

Alice with a cup of Java

What can I do with Alice

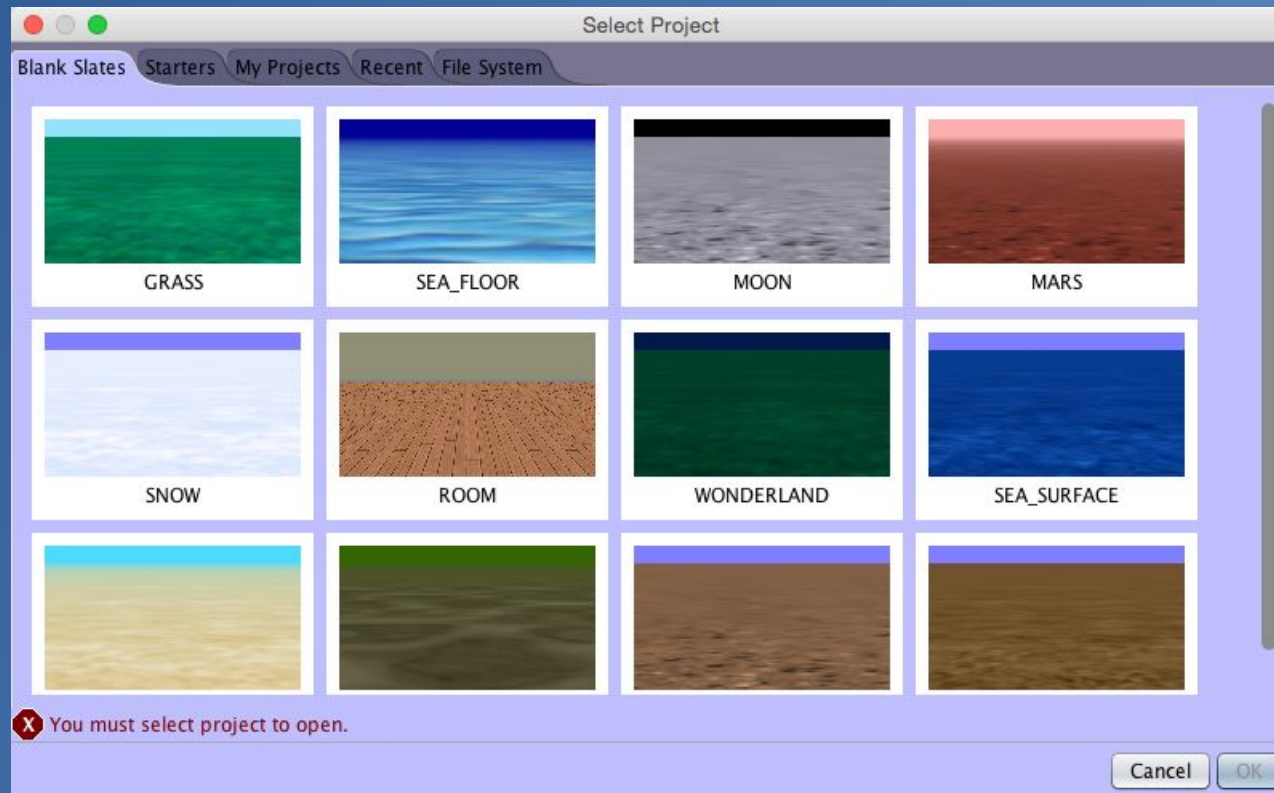
Lets get warmed up

Make sure *Alice* is installed.

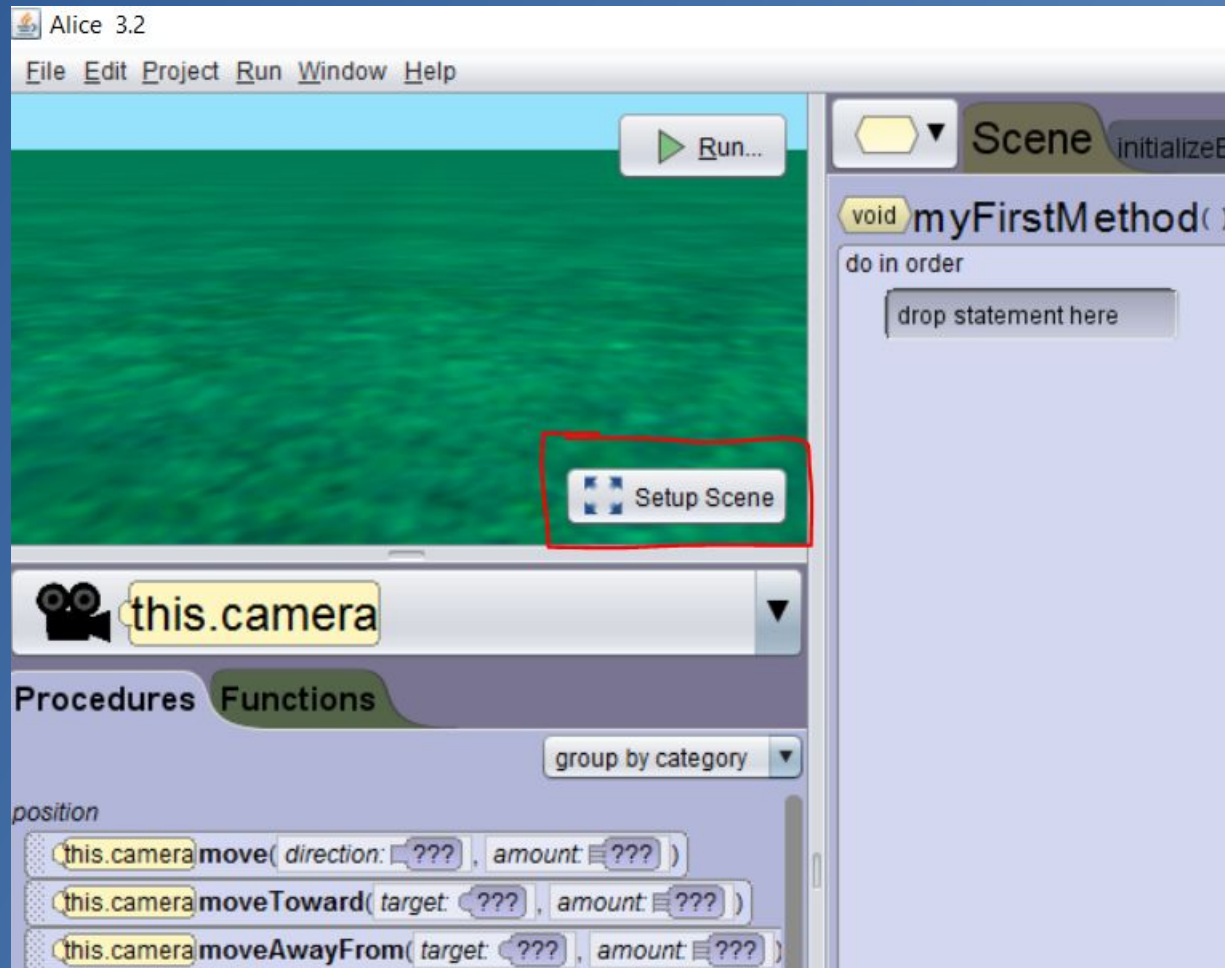
# Start Alice



# Pick a Landscape



# Open Up a scene for editing

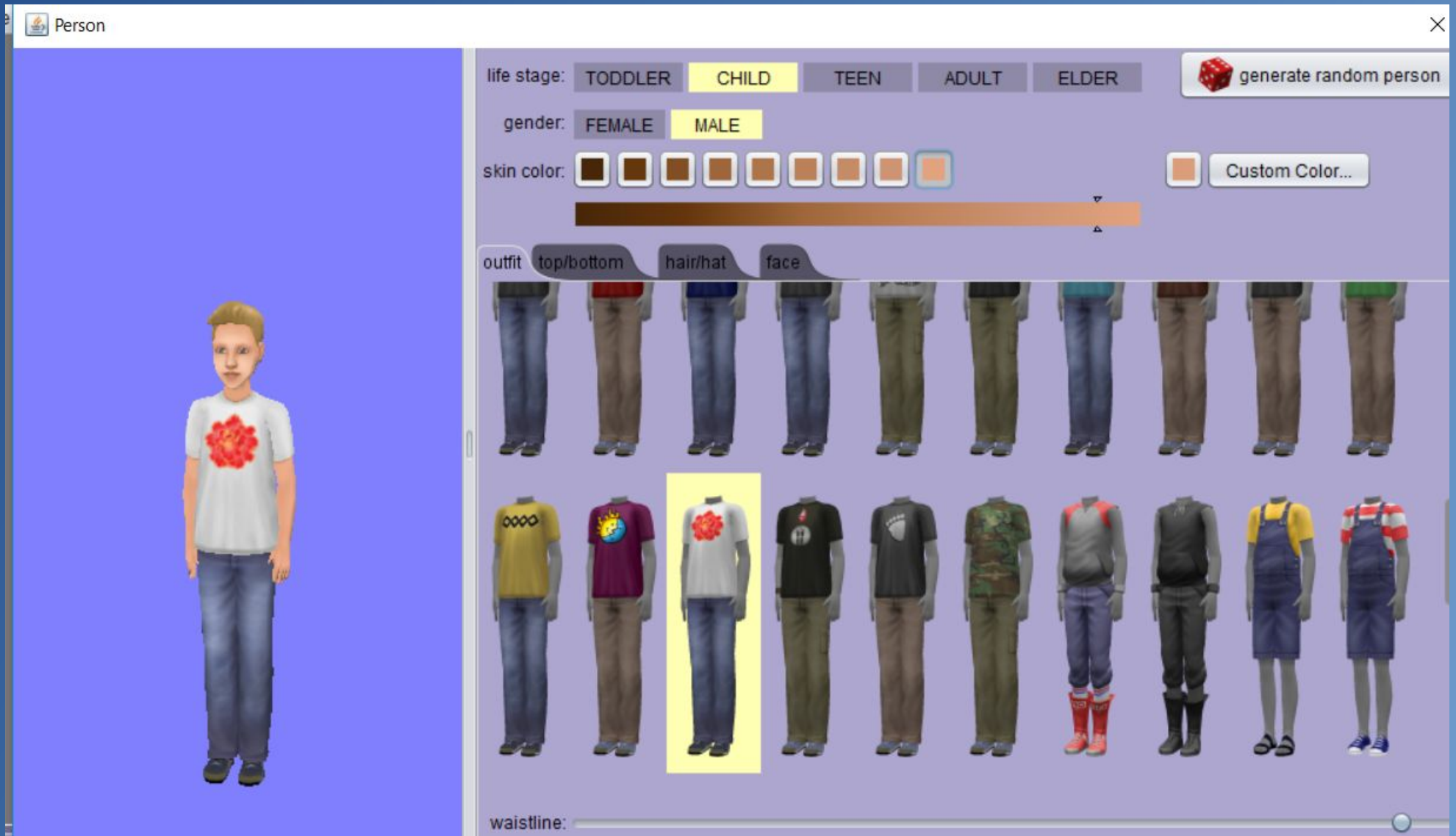


# Pick your avatar

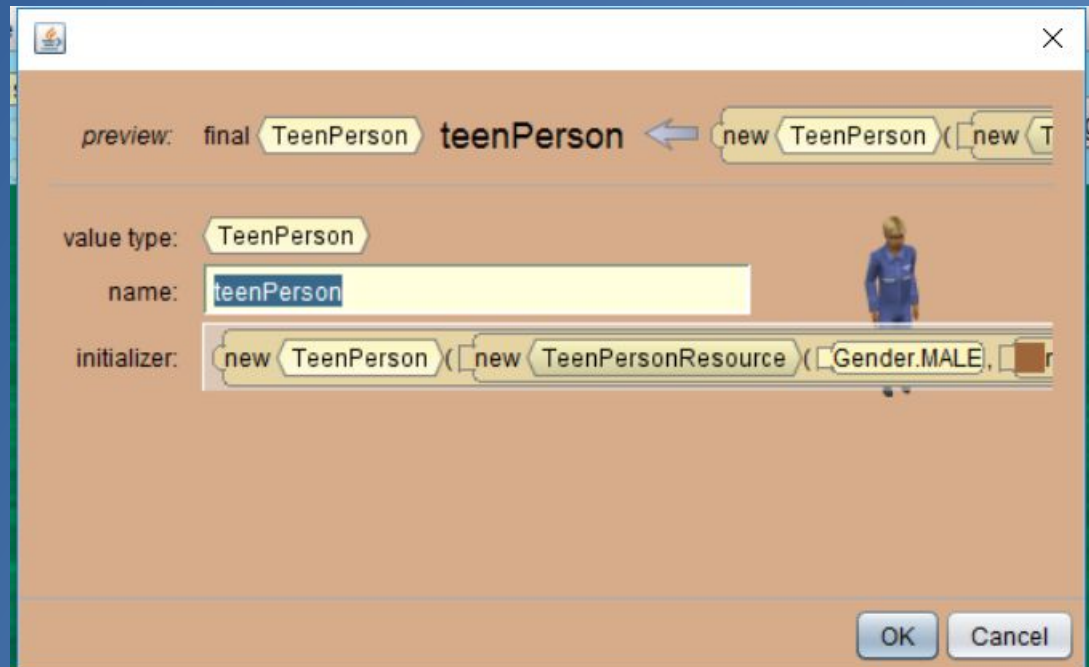




# Customize!!!



# Complete



The image shows a 'New' dialog box from a Java IDE. The dialog has a title bar with a close button. The main area is divided into sections for 'preview', 'value type', 'name', and 'initializer'. The 'preview' section shows a final variable 'teenPerson' of type 'TeenPerson' being assigned a new instance of 'TeenPerson'. The 'value type' section shows 'TeenPerson' selected. The 'name' section has a text field containing 'teenPerson'. The 'initializer' section shows a new instance of 'TeenPerson' being created with a 'TeenPersonResource' of type 'Gender.MALE'. The dialog has 'OK' and 'Cancel' buttons at the bottom right.

preview: final TeenPerson teenPerson ← new TeenPerson ( new T

value type: TeenPerson

name: teenPerson

initializer: new TeenPerson ( new TeenPersonResource ( Gender.MALE, r

OK Cancel

# Move your avatar around with one shots



# Or also using properties



# Experiment with different movements

You can always UNDO!

Lets do a cartwheel!

Start with the right arm

Roll or Turn?

How much?

Now the left arm



How about the hips?

Roll or Turn?

How much?

What's really going on here?

Ok lets roll!

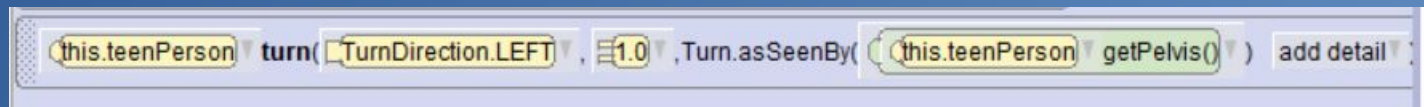
Ok lets roll!

Ok lets turn!

Wait that didn't Work!

We need an axis.

# Lets use “as seen by”





# And back to the code

```
public void myFirstMethod() {  
    this.teenPerson.getRightShoulder().turn( TurnDirection.BACKWARD, 0.375 );  
    this.teenPerson.getLeftShoulder().turn( TurnDirection.BACKWARD, 0.375 );  
    this.teenPerson.getRightHip().turn( TurnDirection.RIGHT, 0.125 );  
    this.teenPerson.getLeftHip().turn( TurnDirection.LEFT, 0.125 );  
    this.teenPerson.turn( TurnDirection.LEFT, 1.0, Turn.asSeenBy( this.teenPerson.getPelvis() ) );  
}
```

Now we need to move

# Doing things at the same time with do Together

```
ThreadUtilities.doTogether( ()-> {  
    this.teenPerson turn( TurnDirection.LEFT , 1.0 , Turn.asSeenBy( this.teenPerson.getPelvis() ) add detail );  
}, 0 -> {  
    this.teenPerson move( MoveDirection.LEFT , 0.5 , Move.asSeenBy( this.camera ) add detail );  
});
```

# Back to the Code

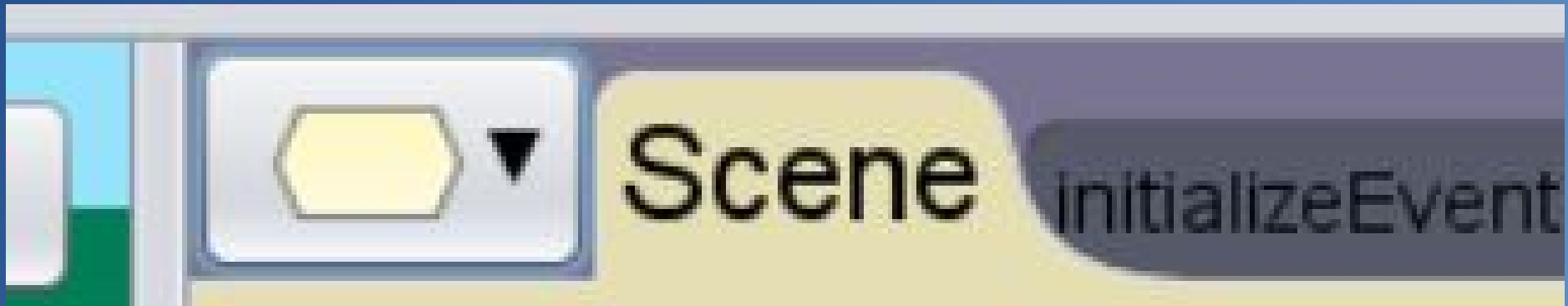
```
public void myFirstMethod() {
    this.teenPerson.getRightShoulder().turn( TurnDirection.BACKWARD, 0.375 );
    this.teenPerson.getLeftShoulder().turn( TurnDirection.BACKWARD, 0.375 );
    this.teenPerson.getRightHip().turn( TurnDirection.RIGHT, 0.125 );
    this.teenPerson.getLeftHip().turn( TurnDirection.LEFT, 0.125 );
    doTogether( ()-> {
        this.teenPerson.turn( TurnDirection.LEFT, 1.0, Turn.asSeenBy( this.teenPerson.getPelvis() ) );
    }, ()-> {
        this.teenPerson.move( MoveDirection.LEFT, 0.5, Move.asSeenBy( this.camera ) );
    } );
}
```

And we have a cartwheel!

Again! Again!

Well for that we need to make  
some changes

# Creating a cartwheel procedure

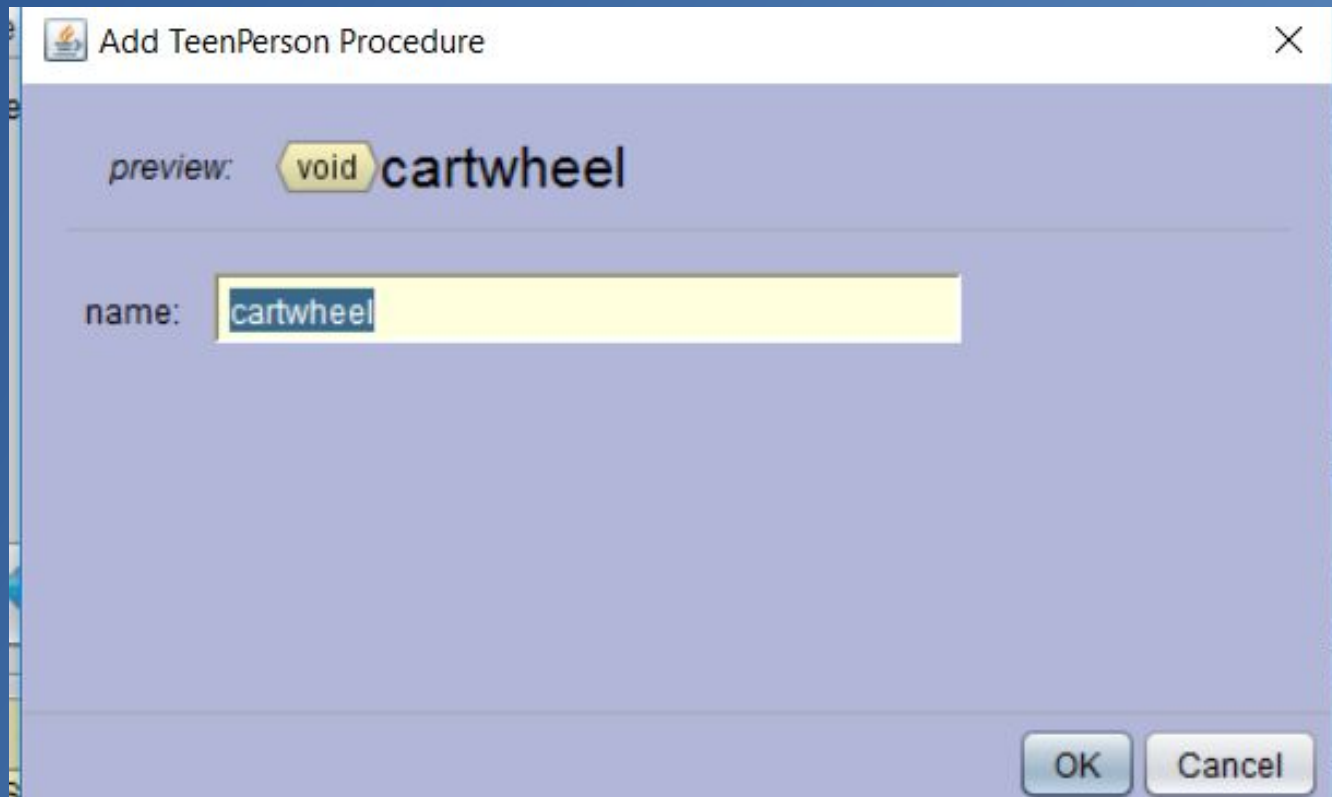




# Creating a cartwheel procedure



# Name it



A dialog box titled "Add TeenPerson Procedure" with a close button (X) in the top right corner. The dialog has a light purple background. It contains a "preview:" label followed by a yellow button labeled "void" and the text "cartwheel". Below this is a "name:" label followed by a yellow text input field containing the text "cartwheel". At the bottom right, there are two buttons: "OK" and "Cancel".

Add TeenPerson Procedure

preview: void cartwheel

name: cartwheel

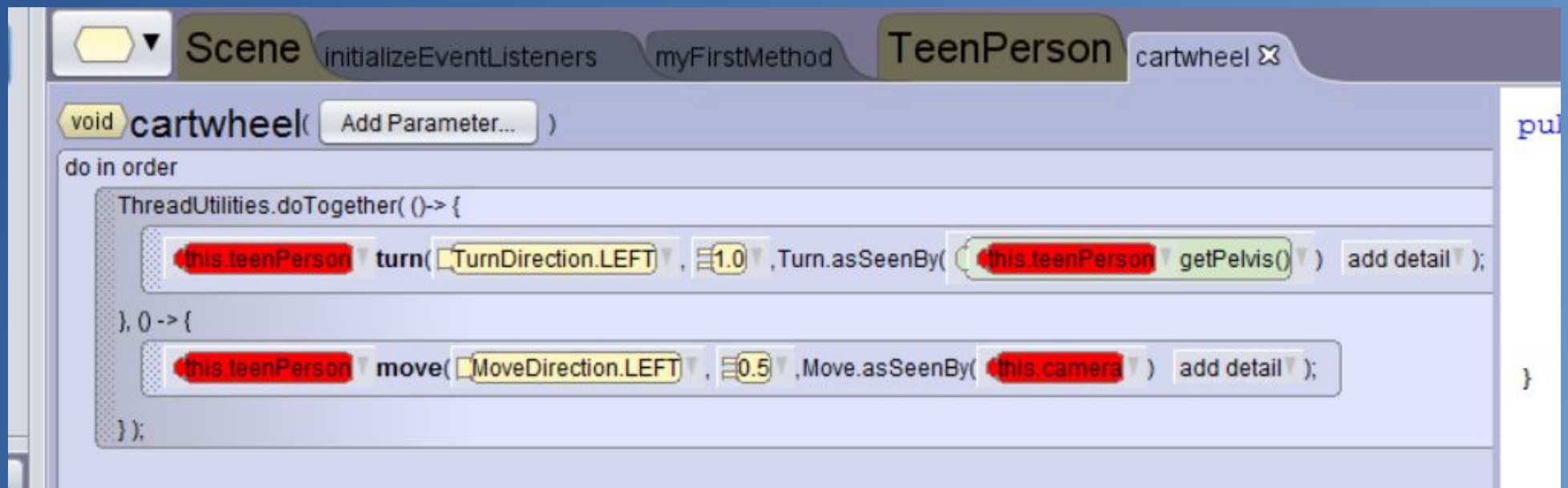
OK Cancel

# Move your Do Together code to the Clipboard

```
ThreadUtilities.doTogether( ()-> {  
    this.teenPerson turn( TurnDirection.LEFT , 1.0 , Turn.asSeenBy( this.teenPerson getPelvis() ) add detail );  
}, 0 -> {  
    this.teenPerson move( MoveDirection.LEFT , 0.5 , Move.asSeenBy( this.camera ) add detail );  
});
```



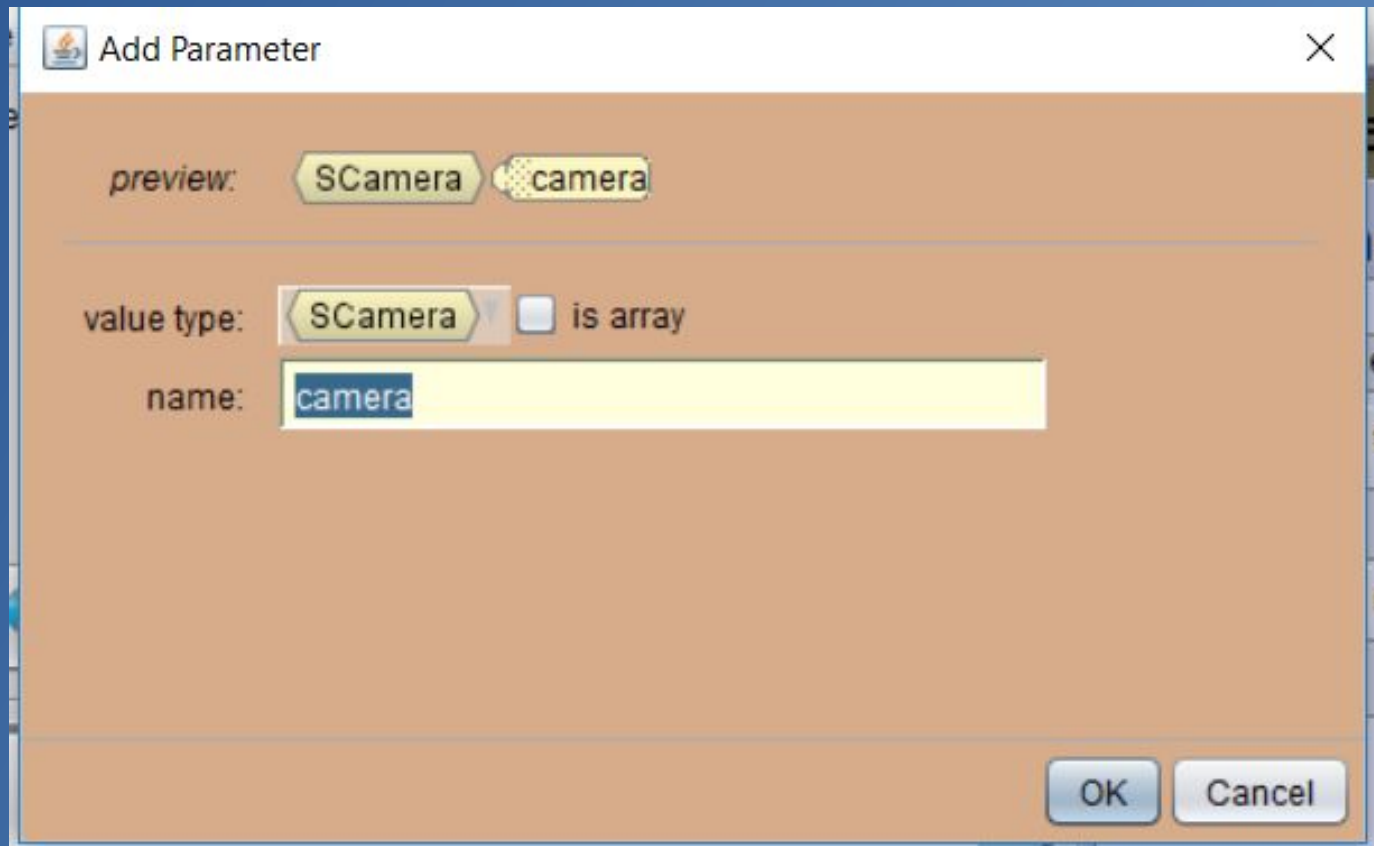
# Drag it to the Cartwheel Procedure



# Replace with this Now we need a camera!

```
void cartwheel( Add Parameter... )  
do in order  
{  
  ThreadUtilities.doTogether( ()-> {  
    this turn( TurnDirection.LEFT, 1.0, Turn.asSeenBy( this getPelvis() ) add detail );  
  }, 0 -> {  
    this move( MoveDirection.LEFT, 0.5, Move.asSeenBy( this camera ) add detail );  
  }  
});
```

# Adding the camera



The image shows a software dialog box titled "Add Parameter" with a close button (X) in the top right corner. The dialog has an orange background and contains the following fields:

- preview:** A dropdown menu showing "SCamera" and a text input field containing "camera".
- value type:** A dropdown menu showing "SCamera" and a checkbox labeled "is array" which is currently unchecked.
- name:** A text input field containing "camera".

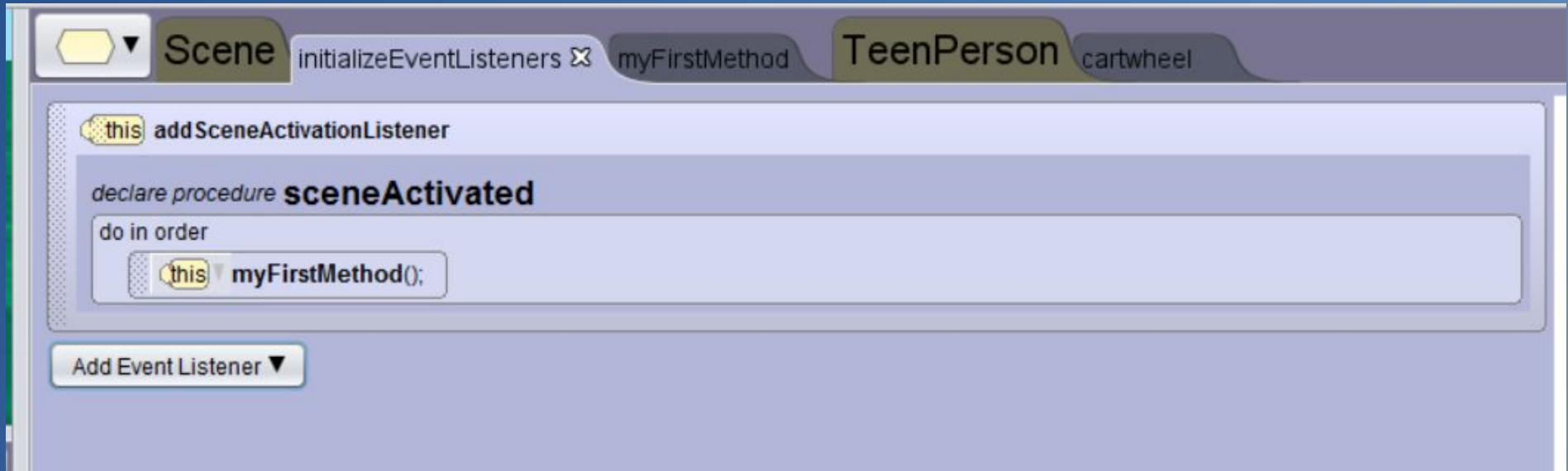
At the bottom right of the dialog are two buttons: "OK" and "Cancel".

# And we have a cartwheel procedure!





# Listening for a Key press





Try again. Press spacebar.

Break!

But what is really happening??? For  
that we need to look in an IDE

# NetBeans IDE: What is it?

Josh Juneau

Apress and Oracle Author

Java EE Application Developer

# Why is Coding Sometimes Difficult?



# Why is Coding Sometimes Difficult?

- In order to run a Java program, one must compile it first, then execute it. The compilation process is not too difficult, but it can become cumbersome if there are a lot of project dependencies.
- Code can be hard to read in some cases, especially when you get into lots of different class files, loops, or nested constructs.
- Code can become unorganized
- Etc, etc, etc...



# So What is an IDE?

A tool that is meant to help make coding **much** easier.

I.D.E = Integrated Development Environment

- Organization
- Ease the development process
- Easier to Read Code
- Shortcuts!

# NetBeans Overview

NetBeans IDE lets you quickly and easily develop Java desktop, mobile, and web applications, many other languages.

It is free and open source and has a large community of users and developers around the world.



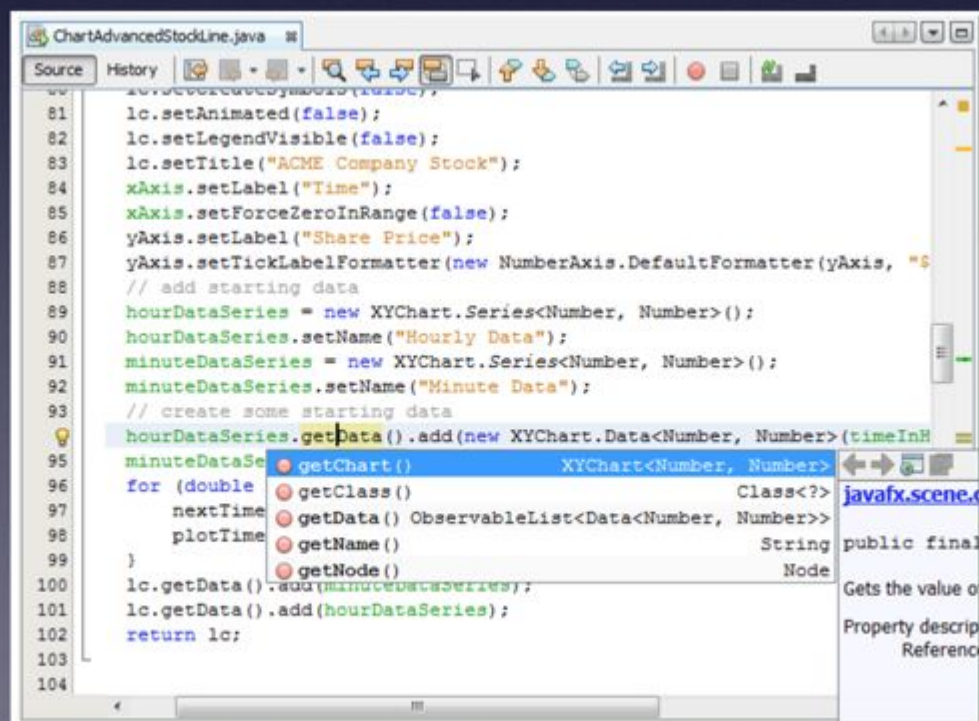
# Smart and Fast Coding

Auto-completion

Color coded Syntax

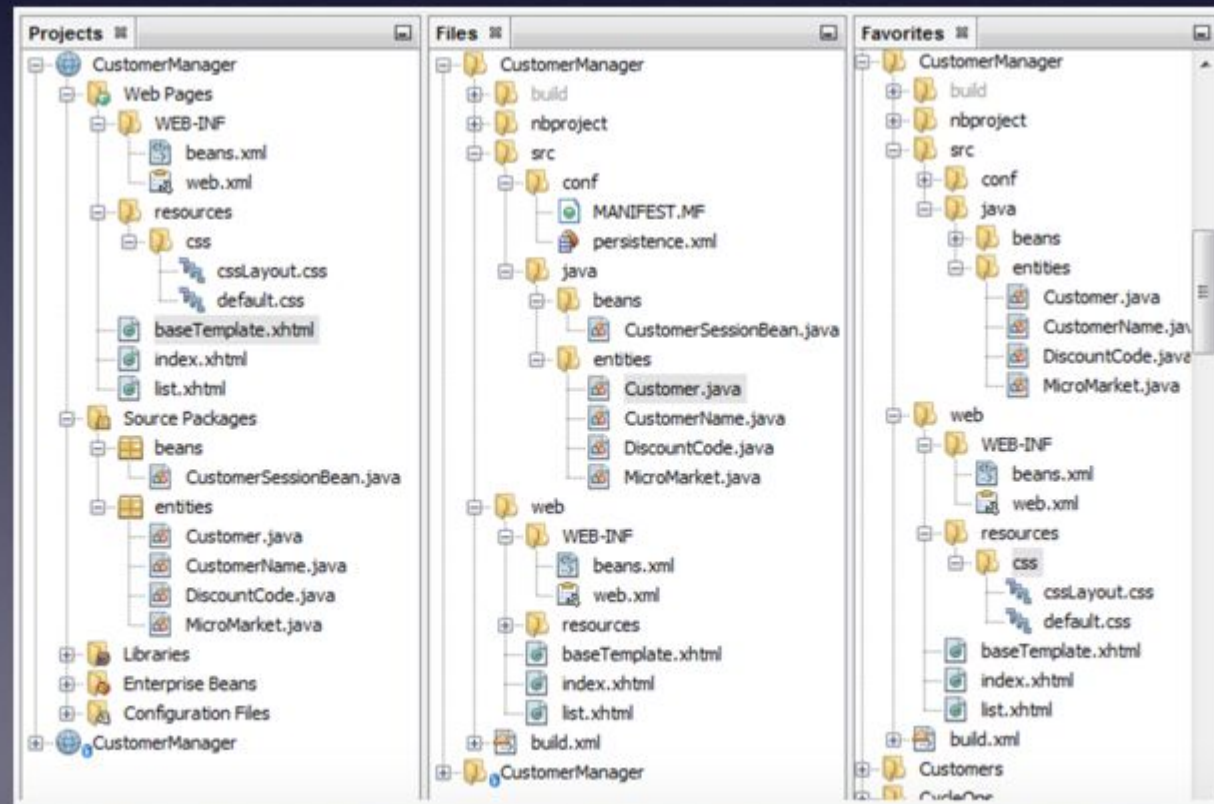
Hints and Tips

Code Generation, Refactoring



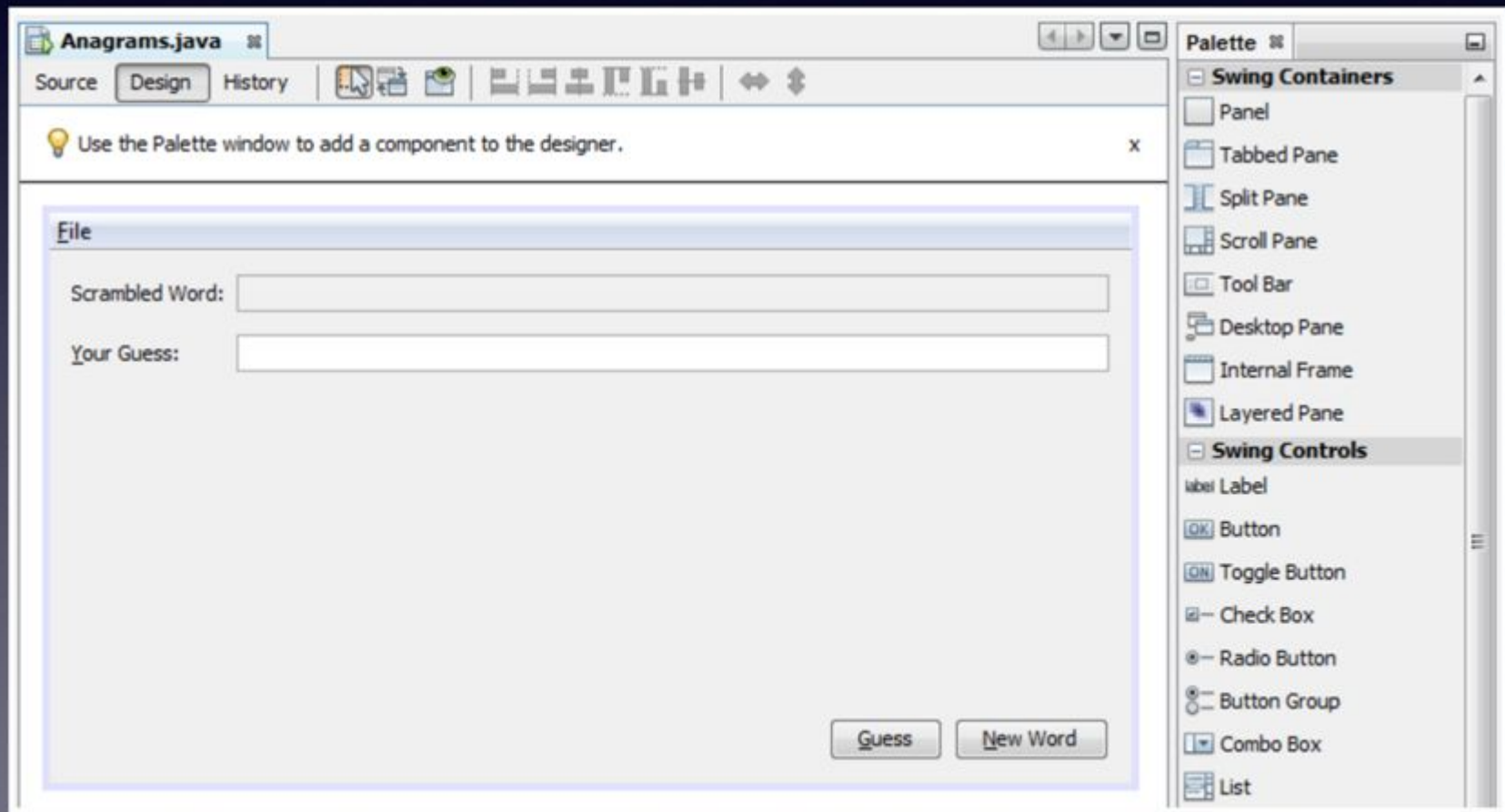
# Easy Organization

Create folders, group code together  
Organized Projects (Web apps, Desktop apps, etc)  
Different Symbols to Help Recognize Code Types



# Drag and Drop

Drag and Drop for Easy GUI Creation

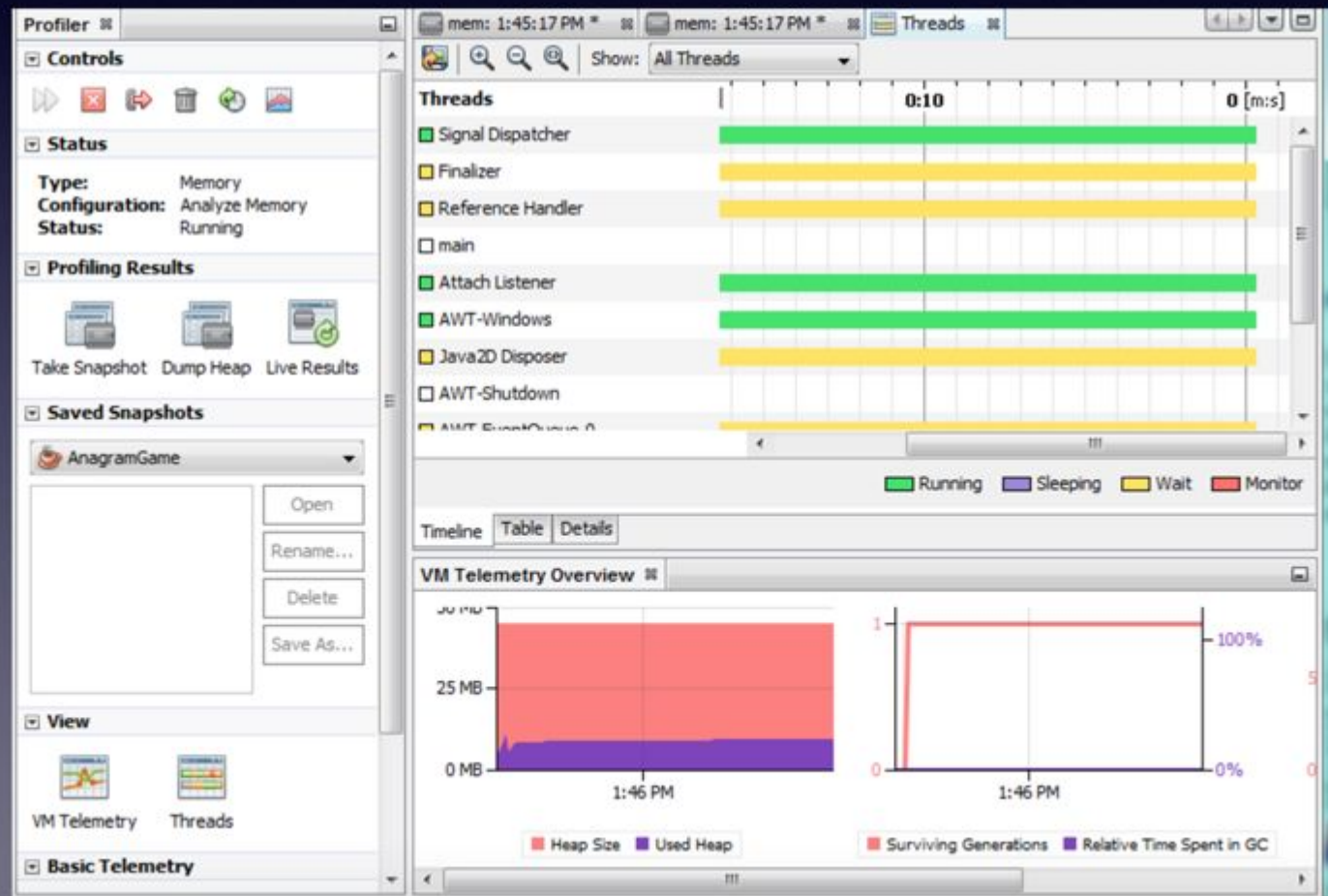




# Test...Get Rid of Bugs!

Bug finder...runs basic tests.

Profiler...where are my problem areas.

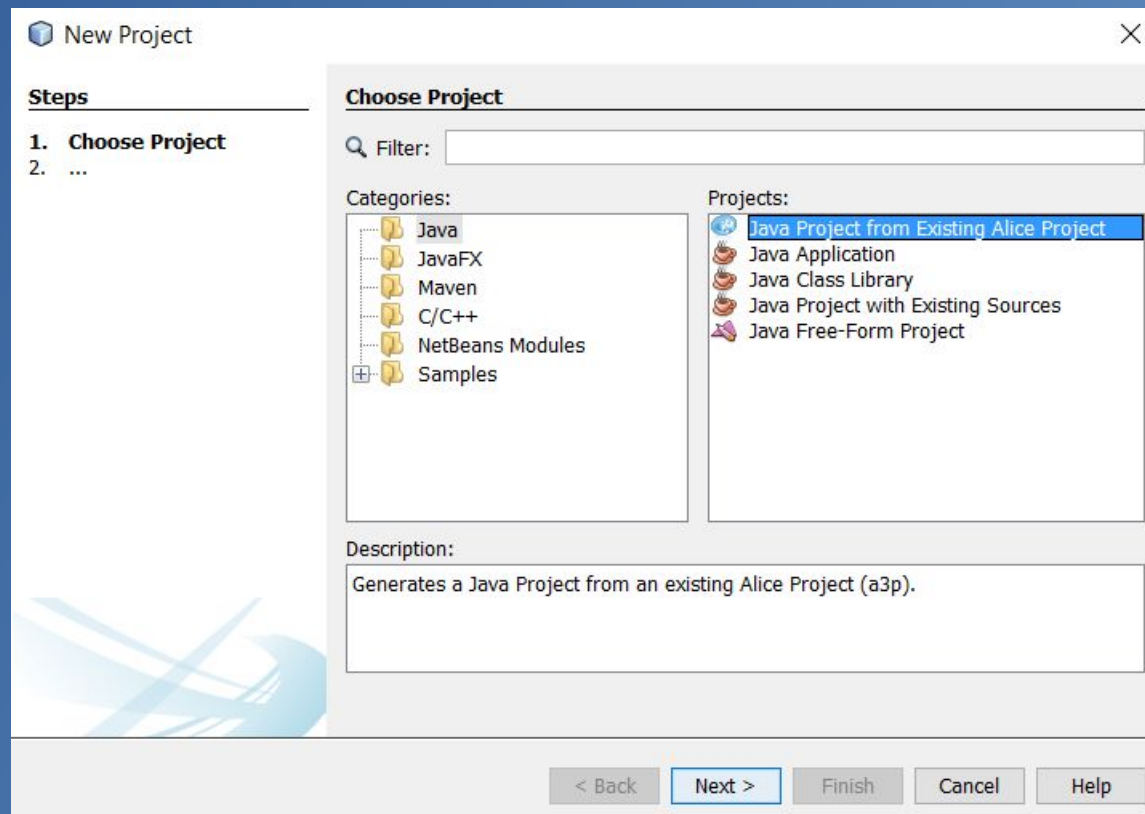


# NetBeans Makes Development Easy

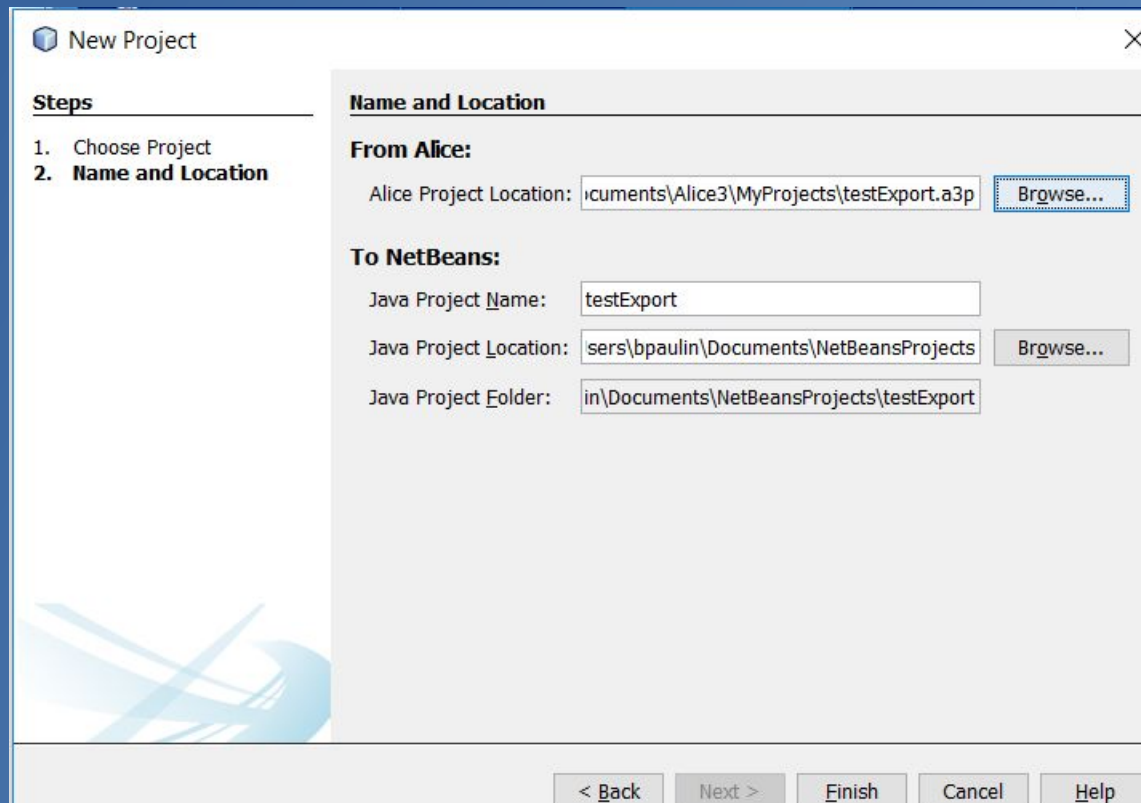


OK so lets do jumping jacks in  
Netbeans!

# Start by exporting your project to Netbeans

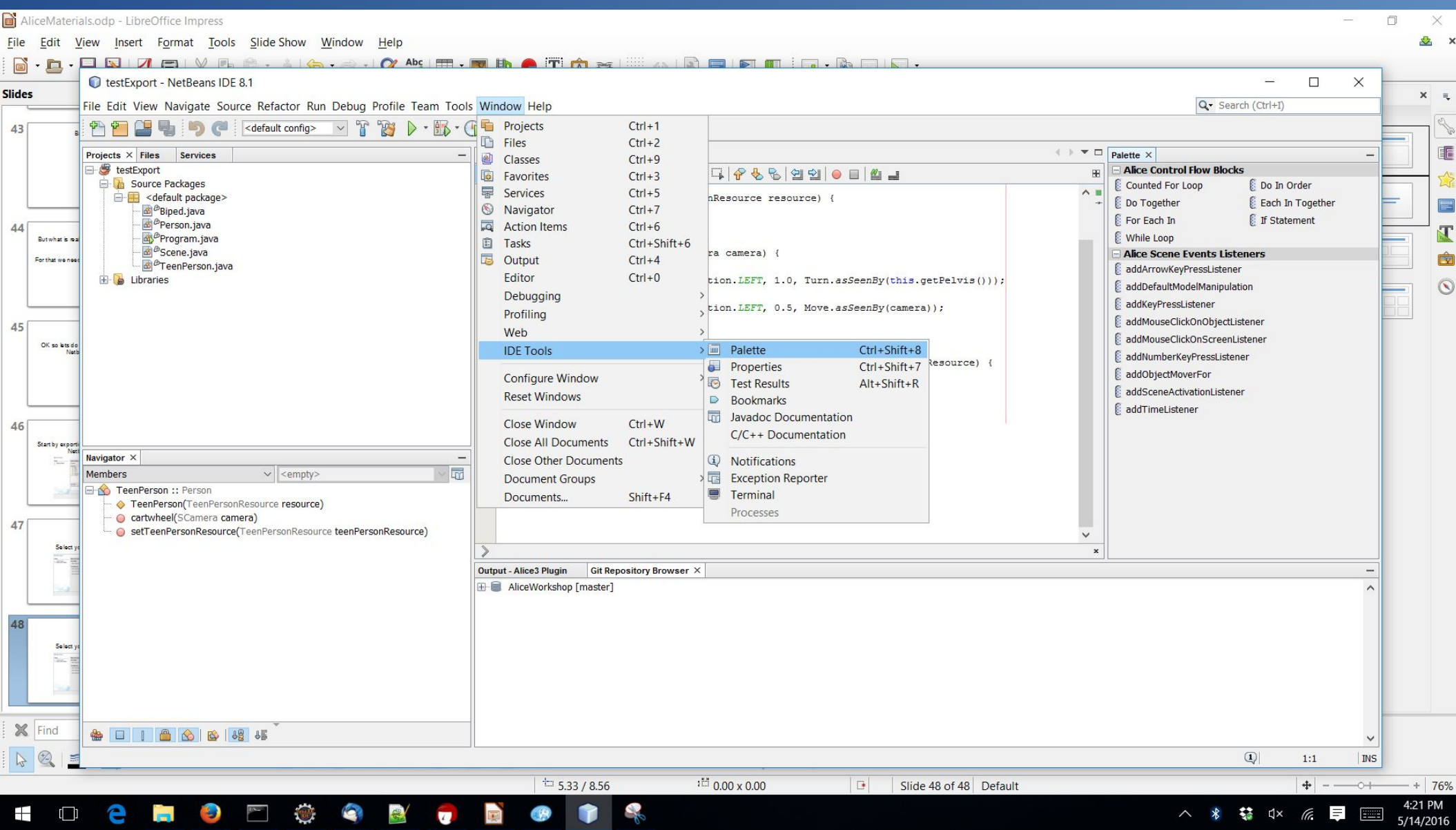


# Select your project

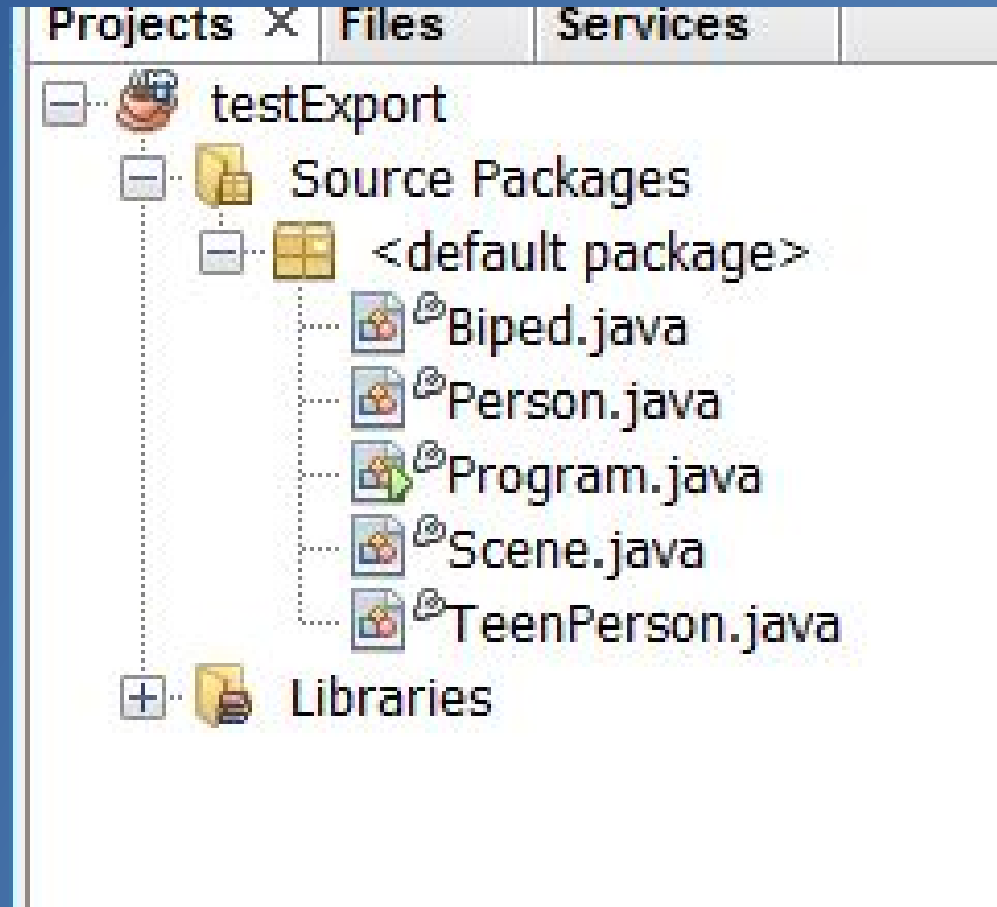
The image shows the 'New Project' dialog box in NetBeans. The title bar says 'New Project' with a close button. On the left, a 'Steps' pane shows '1. Choose Project' and '2. Name and Location', with the second step being active. The main area is titled 'Name and Location'. Under 'From Alice:', the 'Alice Project Location' is set to 'cuments\Alice3\MyProjects\testExport.a3p' with a 'Browse...' button. Under 'To NetBeans:', the 'Java Project Name' is 'testExport', the 'Java Project Location' is 'ers\bpaulin\Documents\NetBeansProjects' with a 'Browse...' button, and the 'Java Project Folder' is 'in\Documents\NetBeansProjects\testExport'. At the bottom are buttons for '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.



# Turn on the Palette



# Looking at what's there



# I can add new things in Scene

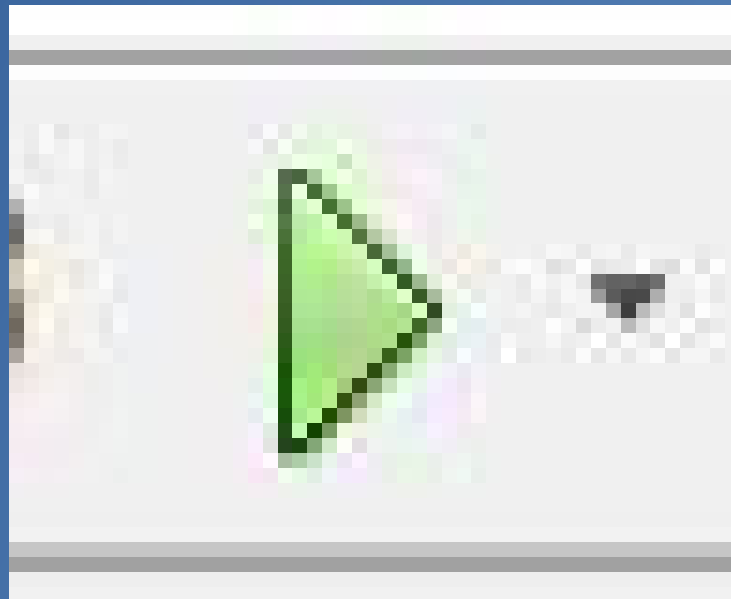
```
private void initializeEventListeners() {
    this.addSceneActivationListener((SceneActivationEvent event) -> {
        this.myFirstMethod();
    });
    this.addKeyPressListener((KeyEvent event) -> {
        this.teenPerson.cartwheel(this.camera);
    });
}

public void myFirstMethod() {
    this.teenPerson.getRightShoulder().turn(TurnDirection.BACKWARD, 0.375);
    this.teenPerson.getLeftShoulder().turn(TurnDirection.BACKWARD, 0.375);
    this.teenPerson.getRightHip().turn(TurnDirection.RIGHT, 0.125);
    this.teenPerson.getLeftHip().turn(TurnDirection.LEFT, 0.125);
}
```

# I can add new things in TeenPerson

```
class TeenPerson extends Person {  
  
    public TeenPerson(TeenPersonResource resource) {  
        super(resource);  
    }  
  
    public void cartwheel(SCamera camera) {  
        doTogether(() -> {  
            this.turn(TurnDirection.LEFT, 1.0, Turn.asSeenBy(this.getPelvis()));  
        }, () -> {  
            this.move(MoveDirection.LEFT, 0.5, Move.asSeenBy(camera));  
        });  
    }  
}
```

# Running the code



# Adding a Jumping Jacks Method

```
public void cartwheel(SCamera camera) {  
    doTogether(() -> {  
        this.turn(TurnDirection.LEFT, 1.0, Turn.asSeenBy(this.getPelvis()));  
    }, () -> {  
        this.move(MoveDirection.LEFT, 0.5, Move.asSeenBy(camera));  
    });  
}  
  
public void jumpingJacks()  
{  
    |  
}
```

# Adding a Jumping Jacks Method to TeenPerson

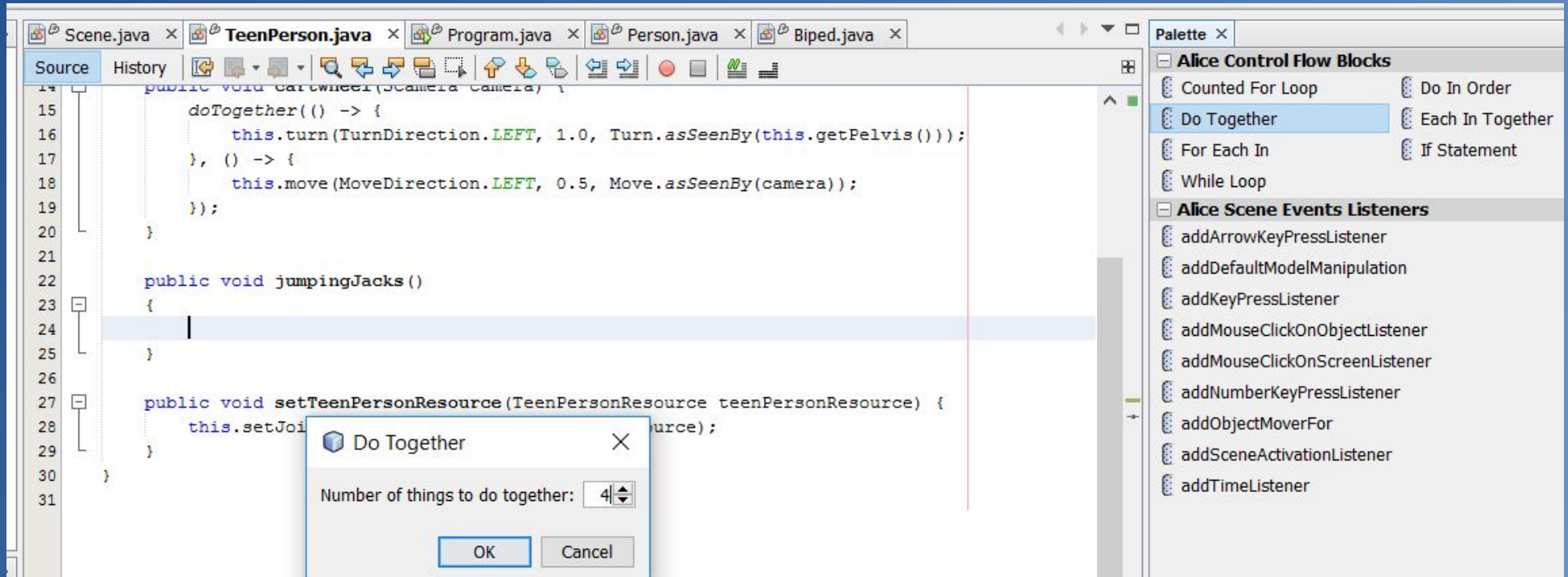
```
public void cartwheel(SCamera camera) {  
    doTogether(() -> {  
        this.turn(TurnDirection.LEFT, 1.0, Turn.asSeenBy(this.getPelvis()));  
    }, () -> {  
        this.move(MoveDirection.LEFT, 0.5, Move.asSeenBy(camera));  
    });  
}  
  
public void jumpingJacks()  
{  
    |  
}
```

# Hooking Jumping Jacks to the Key Listener in Scene

```
private void initializeEventListeners() {  
    this.addSceneActivationListener((SceneActivationEvent event) -> {  
        this.myFirstMethod();  
    });  
    this.addKeyPressListener((KeyEvent event) -> {  
        this.teenPerson.jumpingJacks();  
    });  
}
```



# Using the Palette



# Do Together Generated with Palette

```
public void jumpingJacks()  
{  
  
    //start a Thread for each Runnable and wait until they complete  
    doTogether(() -> {  
        //TODO: Code goes here  
    }, () -> {  
        //TODO: Code goes here  
    }, () -> {  
        //TODO: Code goes here  
    }, () -> {  
        //TODO: Code goes here  
    });  
  
}
```

Now it's your turn! If you get stuck  
try doing it inside Alice first and look  
at the code

Write a story to implement in Alice

Build your story

Demo Time!