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, scroll down a bit until you see Java SE 8, and click the JDK Download button to the right.
- 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	kspace		
Eclipse store Choose a we	s 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	is 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 to the Eclipse ID w IDE configuration settings w the IDE's most fiercely contested ences te a Hello World application	E for Java	a Developers Overview Get an overview of the features	Workbench
ew IDE configuration settings v the IDE's most fiercely contested ences te a Hello World application		Overview Get an overview of the features	
te a Hello World application			
ed walkthrough to create the famous Hello in Eclipse		Tutorials Go through tutorials	
te a new Java project a new Java Eclipse project	%	Samples Try out the samples	
kout projects from Git out Eclipse projects hosted in a Git tory		What's New Find out what is new	
ort existing projects t existing Eclipse projects from the tem or archive			
ch the Eclipse Marketplace ce your IDE with additional plugins and		Always show	Welcome at start up
	rt existing projects existing Eclipse projects from the em or archive th the Eclipse Marketplace e your IDE with additional plugins and your Marketplace favorites	t existing projects existing Eclipse projects from the em or archive the the Eclipse Marketplace e your IDE with additional plugins and your Marketplace favorites	t existing projects existing Eclipse projects from the em or archive the Eclipse Marketplace e your IDE with additional plugins and your Marketplace favorites

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	iick Access	含 🖏 Jav
Package Explorer S LolavaProject LolavaProject LolavaProject LolavaProject	RatherGoodClass.java X Import acm.program. J public class Rather S }	GraphicsProgram; GoodClass extend	s GraphicsPro	gram {		E Outline X ■ I I I P I I I I P RatherGood	Class
	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:

ente a lava Pro-l	New Java Pr	oject	_
eate a Java Proj inter a project nam	ect ie.		
Project name:			
Use default loc	ation		
Location: /Users			Browse
JRF			
		(Jaure 07, 40)	
 Use an exec 	ution environment JRE:	Javase-10	
 Use a projection 	t specific JRE:	Java SE 10 [10]	0
 Use default 	JRE (currently 'Java SE 10 [10]')		Configure JREs
Project layout			
O Use project	folder as root for sources and clas	s files	
 Create sepa 	rate folders for sources and class	files <u>Co</u>	onfigure default
Working onto			
working sets			
Add project	to working sets		New
Working sets:		٥	
2)	< Back Nex	t > Cancel	
. /			

Enter a project name that describes the project. For your homework assignments, for example, you may want to create a separate project for each set of assignments, so a project name like "Set01" might work well for the first set. Leave the rest untouched and click *Finish*. Depending on your installation, you *may* see the following dialog:

	New module-info.java
Create modul Create a new r	e-info.java module-info.java file.
Module name:	fevbev
	Don't Create Create
-	

If that is the case, click the Don't Create button (things won't work otherwise).

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 *Configure Build Path...*:

Package Explorer 🔀						- 8	B. Outline 23	E
							An outline is not a	vailable.
🖻 LeJavaProject	New Go Into	►						
	Open in New Window Open Type Hierarchy Show In て発W	F4						
	 Copy Copy Qualified Name Paste Delete 	жс ж∨ ⊠						
	🚴 Remove from Context 🛛 🔿	Շ∺∔						
	Build Path Source て第S Refactor て第T		B Link Source ♣ New Source F	older				
	≧ Import ☑ Export		Use as Source Add External Add Libraries.	e Folder Archives				
		F5	Sconfigure Bui	ld Path				
	Coverage As Run As Debug As	* * *	Declaration				+1	, ∨ ⊓ E
	Validate Restore from Local History			Resource	Path	Location	Туре	
	Team Compare With Configure	* * *						
	Properties	۶						

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 (if your dialog box has two entries, *Modulepath* and *Classpath*, select *Classpath* before you continue):

type filter text Java Build Path Java Code Style Java Editor Jake Editor Add External JARs. Add Library Add External Class Folder Add External Class Folder Migrate JAR Flie		Properties for LeJavaProject	
 Resource Builders Coverage Java Build Path Java Code Style Java Editor Java Editor JARE System Library [JavaSE-1.8] Add JARS Add External JARS. Add Library Add Library Add External Class Folder Add External Class Folder Migrate JAR Flie 	type filter text 🛛 🔇	Java Build Path	$\langle \neg + \neg \rangle$
	 Resource Builders Coverage Java Build Path Java Code Style Java Compiler Java Editor Javadoc Location Project References Refactoring History Run/Debug Settings Task Repository Task Repository Validation WikiText 	Image: Contract of the second sec	Corder and Export Add JARs Add JARs Add External JARs Add Library Add Library Add Class Folder Add External Class Folder Edit Edit Remove Migrate JAR File
	0		Apply

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 Javadoc Location Project References Refactoring History Run/Debug Settings Task Tags Validation WikiText 	Source Projects Libraries JARs and class folders on the build path:	Add JARs Add JARs Add External JARs Add Uariable Add Library Add Class Folder Add External Class Folder Edit Remove Migrate JAR File
?)		Apply Cancel Apply and Close

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:

		Properties for LegavdProject	
type filter te	ext	Java Build Path	(
Resource Builders Coverage	• •	CH Source CH Projects A Order and Export	
Java Build			
■ Java Cooc Java Cooc	Select the locatio	on (folder, JAR or zip) containing the source for 'acm.jar':	
▶ Java Edite	 Workspace lo 	ocation	Ro
Javadoc L	Path:	Browse	
Refactorir	<u> </u>		
Run/Debu	External locat	tion	
Task Report Task Tage	Path:	External File	
► Validation		Enternal Falder	er
WikiText	Encoding:	Default (UTF-8)	
	1		Folder
	-		
	?	Cancel OK	
			_
	-		
	-	Migrate J	AR File
	-	Migrate J	AR File
	-	Migrate J	AR File
	-	Migrate J	AR File
	-	Migrate J	AR File Apply
	-	Migrate J	AR File

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:

ava Class Create a new Java	class.	C
Source folder:	LeJavaProject/src	Browse
Package:	(default)	Browse
Enclosing type:		Browse
Name:		
Modifiers:	public package private protected abstract final static	
Superclass:	java.lang.Object	Browse
Interfaces:		Add
		Remove
Which method stub	s would you like to create?	
	public static void main(String[] args)	
	Constructors from superclass	
	Inherited abstract methods	
Do you want to add	comments? (Configure templates and default value here)	
	Generate comments	
(?)	Cancel	Finish

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	alit package is discouraged.	C
Source folder:	LeJavaProject/src	Browse
Package:	(default)	Browse
Enclosing type:		Browse
Name:	RatherGoodClass	
Modifiers:	• public Opackage Oprivate Oprotected	
	abstract final static	
Superclass:	acm.program.GraphicsProgram	Browse
nterfaces:		Add
		Remove
Which method stub	s would you like to create?	
	public static void main(String[] args)	
	Constructors from superclass	
	Inherited abstract methods	
Do you want to add	Comments? (Configure templates and default value nere)	
	Conorate comments	
2	Cancel	Finish
\odot		

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:

	🧃 Java - LeJava	roject/src/hatherGoodGlass.j	ava - Eclipse		
3• 12• 13 16 ☆• ⊙• 9₄• # (3• 😂 😂 🖋 • 🕸 🏅 🗉	∏ ≥ ½ • 🖗 • ← + →	•	Quick Acc	cess 🛛 😰 🐯 Jav
I Package Explorer ⊠ □ E] 🗾 RatherGoodClass.java 🛿			🗖 🗖 📴 Outli	ne 🛛 🗖 [
E 😫 🐌 .	 1 import acm.program. 	GraphicsProgram;			$\downarrow^a_z \And \And^s \bullet \And^\iota$
 ValvaProject V⊕ arc V⊕ arc V⊕ (dofault package) Interv6codClassjava M RE System Library (JavaSE-1.8) M Referenced Libraries 	<pre>Q 3 public class Rather 4 5 5 } 6</pre>	<u>GoodClass</u> extends Graphics	Program (ο ο Ο Γ	RatherGoodClass
	Problems X @ Javadoc	Declaration			\$°
	Problems X @ Javadoc 0 errors, 1 warning, 0 others Description	Declaration	Path	Location Typ	in △ □ 10
	 Que transition of the second s	Declaration Resource	Path	Location Typ	e

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:

1	Errors exist in required project(s):
	LeJavaProject
	Proceed with launch?
🗌 Alw	vays launch without asking
	Cancel

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	Source Refactor Navigate Search	Projec	t Run Window Help		🕗 🌒 🛜 🖬) [4] 📰	Thu 12 Oct 11:59	AM cds	ର ≔
Eclipse File Edit Package Explorer S File Be Dragons ► M ARE System Library (JavaS V → Are Forecod Libraries File Forecod Libraries File Forecod Libraries File S > Homework Assignments (or File S > Homework Assignments) File S > Homework Assignments (or File S > Homework	Source Refactor Navigate Search Toggle Comment Add Block Comment Add Block Comment Add Block Comment Remove Block Comment Generate Element Comment Shift Right Shift Left Shift Left Shift Left Correct Indentation Format Format Add Import	Projec 第7 ^第/ 个第/ ~第 光 て第J 第1 企業F	t Run Window Help gons/src/here/be/dragons/Rr 	atherGoodClass.java	O O	• (*) 😨	Thu 12 Oct 11:59	Quick Access Quick Access agons dClass ttMethod() : void	Q := ₽ ₿ • ₽
► Very > Recitation Class (private-git in the second s	Add import Organize imports Sort Members Clean Up Override/Implement Methods Generate Delegate Methods Generate Delegate Methods Generate Constructor using Fields Generate Constructor using Fields Generate Constructor using Fields Generate Strings Find Broken Externalized Strings	ひ ж M 介 策 O							
			Problems 🕱 @ Javadoc	🚯 Declaration				→	~
			0 items						
			Description		A Resource	Path	Location	Туре	
				Writable Si	mart 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:

- Right-click the destination folder in your Eclipse project and choose *Import...* 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:

	ecli	pse-workspace - Eclipse			
📑 • 🔝 🐘 🔅 • • • •	🎍 🖶 🞯 • 🗁 🛷 • 🎘 🚇 • 🔪 🖁	2 • {} • \$> \$> • -> •		Quick A	cess 🔡 😭
Package Explorer 😫			- 8	E Outline 🔀	· · ·
	> マ			An outline is no	t available.
V 🖉 LeJavaProject					
JRE System Library [Java	aSE-1.8]				
C ^m src					
Referenced Libraries					
Drag & Drop					
Diag & Diop					
Progr					
va					
dyInt.java	Deckiane 😒 🖉 laundae	Provincentian			
dyint java	Problems 🕱 @ Javadoc	Q Declaration		1	
/int.java	Problems 23 @ Javadoc 0 items Description	Declaration	Dath		
intjava	Problems 23 @ Javadoc 0 items Description	Declaration	Path	Location	b 🐉 マ 🗆 Type
yInt.java > ☆ > ■ > ■ ProgrC > [] HandyInt.java 237.82 GB available	Problems 2 @ Javadoc O items Description	Declaration	Path	Location	Type
Int_java	Problems 23 @ Javadoc O items Description	Declaration	Path	Location	o oo ⊂ □ Type
Int.java	Problems 23 @ Javadoc O items Description	Declaration	Path	Location	Type

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

🖸 • 🔛 🕼 : 發 • 🚺 • 🚰 • '	∿	• 🗁 🔗 • 🥭 🔍 🔌 🔤 • 🖓 • 🖏 ♦ 🗢 • ⇒ • 👘 (Quick Access	费
🛱 Package Explorer 🔀		- E	🗆 🗄 Outlin 🖾 🗖	
E 4	\$ 50 ▼		5	~
▼ 🔁 LeJavaProject		File Operation	An outline is not	
JRE System Library			available.	
Referenced Librarie	Select how files	should be imported into the project:		
	💿 Copy file	S		
	C Link to fi	les		
	Creat	te link locations relative to: PROJECT_LOC		
<u>c</u>	Configure Drag	and Drop Settings		
		1.0000		
(?	Cancer		
	<i>?</i>			
	?			
	•			
	0	Problems X @ Javadoc Q Declaration		• 6
	0	Problems X @ Javadoc Q Declaration 0 items		° E
	0	Problems X @ Javadoc Q Declaration 0 items Description	the second seco	• =
	ý	Problems X @ Javadoc & Declaration 0 items Description	ting the second	• =

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.

•••	j eclipse-workspace - LeJavaProject/src/HandyInt.java - Eclipse							
📬 • 🗟 🕼 र 💽 • 💁 • 😭 • 😂 /	ダ・参: 学 🎜 ② 副 圓 面: ④・ № 泉・茶・♡ 수・◇・ Ouick Access 🔮 👹							
🚦 Package Explorer 🐹 📄 😒 🐨 🗖 🗖	🔐 Handyint.java 🕱 🧧 📴 Outline 🕱 👘 🗖							
▼ 🔂 LeJavaProject	📴 1 package gragramming.set6.arithmetic;							
JRE System Library [JavaSE-1.8]	2 3 · /** Ing. set6.arithmetic " does not match the expected package "" ning.set6.arithmetic							
src	4 * An integer that provid 2 quick fixes available:							
Handvint.iava	5 */							
Referenced Libraries	7 /** The integer repre 🗄 Remove package declaration 'package programming.set6.arithmetic' Int(int)							
	8 private int theInt; D int							
	100 /** andyInt) : HandyInt							
	11 * Constructs a new handy integer representing the given int. Sub(HandyInt): HandyInt							
	12 ● ▲ toString(): String							
	14 * the integer to be represented by this instance.							
	15 s/ 16@ public HandyInt(int theInt) {							
	17 this.theInt = theInt;							
	280 /**							
	21 * Constructs a new handy integer representing the integer represented by							
	22 * the given handy threef.							
	24 ★ @param handyInt 25 ★ the handy integer to intialize the new object with. 26 ★/							
	270 public HandyInt (HandyInt) {							
	28 this.theInt = handyInt.theInt;							
	👔 Problems 🕃 @ Javadoc 🔯 Declaration 🏦 😨 🍷 🗖							
	1 error, 0 warnings, 0 others							
	Description A Resource Path Location Type							
	P Servis (Titem)							
	Writable Smart Insert 1:1							