“ aus. Bei ActionPerformed (also Button geklickt) schreiben wir Start rein und drücken Enter.



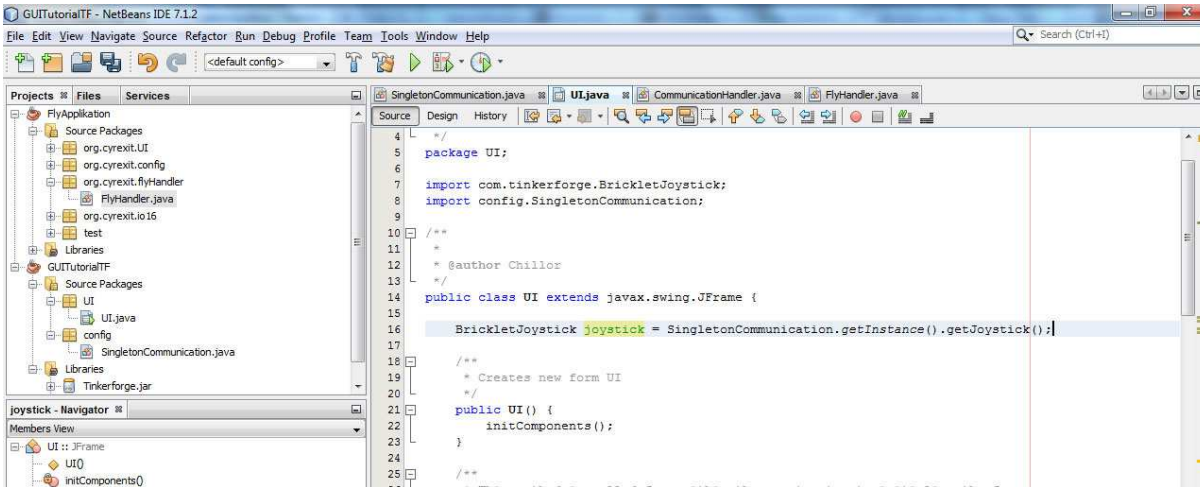
Automatisch wird im Quelltext eine neue Funktion hinzugefügt für unseren Button. Dafür müssen wir ansonsten nichts machen. Nun wollen wir, wenn der Button geklickt wird, dem Joystickbricklet einen Listener hinzufügen. Das sieht folgendermaßen aus:

```
private void start(java.awt.event.ActionEvent evt) {

    joystick.setPositionCallbackPeriod(50);
    joystick.calibrate();
    joystick.setDebouncePeriod(200);
    joystick.addListener(new BrickletJoystick.PositionListener() {

        @Override
        public void position(short x, short y) {
            jLabel2.setText(String.valueOf(x));
            jLabel4.setText(String.valueOf(y));
        }
    });



    // TODO add your handling code here:
}
```

```
joystick.setPositionCallbackPeriod(50);
joystick.calibrate();
joystick.setDebouncePeriod(200);
joystick.addListener(new BrickletJoystick.PositionListener() {

   @Override
```

```
    public void position(short x, short y) {
        jLabel2.setText(String.valueOf(x));
        jLabel4.setText(String.valueOf(y));
    }
});
```

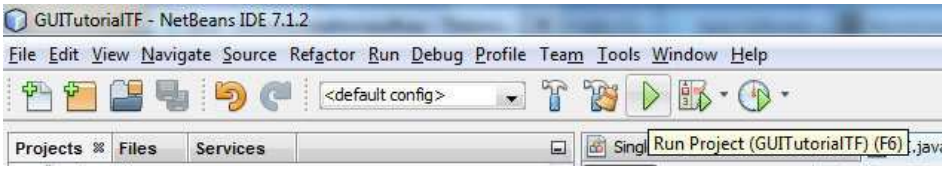
Da wir die Variable joystick noch nicht definiert haben, bekommen wir eine Fehlermeldung. Deshalb schreiben wir nun folgendes:



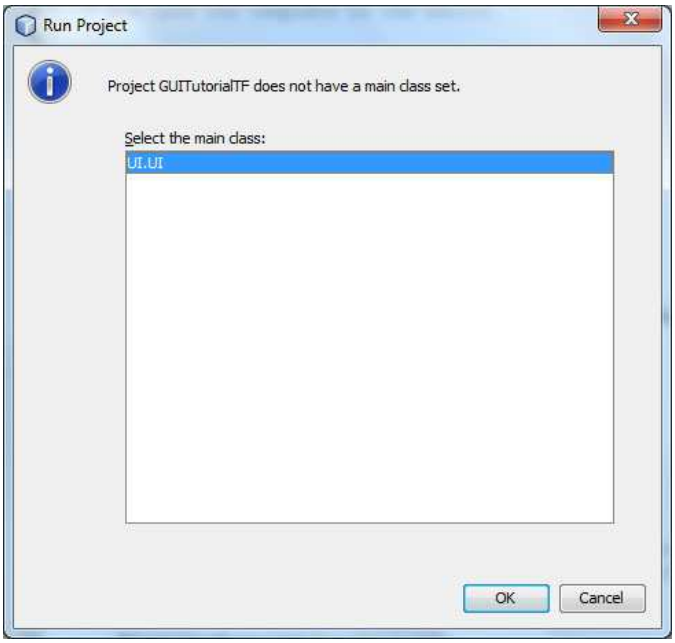
BrickletJoystick joystick = singletonCommunication.getInstance().getJoystick();

Damit sollten nun keine Fehler mehr angezeigt werden. Für die Leute die das jetzt selbst tippen ein kleiner Tipp:
Wenn ihr „Brick“ eingetippt habt drückt mal strg+leertaste dann werden euch automatisch alle Möglichkeiten angezeigt und ihr könnt es einfach mit enter bestätigen. Dadurch spart man sich mit der Zeit viel Schreibarbeit.

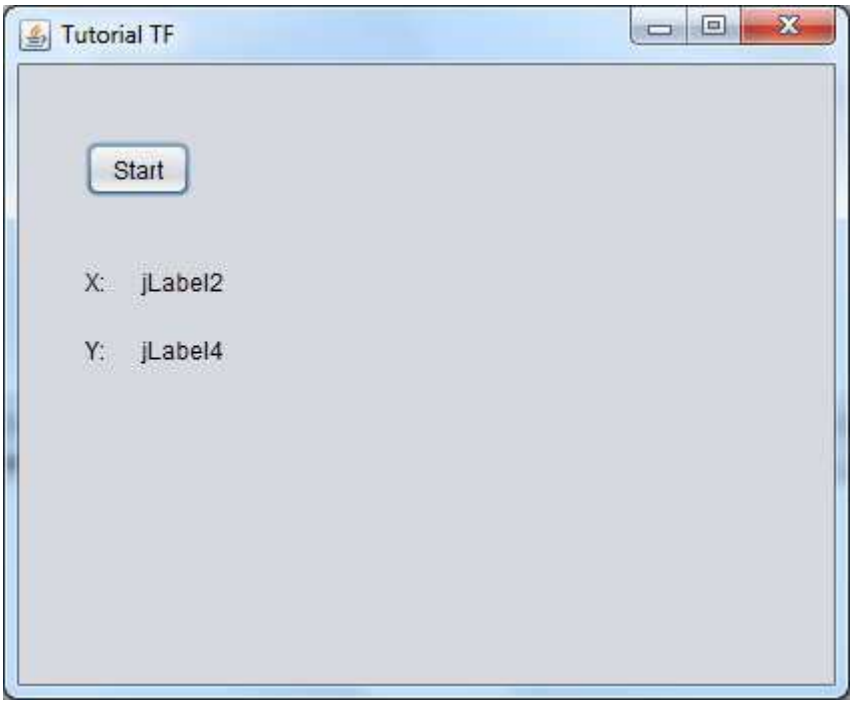
Nun ist unser Programm eigentlich schon fertig.



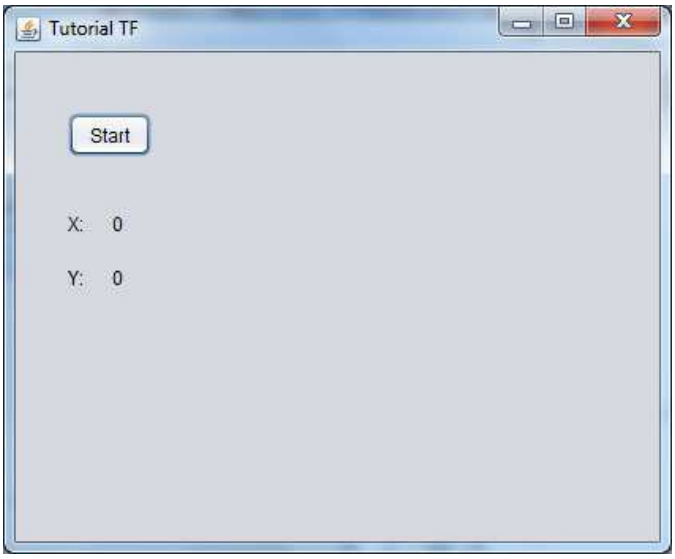
Wir klicken jetzt auf den grünen Pfeil (oder drücken F6) um das Programm zu starten.
WICHTIG: die Bricks/lets müssen verbunden sein mit dem Rechner!!



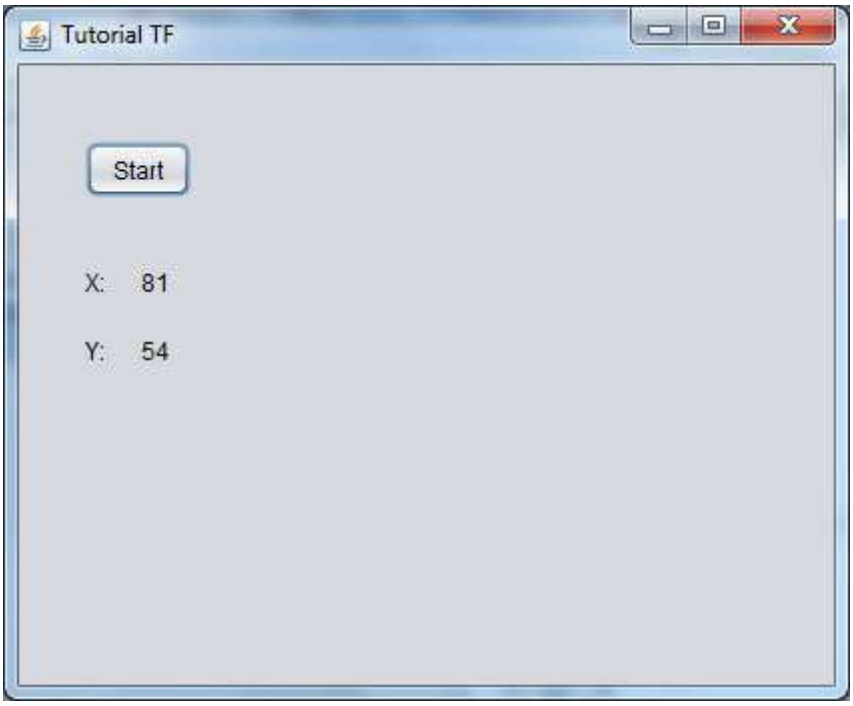
Da wir noch keine Hauptklasse definiert haben, werden wir nun aufgefordert das zu tun. Einfach ok klicken.



Unser Programm sollte nun so aussehen.
Sobald wir auf Start klicken:



Wenn man jetzt den Joystick bewegt werden die Werte in die entsprechenden Labels geschrieben



So ich hoffe ihr habt ein wenig was gelernt! Wenn Fragen auftreten einfach stellen!