Getting Started with Eclipse

Introduction

This guide will help get you set up to be able to work on your homework assignments. Yay! Get a cup of coffee and work your way through it.

First, you will have to decide between the following two options:

- 1. Working on the computer accounts we provide.
- 2. Working on your own laptop.

This guide will guide you through both of these options. We will first look at how to get each option set up properly. After that, we will look at the Eclipse development environment.

Contents

- Introduction
- · Getting Everything Ready and Installed
 - Working With the Accounts We Provide
 - Working on Your Own Computer
- Working With Eclipse
 - A Tour Through Eclipse
 - Writing and Executing Java Programs
 - Formatting Source Code
 - Importing Classes

Getting Everything Ready and Installed

Working With the Accounts We Provide

Working with the accounts we provide has the advantage that most things are already set up for you. To do so, either go to the *Grundausbildungspool* in the basement of *Hermann-Rodewald-Straße* 3 or login to the servers using your own computer. In both cases, you need to login with the credentials you got when you signed up for studying computer science (the user name will most likely be something like *stu0000*). To login using your own computer, you need to install the ThinLinc client application. When you start the ThinLinc client, you are not only prompted for your credentials, but also for a server address; use thinlinc.informatik.uni-kiel.de. Once you are logged in, you can find a development environment called *Eclipse* through the start menu.

Whatever way you choose: to get ready to work on your homework assignments, you do have to download the ACM Java library. Click this link and save the file somewhere you'll be able to find it again. For better documentation inside of Eclipse, also download this archive and save it at the same place as the ACM Java library.

Working on Your Own Computer

To be able to work on your own computer, you need to make sure that you have three things installed:

- The Java Development Kit. To download that, go to this site and use the left big button to download the Java Platform (JDK).
- The Eclipse development environment. Simply follow this tutorial to install the Eclipse IDE for Java Developers.
- The ACM Java library most of our assignments use. Click this link and this link and save the files somewhere you'll be able to find them again.

Working With Eclipse

This section will help you get to grips with the Eclipse development environment. We'll start with a small tour around Eclipse, and finish by walking through all the steps necessary to actually write and execute your first Java program.

A Tour Through Eclipse

When you start Eclipse for the first time, you'll see something like the following dialogue that asks you for where you want your workspace to be saved:

elect a wor	Workspace Launcher		
Eclipse store	es your projects in a folder called a workspace. orkspace folder to use for this session.		
Workspace:	/Users/cds/Documents/workspace	<u>~</u>	Browse
Use this a	as the default and do not ask again		
		Cancel	ОК

The workspace is where you will be saving all your homework. If you haven't worked with Eclipse before, just click OK and thereby accept the default location. Eclipse then loads and greets you with a nice welcome screen. Feel free to switch off the "Always show Welcome at start up" option at the bottom and dismiss the screen by clicking the X at the top.

Welcome	eclipse-w	vorkspace -	Eclipse	🟠 🗇 🖓 🕂 🔚 🗖 🗗
ecli	OSE Welcome to the Eclipse IDE	for Java	a Developers	Workbench
0	Review IDE configuration settings Review the IDE's most fiercely contested preferences		Overview Get an overview of the features	
Ë	Create a Helio World application A guided walkthrough to create the famous Helio World in Eclipse		Tutorials Go through tutorials	
0	Create a new Java project Create a new Java Eclipse project	1	Samples Try out the samples	
•	Checkout projects from Git Checkout Eclipse projects hosted in a Git repository		What's New Find out what is new	
*	Import existing projects Import existing Eclipse projects from the filesystem or archive			
	Launch the Eclipse Marketplace Enhance your IDE with additional plugins and install your Marketplace favorites		✓ Alw	ays show Welcome at start up

What you see next is the actual Eclipse development environment that you'll be spending most of your time in:

	🧿 Java - LeJavaF	Project/src/Rather	oodClass.java	- Eclipse			
° ∎• E N &• O• • • €• € 0•	🖄 😂 🖋 • 🚏 🗾 🗧 🔳	🖬 🔌 🔮 + 🖗 +	$\leftarrow \bullet \bullet \bullet$		Qu	Jick Access	含 🖏 Jav
Package Explorer S LolavaProject LolavaProject LolavaProject LolavaProject	RatherGoodClass.java X Import acm.program. J public class Rather S }	GraphicsProgram;		gram {		© Outline ⊠ ■ Outline ⊠ ■ ■ 1/2 × * P RatherGood	= (• 🗙
	Problems 33 @ Javadoc O errors, 1 warning, 0 others	Q Declaration				p	▼ □
	Description Image: Second stress <		∧ Resource	Path	Location	Туре	

The window is divided into different areas. Most of them contain *views*, such as the *Package Explorer* to the left. The big empty area at the centre of the screen will house the Java source code editor, a text editor to write Java code with. You can drag views around, and close them as you wish. For example, you won't need the *Task List* during our lecture. Feel free to close it now. If you want to reopen a view you have closed, you can find a list of all available views through the menu by clicking *Window -> Show View -> Other*.

Writing and Executing Java Programs

To solve your homework assignments, you will need to know how to create new Java projects, add classes to them, and how to execute them. Let's start by creating a new project. Right-click in the *Package Explorer*, and select *New -> Java Project*. This will open the following dialogue:

class.	
class.	
LeJavaProject/src	Browse
(default)	Browse
	Browse
public package private protected abstract final static	
java.lang.Object	Browse
	Add
	Remove
s would you like to create?	
public static void main(String[] args)	
Generate comments	
Cancel	Finish
	(default) (default) (default) (default) (abstract final static java.lang.Object java.lang.Object joublic static void main(String] args) Constructors from superclass inherited abstract methods comments? (Configure templates and default value here) Generate comments

Enter a project name that describes the project. For your homework assignments, for example, you may want the project name to contain the assignment the project is supposed to solve. Leave the rest untouched and click *Finish*. Your Eclipse workspace will now contain an entry for the new project. Eclipse has to be told that you want to use the ACM library with that project. To configure it that way, right-click on the project, click on *Build Path* and select *Config ure Build Path...*:

······································	L • 🢁 • 🖶 🞯 • 🗁 🔗 • 🕭 •	- 1.0s			Quick Access
		« ([X])			
Package Explorer 🔀				- 0	An outline is not available.
▶ 🛃 LeJavaProject					
	New Go Into	•			
	Open in New Window				
	Open Type Hierarchy Show In て第W	F4			
	Сору	ЖC			
	Copy Qualified Name Paste	жv			
	X Delete	æv ⊠			
	& Remove from Context	}ж1			
	Build Path	•	Surce		
	Source て第S Refactor て第T	*	Source Folder		
	≥ Import		💭 Use as Source Folder a Add External Archives		
	🖆 Export		🛋 Add Libraries		
	🔗 Refresh	F5	🌺 Configure Build Path		
	Close Project Assign Working Sets				
	Coverage As	►			
	Run As Debug As	•	Leclaration		* * * -
	Validate				
	Restore from Local History		 Resource 	Path Location	Туре
	Team Compare With	•			
	Configure	•			
	Properties	¥١			
JavaProject					

Remember the ACM library you downloaded? This is where we need it. Since you will be making use of that library, we need to configure the project such that it knows that the library exists. To that end, switch to the *Libraries* tab:

type filter text	Properties for LeJavaProject	á
type filter text Resource Builders Coverage Java Build Path Java Code Style Java Compiler Java Compiler Java Editor Java Editor Javadoc Location Project References Refactoring History Run/Debug Settings Task Repository Task Tags Validation WikiText	Java Build Path	Add JARs Add External JARs Add Variable Add Library Add Class Folder
		Add External Class Folder Edit Remove Migrate JAR File Apply
?		Cancel Apply and Close

A library in Java is distributed as a file with the `.jar` file extension. Click on the Add External JARs button and select the downloaded ACM library (if you don't remember where you saved it, simply download the library again). Once you have found the library, it should show up in the dialogue. If you expand its entry, it should look something like this:

type filter text	Java Build Path	$(\neg \bullet \neg \neg)$
 Resource Builders Coverage Java Build Path Java Code Style Java Compiler Java Editor Java Editor Java Editor Project References Refactoring History Run/Debug Settings Task Repository Task Tags Validation WikiText 	Image: Source attachment: (None) Add Exemption: (None) Image: Access rules: (No restrictions) Add Image: Access rules: (No restrictions) Add <td>port Id JARs ternal JARs Variable d Library lass Folder Edit Edit Remove te JAR File Apply</td>	port Id JARs ternal JARs Variable d Library lass Folder Edit Edit Remove te JAR File Apply
?	Cancel	Apply and Clos

The ACM library is not ready to be used, but Eclipse won't display its documentation. We like documentation, so we tell Eclipse where to find it. Doubleclick the Source attachment entry, which will open the following dialog:

ype filter to	ext	Java Build Pa	ath				⇔・⇒
Resource Builders			(# Source	Projects	🛋 Libraries	Order and Export	
Coverage		5	Source Attachr				
Java Builc Java Code		on (folder, JAR or zi	in) containing the	source for lac	m iar!		
Java Cout Java Com	Workspace lo		p/containing the	source for ac	.m.jar .		
Java Edito	workspace to	cation					Rs
Javadoc L Project Re	Path:					Browse	
Refactorir	External locat						
Run/Debu	External locat	ion					
Task Repo Task Tags	Path:					External File)
Validation						External Folder	er
WikiText	incoding:	Default (UTF-8) 💌				External Folder	
	\						Folder
	?				Canc	el OK	
						Migrate JA	R File
							Apply
_						Cancel Ap	ply and Clos
?)							

Select External location and click the External File... button to select the acm.zip file you downloaded. Click OK to dismiss the dialog, click Apply and Close to dismiss the build path configuration and you're all set!

You are now ready to add your first class to your newly configured project. Right-click your new project in the Package Explorer and select New -> Class:

class.	C
LeJavaProject/src	Browse
(default)	Browse
	Browse
• public opackage private protected	
abstract final static	
java.lang.Object	Browse
	Add
	Remove
s would you like to create?	
public static void main(String[] args)	
Constructors from superclass	
Inherited abstract methods	
Generate comments	
Cancel	Finish
	default; default; default; default; default; default; default; default of the create? default value bere; def

Give your class a proper name. The dialogue will tell you if your class name is valid or not. For the first few programming exercises, you want your class to extend a superclass, such as GraphicsProgram. To choose a superclass, click the corresponding *Browse* button. The result may look something like this:

	New Java Class	
ava Class	fault package is discouraged.	C
Source folder:	LeJavaProject/src	Browse
Package:	(default)	Browse
Enclosing type:		Browse
Name:	RatherGoodClass	
Modifiers:	public package private protected abstract final static	
Superclass:	acm.program.GraphicsProgram	Browse
Interfaces:		Add
		Remove
Which method stubs	s would you like to create?	
	public static void main(String[] args)	
	Constructors from superclass	
D	Inherited abstract methods	
Do you want to add	comments? (Configure templates and default value here)	
(?)	Cancel	Finish

Note that the dialogue still gives us a warning because we have left the package name empty. You can safely ignore that warning for the moment. There will come a time when you will actually use packages, but now is not that time. Instead, click *Finish*. Eclipse will create the class for you and open it in an editor for you to start writing code:

		Project/src/RatherGo		a - Eclipse			[
Ů• 12• 🛛 🖄 🎋• O• 🂁• 🛱 🕫	• 🖄 🖾 🔗 • 🐨 🧾 🐑 🔳	🔳 🔌 🔮 • 🖗 • 🤉	$= \leftarrow = =$			uick Access	😰 🐉 Ja
Package Explorer 🛛 🗖 🗖	🕗 RatherGoodClass.java 🔀					E Outline 🛙	-
EdavaProject V arc V (dofault package) ► () RatherGoodClass.java ► (Aff System Library (JavaSE-1.6) ► (Aff System Library) ► (Aff System Librar	<pre>1 Amport acm.program. 2 Amport acm.program. 3 public class Rather 4 5 } 6 }</pre>		GraphicsPro	gram (P E 12 X	
	Problems 22 @ Javadoc 0 errors, 1 warning, 0 others Description	-	Resource	Path	Location	Туре	
		Writable S	mart Insert	1:1			

Let's say you have actually written a bit of code and want to see if it works. Save your class. Then, right-click it in the *Package Explorer* and select *Run As* - *Java Applet* (or *Java Application*). If everything is fine, a window should pop up and your program should be executing. If instead Java cannot execute your program because of errors in your source code, Eclipse will tell you so:

	Errors exist in required project(s):		
· ·	LeJavaProject		
	Proceed with launch?		
Alwa	ays launch without asking		
		Cancel	Proceed
_			

In that case, hit *Cancel* and go hunt that error down. Remember the *Problems* view at the bottom of the Eclipse window? That's actually quite helpful for hunting down errors: it lists every error Eclipse was able to find. Double-clicking the error will jump to the offending part of the source code so you can fix it.

Formatting Source Code

The automatic tests we run on the source code you submit won't let you hand in badly formatted code. This is because we want to force you to write code that is properly readable. Fortunately, formatting your code properly is a matter of two clicks of a mouse button:

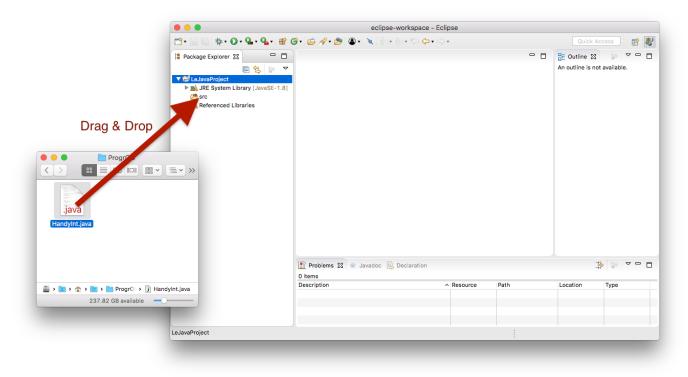
🗯 Eclipse File Edit So	ource Refactor Navigate Search	Projec	t Run Window Help	🖸 🖗 🛜 🗣 🗔	Thu 12 Oct 11:59 AM cds Q :三
	Toggle Comment		gons/src/here/be/dragons/RatherGoodCla	ass.java - Eclipse	
	Add Block Comment Remove Block Comment	^೫/ ^೫∖	≷∯+∦·♥≎+⇒+		Quick Access
	Generate Element Comment	∠.₩J	🚺 *RatherGoodClass.java 🔀	- 8	📴 Outline 🕱 🗖 🗖
A de system Library (Javas Second Libraries P			<pre>package here.be.dragons; public class RatherGoodClass { public void excellentMethod() { 7 } 8 Problems 23 @ Javadoc @ Declaration 0 items Description</pre>		
			Writable	Smart Insert 8 : 1	

Importing Classes

There will be assignments where we ask you to import existing classes into your workspace. There are (at least) two ways to do so:

- 1. Right-click the destination folder in your Eclipse project and choose Import....
- 2. Use drag & drop.

We will now describe the latter. Start by downloading the respective class. Then, import it into your Java project by dragging and dropping it from your file explorer to your source folder, like this:



Eclipse will ask you whether you want to copy the file into your project or just link to it. You want the former:

Package Explorer 🔀				📑 Outlin 🕄 🗖 🗖			
	Select how fill	File Operation File Should be imported into the project: Copy files Link to files					
		ate link locations relative to: PROJECT_LOC ag and Drop Settings Cancel	OK				
		Problems 🕄 @ Javadoc 😥 Declaration		⇒⇒ ≥ ⊏ ∈			
		Description	^ Resource	Path			

The imported file may declare to be in a different package than the one you imported it to, resulting in a compiler error. To fix that, simply open up the file, point the mouse cursor at the wrong package declaration, and choose to move the file to that package.

