Coding With Eclipse

Pratical Tips and Tricks

Pr. Olivier Gruber

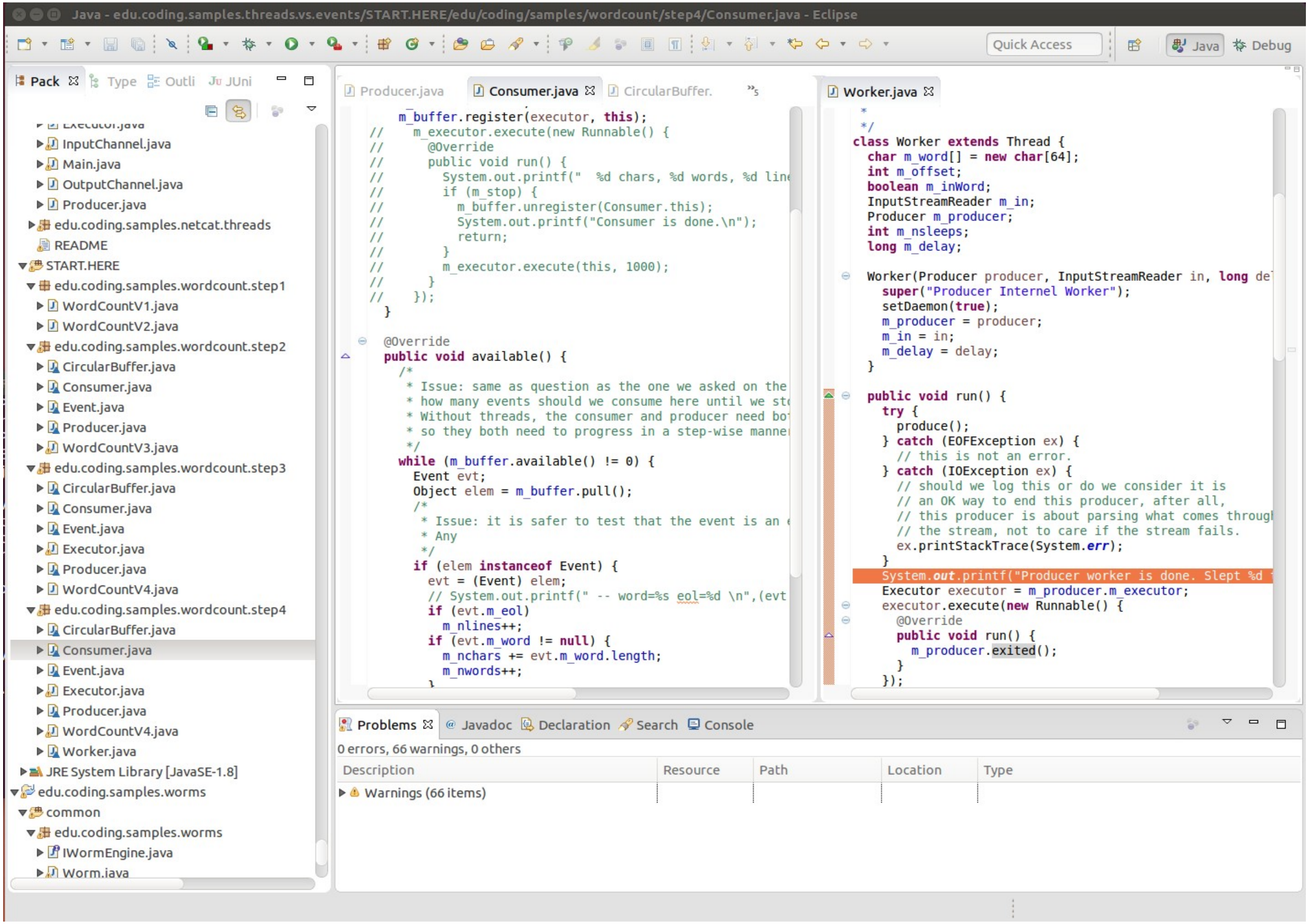
olivier.gruber@imag.fr

Full-time Professor
Université Grenoble Alpes

Outline

- The Workbench and workspace basics
  - Perspectives and views
  - Workspaces and projects
- Developing in Java
  - Java project basics
  - Coding key features
  - Debugging key features
  - Testing basics
  - Performance basics

Eclipse Workbench – The Java Perspective



```
Executor.java
InputChannel.java
Main.java
OutputChannel.java
Producer.java
edu.coding.samples.netcat.threads
README
START.HERE
  edu.coding.samples.wordcount.step1
    WordCountV1.java
    WordCountV2.java
  edu.coding.samples.wordcount.step2
    CircularBuffer.java
    Consumer.java
    Event.java
    Producer.java
    WordCountV3.java
  edu.coding.samples.wordcount.step3
    CircularBuffer.java
    Consumer.java
    Event.java
    Executor.java
    Producer.java
    WordCountV4.java
  edu.coding.samples.wordcount.step4
    CircularBuffer.java
    Consumer.java
    Event.java
    Executor.java
    Producer.java
    WordCountV4.java
    Worker.java
JRE System Library [JavaSE-1.8]
edu.coding.samples.worms
  common
    edu.coding.samples.worms
      IWormEngine.java
      Worm.java
```

```
      m_buffer.register(executor, this);
//      m_executor.execute(new Runnable() {
//        @Override
//        public void run() {
//          System.out.printf("  %d chars, %d words, %d line
//          if (m_stop) {
//            m_buffer.unregister(Consumer.this);
//            System.out.printf("Consumer is done.\n");
//            return;
//          }
//          m_executor.execute(this, 1000);
//        }
//      });
    }

    @Override
    public void available() {
      /*
       * Issue: same as question as the one we asked on the
       * how many events should we consume here until we sto
       * Without threads, the consumer and producer need bot
       * so they both need to progress in a step-wise manne
       */
      while (m_buffer.available() != 0) {
        Event evt;
        Object elem = m_buffer.pull();
        /*
         * Issue: it is safer to test that the event is an e
         * Any
         */
        if (elem instanceof Event) {
          evt = (Event) elem;
          // System.out.printf(" -- word=%s eol=%d \n",(evt
          if (evt.m_eol)
            m_nlines++;
          if (evt.m_word != null) {
            m_nchars += evt.m_word.length;
            m_nwords++;
```

```
 *
 */
class Worker extends Thread {
  char m_word[] = new char[64];
  int m_offset;
  boolean m_inWord;
  InputStreamReader m_in;
  Producer m_producer;
  int m_nsleeps;
  long m_delay;

  Worker(Producer producer, InputStreamReader in, long de
    super("Producer Internel Worker");
    setDaemon(true);
    m_producer = producer;
    m_in = in;
    m_delay = delay;
  }

  public void run() {
    try {
      produce();
    } catch (EOFException ex) {
      // this is not an error.
    } catch (IOException ex) {
      // should we log this or do we consider it is
      // an OK way to end this producer, after all,
      // this producer is about parsing what comes throug
      // the stream, not to care if the stream fails.
      ex.printStackTrace(System.err);
    }
    System.out.printf("Producer worker is done. Slept %d
    Executor executor = m_producer.m_executor;
    executor.execute(new Runnable() {
      @Override
      public void run() {
        m_producer.exited();
      }
    });
```

0 errors, 66 warnings, 0 others

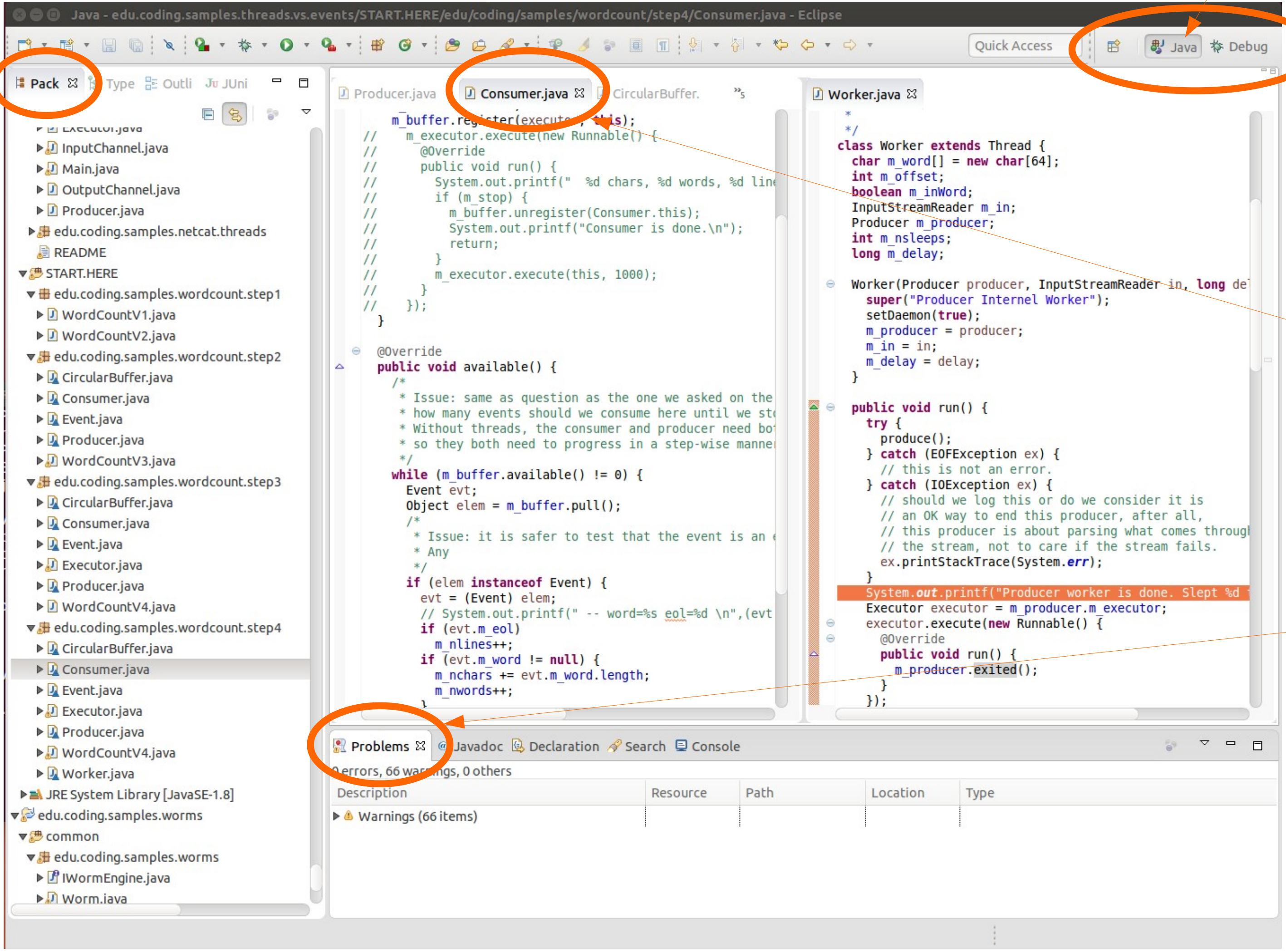
Description	Resource	Path	Location	Type
Warnings (66 items)				

Eclipse Workbench

- Perspectives
  - A **perspective** is a activity-oriented layout of **views** on the screen
    - Java perspective or C/C++ perspective
    - Debugging perspective
  - The screen shows the current perspective
    - But you can have multiple perspectives open
    - You can switch between them
    - Menus and toolbars change when switching between perspectives
    - To open a perspective, use menus: Window → Perspective
  - A perspective can host any view
    - To open a view, use menus: Window → Show View
  - A perspective layout is a grid
    - Views can be moved around by drag and drop

Eclipse Workbench

- Each Perspective
  - A activity-oriented organization of views than can be cusomized



```
▸ InputChannel.java
▸ Main.java
▸ OutputChannel.java
▸ Producer.java
▸ edu.coding.samples.netcat.threads
  README
▾ START.HERE
 ▾ edu.coding.samples.wordcount.step1
   ▸ WordCountV1.java
   ▸ WordCountV2.java
 ▾ edu.coding.samples.wordcount.step2
   ▸ CircularBuffer.java
   ▸ Consumer.java
   ▸ Event.java
   ▸ Producer.java
   ▸ WordCountV3.java
 ▾ edu.coding.samples.wordcount.step3
   ▸ CircularBuffer.java
   ▸ Consumer.java
   ▸ Event.java
   ▸ Executor.java
   ▸ Producer.java
   ▸ WordCountV4.java
 ▾ edu.coding.samples.wordcount.step4
   ▸ CircularBuffer.java
   ▸ Consumer.java
   ▸ Event.java
   ▸ Executor.java
   ▸ Producer.java
   ▸ WordCountV4.java
   ▸ Worker.java
▸ JRE System Library [JavaSE-1.8]
▾ edu.coding.samples.worms
 ▾ common
  ▾ edu.coding.samples.worms
    ▸ IWormEngine.java
    ▸ Worm.java
```

```java
    m_buffer.register(executor, this);
//    m_executor.execute(new Runnable() {
//      @Override
//      public void run() {
//        System.out.printf("  %d chars, %d words, %d line
//        if (m_stop) {
//          m_buffer.unregister(Consumer.this);
//          System.out.printf("Consumer is done.\n");
//          return;
//        }
//        m_executor.execute(this, 1000);
//      }
//    });
  }

  @Override
  public void available() {
    /*
     * Issue: same as question as the one we asked on the
     * how many events should we consume here until we sto
     * Without threads, the consumer and producer need bot
     * so they both need to progress in a step-wise manne
     */
    while (m_buffer.available() != 0) {
      Event evt;
      Object elem = m_buffer.pull();
      /*
       * Issue: it is safer to test that the event is an e
       * Any
       */
      if (elem instanceof Event) {
        evt = (Event) elem;
        // System.out.printf(" -- word=%s eol=%d \n",(evt
        if (evt.m_eol)
          m_nlines++;
        if (evt.m_word != null) {
          m_nchars += evt.m_word.length;
          m_nwords++;
        }
```

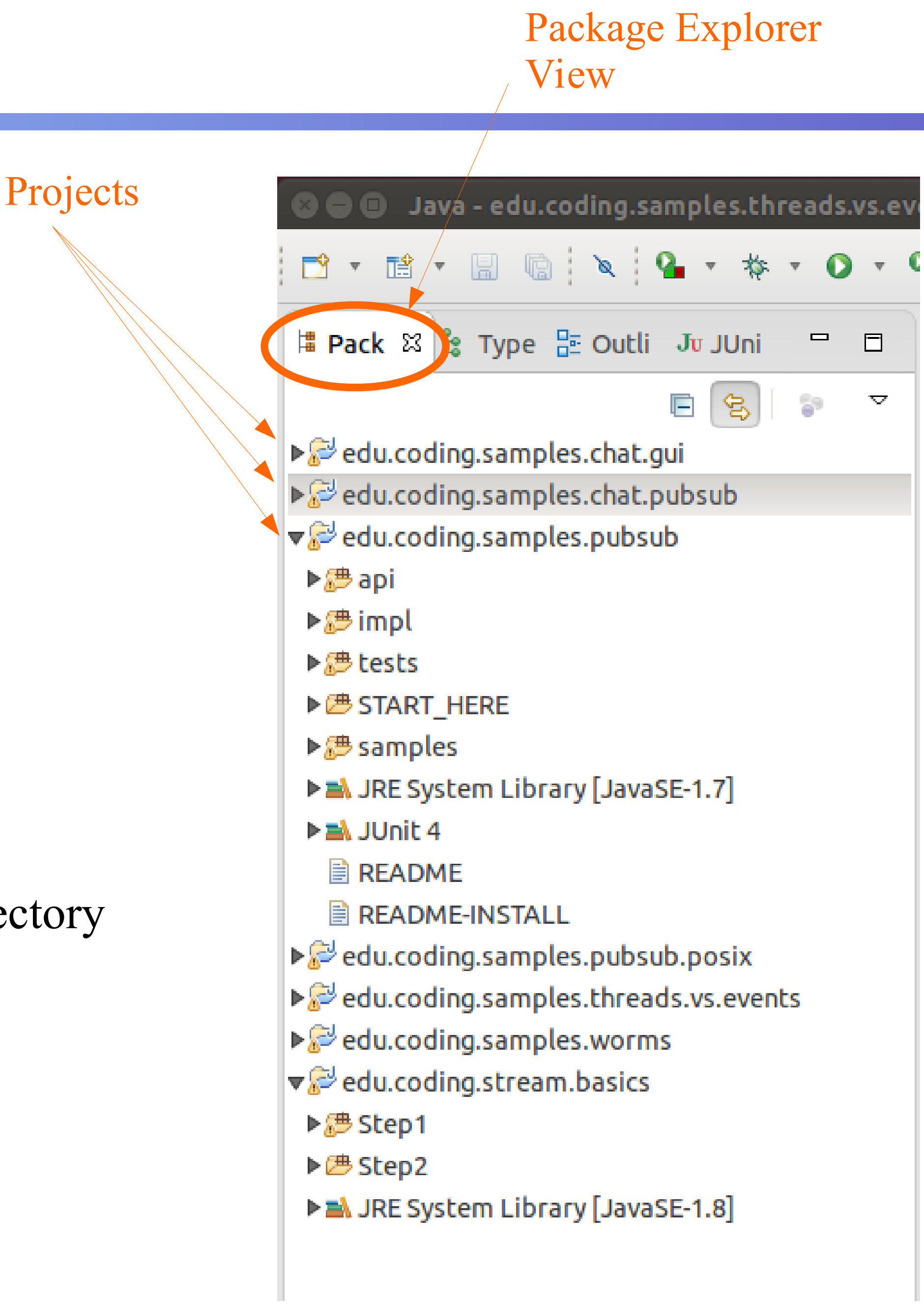
```java
 *
 */
class Worker extends Thread {
  char m_word[] = new char[64];
  int m_offset;
  boolean m_inWord;
  InputStreamReader m_in;
  Producer m_producer;
  int m_nsleeps;
  long m_delay;

  Worker(Producer producer, InputStreamReader in, long de
    super("Producer Internel Worker");
    setDaemon(true);
    m_producer = producer;
    m_in = in;
    m_delay = delay;
  }

  public void run() {
    try {
      produce();
    } catch (EOFException ex) {
      // this is not an error.
    } catch (IOException ex) {
      // should we log this or do we consider it is
      // an OK way to end this producer, after all,
      // this producer is about parsing what comes throug
      // the stream, not to care if the stream fails.
      ex.printStackTrace(System.err);
    }
    System.out.printf("Producer worker is done. Slept %d
    Executor executor = m_producer.m_executor;
    executor.execute(new Runnable() {
      @Override
      public void run() {
        m_producer.exited();
      }
    });
```

0 errors, 66 warnings, 0 others

Description	Resource	Path	Location	Type
▸ Warnings (66 items)				

Eclipse Workspace

- Workspace
  - Corresponds to a directory in the file system
    - Called the **Workspace directory**
  - Contains a set of projects
    - Each project corresponds to a subdirectory
    - Projects may also be linked to workspace
- Created Project
  - Created as a subdirectory to the workspace
- Imported Project
  - From another location than the Workspace directory
- Each project has a nature
  - One project may be a Java project
  - While another may be C/C++ project



```
▶ edu.coding.samples.chat.gui
▶ edu.coding.samples.chat.pubsub
▼ edu.coding.samples.pubsub
  ▶ api
  ▶ impl
  ▶ tests
  ▶ START_HERE
  ▶ samples
  ▶ JRE System Library [JavaSE-1.7]
  ▶ JUnit 4
    README
    README-INSTALL
▶ edu.coding.samples.pubsub.posix
▶ edu.coding.samples.threads.vs.events
▶ edu.coding.samples.worms
▼ edu.coding.stream.basics
  ▶ Step1
  ▶ Step2
  ▶ JRE System Library [JavaSE-1.8]
```

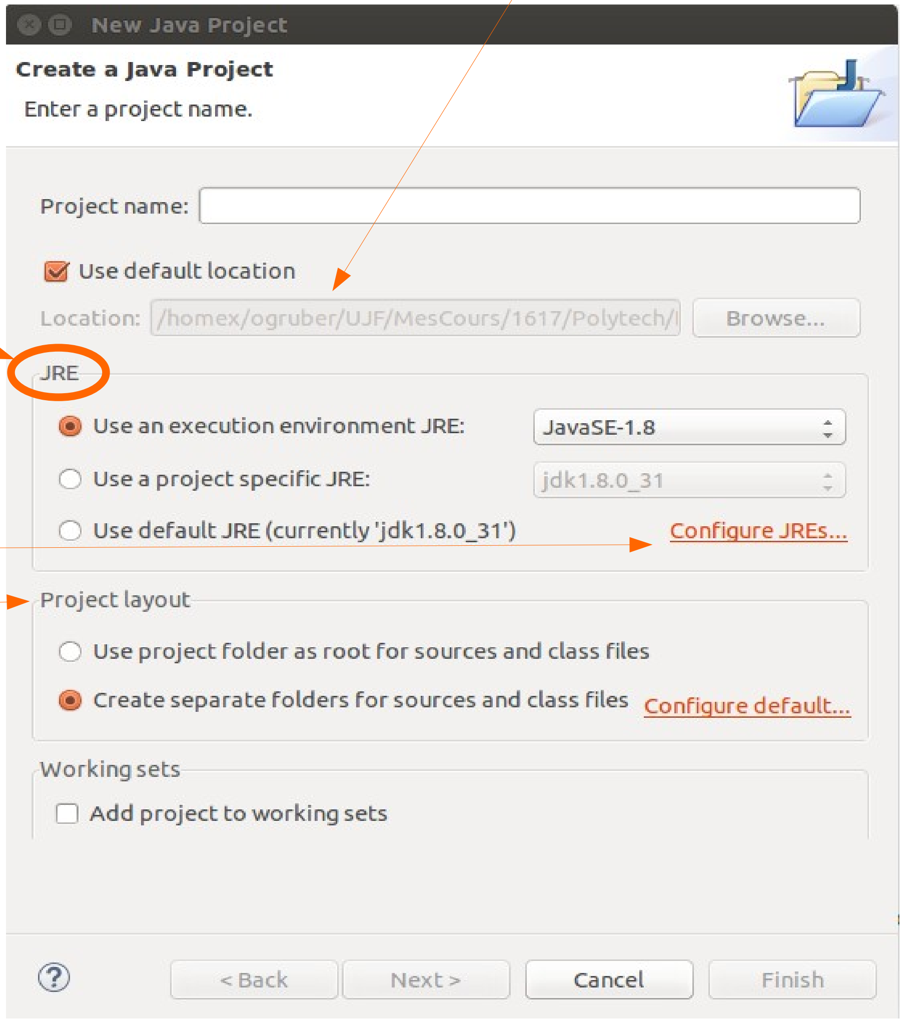
Java Project

- Process Outline
  - Create the project with a name
  - Align compatibility levels
    - Choose a Java level for your code (Java 1.8 for example)
    - Setup Java compiler to a compatible level (Java 1.8)
    - Choose an installed Java platform that is installed (JavaSE-1.8)
  - A bit of history about Java
    - The Java language exist in multiple versions – from 1.0 up to 1.8
    - The Java platform exist also in multiple versions – from 1.0 up to 1.8
    - The Java platform is a runtime to execute Java programs
      - It can be a Java Runtime Enviroment (JRE) or a Java Developer Kit (JDK)
      - As a developer, always **install a JDK**, it has more tools that you will need
    - The platform comes in different profiles
      - Standard Edition (JavaSE), Enterprise Edition (JavaEE), Minimum Edition (J2ME)
      - **Install JavaSE** on a desktop/laptop

Eclipse Workspace – New Java Project

- Creating a new Java project
  - Menu: File → New → Java Project
  - Choose a name
  - Choose a location
- Choose an execution environment
  - Must be compatible with the code
  - Choose the latest is ususally safe
  - You may have to configure JREs
- Choose a layout
  - Better to separate sources and class files

Default location in the Worskpace directory


New Java Project

Create a Java Project

Enter a project name.

Project name:

[✓] Use default location

Location: /homex/ogruber/UJF/MesCours/1617/Polytech/

Browse...

JRE

[● Use an execution environment JRE: JavaSE-1.8]

○ Use a project specific JRE: jdk1.8.0_31

○ Use default JRE (currently 'jdk1.8.0_31')

Configure JREs...

Project layout

○ Use project folder as root for sources and class files

[● Create separate folders for sources and class files]

Configure default...

Working sets

☐ Add project to working sets

< Back | Next > | Cancel | Finish

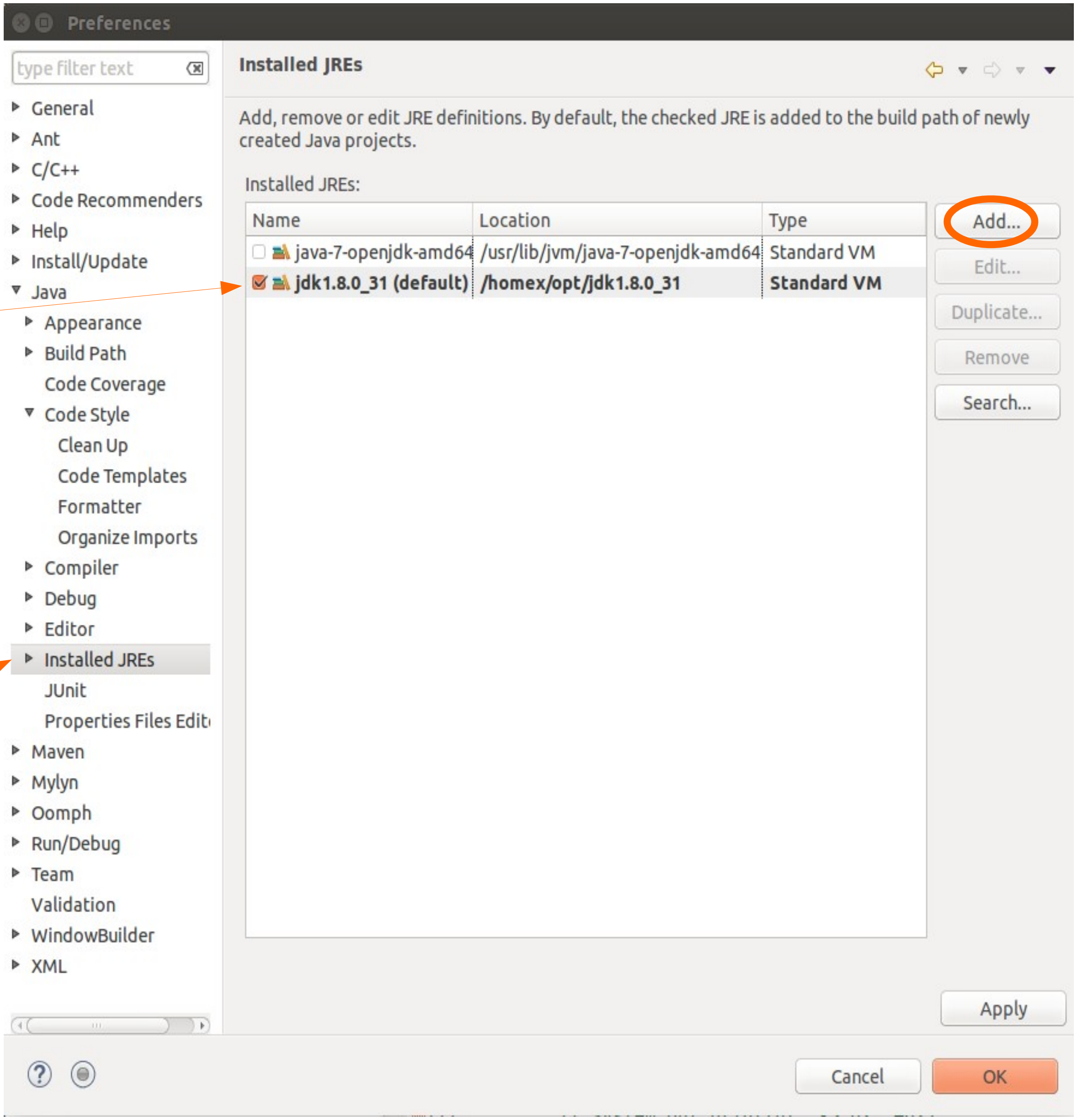
Eclipse Workspace – Java Runtime Environment Setup

- Configuring JREs
  - Installed JREs and JDKs
  - **Always install a JDK**
  - **From Oracle website**
  - Make it the default

To get this dialog, use menus

Window → Preferences

Drill down to
Java
Installed JREs



Installed JREs

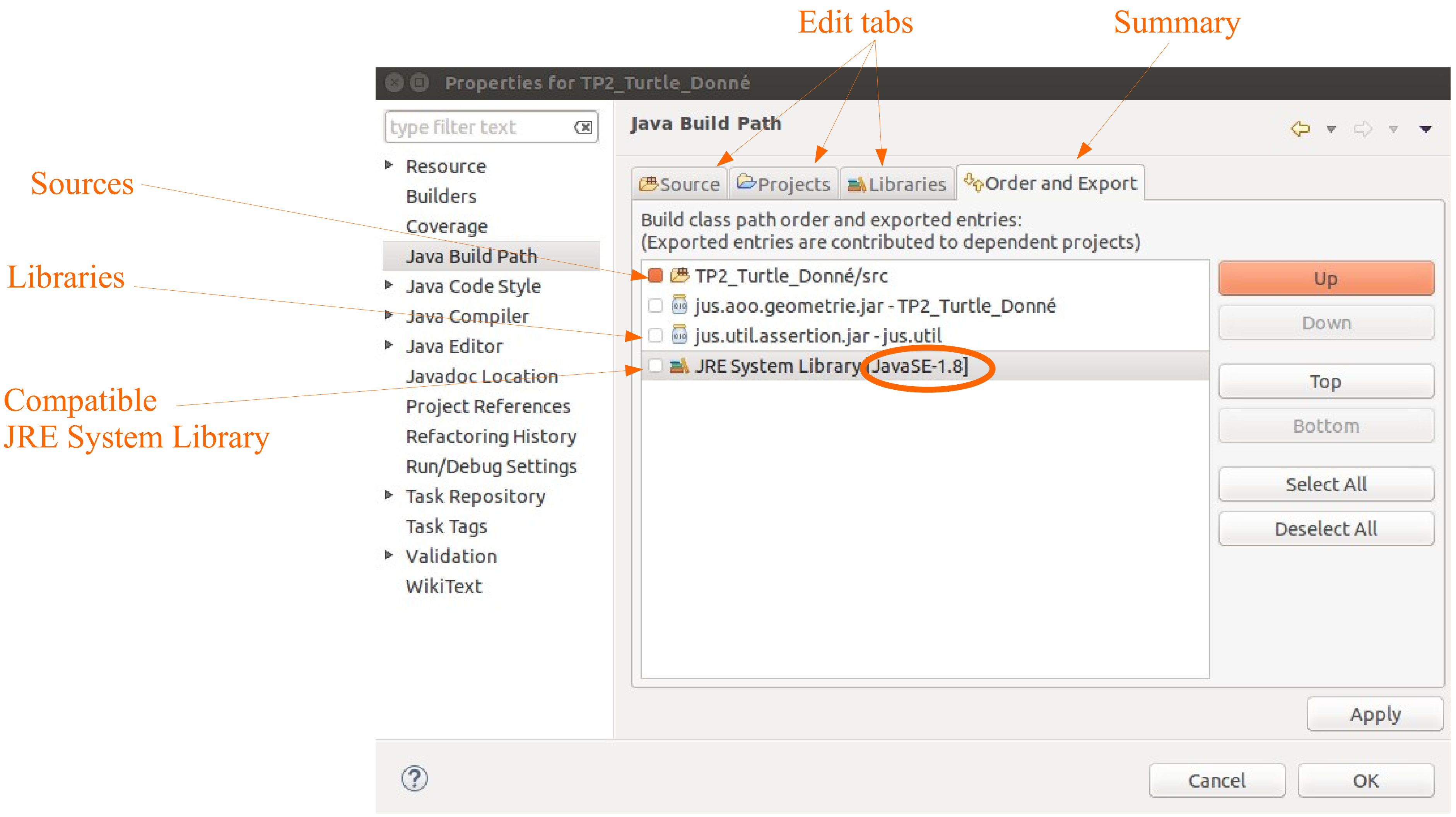
Add, remove or edit JRE definitions. By default, the checked JRE is added to the build path of newly created Java projects.

Installed JREs:

Name	Location	Type
☐ java-7-openjdk-amd64	/usr/lib/jvm/java-7-openjdk-amd64	Standard VM
☑ **jdk1.8.0_31 (default)**	**/homex/opt/jdk1.8.0_31**	**Standard VM**

Eclipse Workspace – Java Project Setup

- Setup your **Java Build Path**
  - Right-click on a project in the **Package Explorer** view → Properties (Alt-Enter)



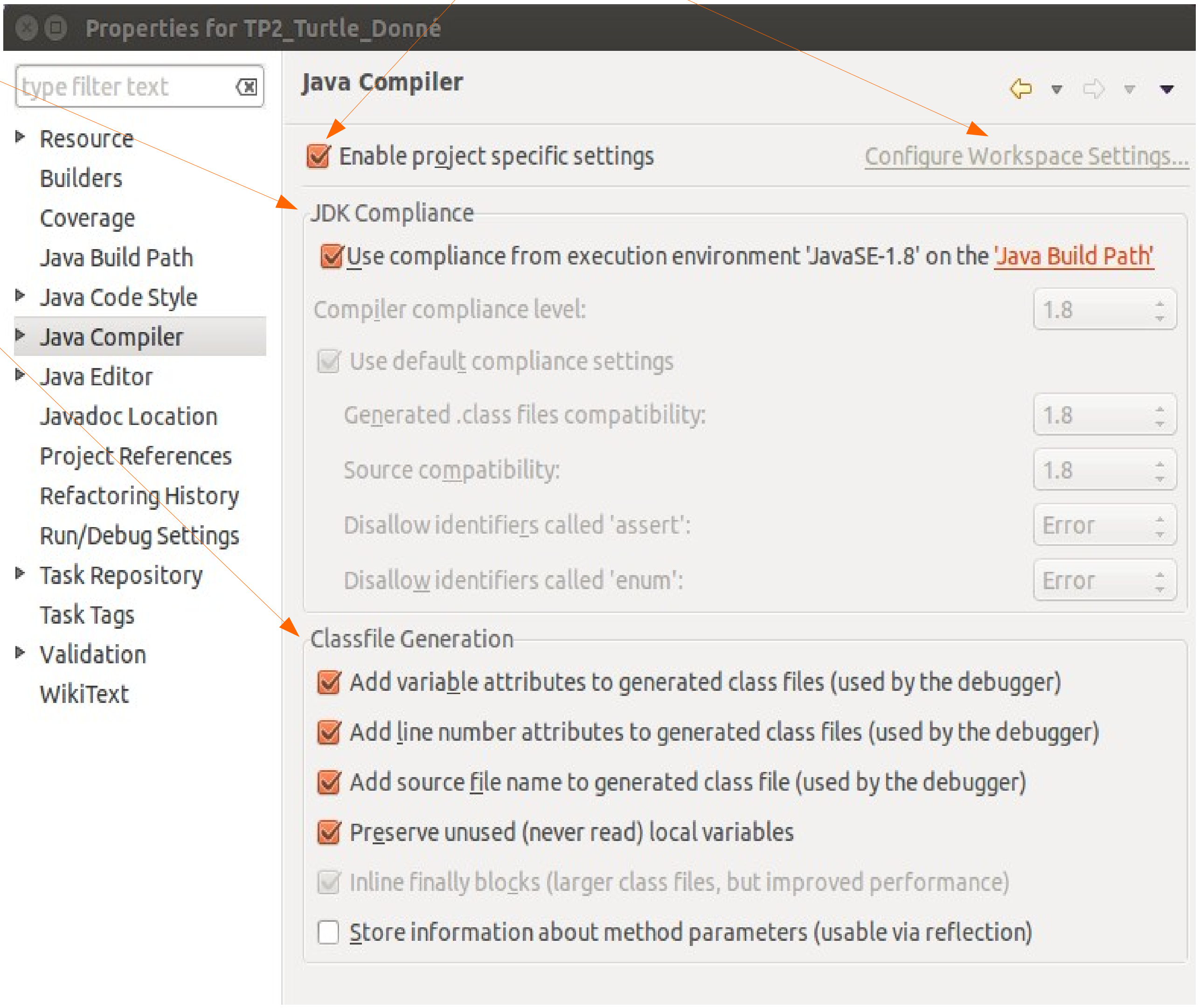
Build class path order and exported entries:
(Exported entries are contributed to dependent projects)

TP2_Turtle_Donné/src
jus.aoo.geometrie.jar - TP2_Turtle_Donné
jus.util.assertion.jar - jus.util
JRE System Library [JavaSE-1.8]

Eclipse Workspace – Java Project Setup

- Setup the Java Compiler
  - Choose JDK compliance
  - Select debug infos

Two levels of settings



☑ Enable project specific settings

JDK Compliance

☑ Use compliance from execution environment 'JavaSE-1.8' on the 'Java Build Path'

Compiler compliance level: 1.8

☑ Use default compliance settings

Generated .class files compatibility: 1.8

Source compatibility: 1.8

Disallow identifiers called 'assert': Error

Disallow identifiers called 'enum': Error

Classfile Generation

☑ Add variable attributes to generated class files (used by the debugger)

☑ Add line number attributes to generated class files (used by the debugger)

☑ Add source file name to generated class file (used by the debugger)

☑ Preserve unused (never read) local variables

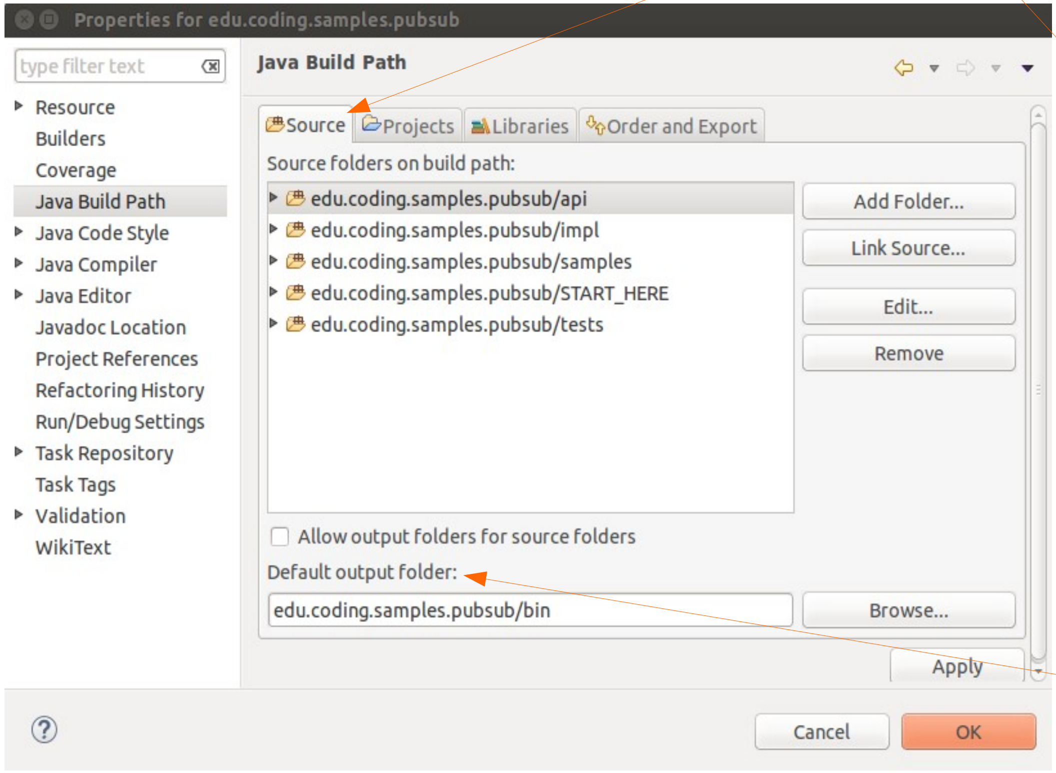
☑ Inline finally blocks (larger class files, but improved performance)

☐ Store information about method parameters (usable via reflection)

Eclipse Workspace – Java Project Setup

- Projects
  - You can use multiple source directories
  - But usually one output folder

Where your code is



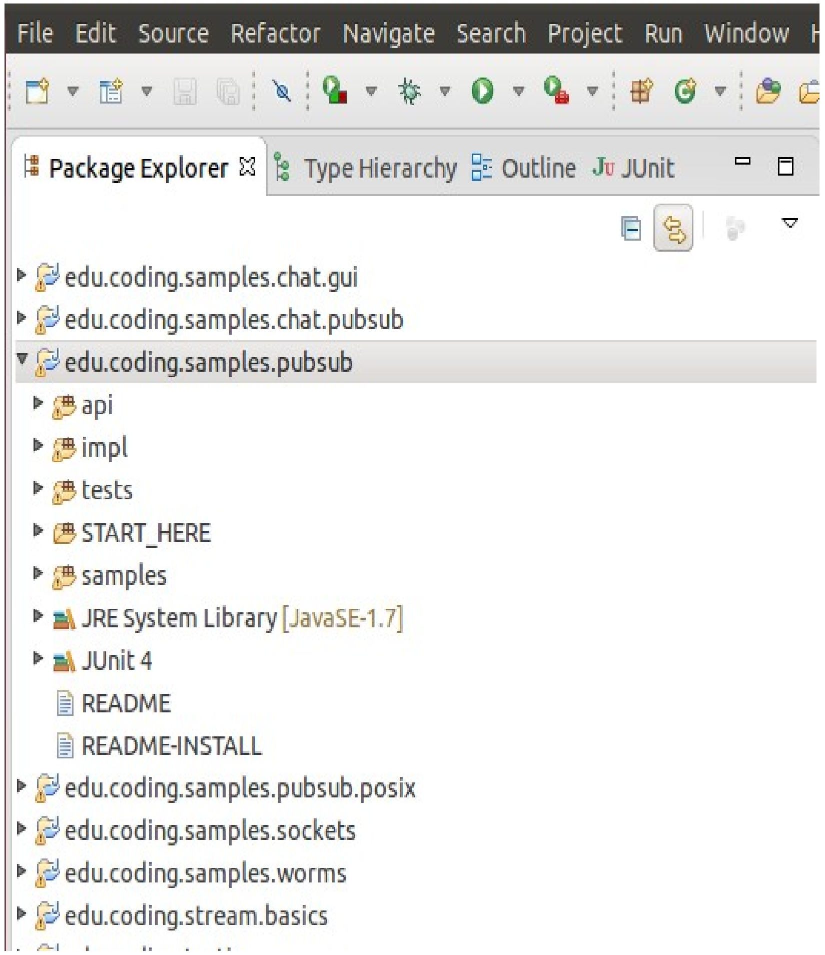
Properties for edu.coding.samples.pubsub

Java Build Path

Source folders on build path:
- edu.coding.samples.pubsub/api
- edu.coding.samples.pubsub/impl
- edu.coding.samples.pubsub/samples
- edu.coding.samples.pubsub/START_HERE
- edu.coding.samples.pubsub/tests

☐ Allow output folders for source folders

Default output folder: edu.coding.samples.pubsub/bin



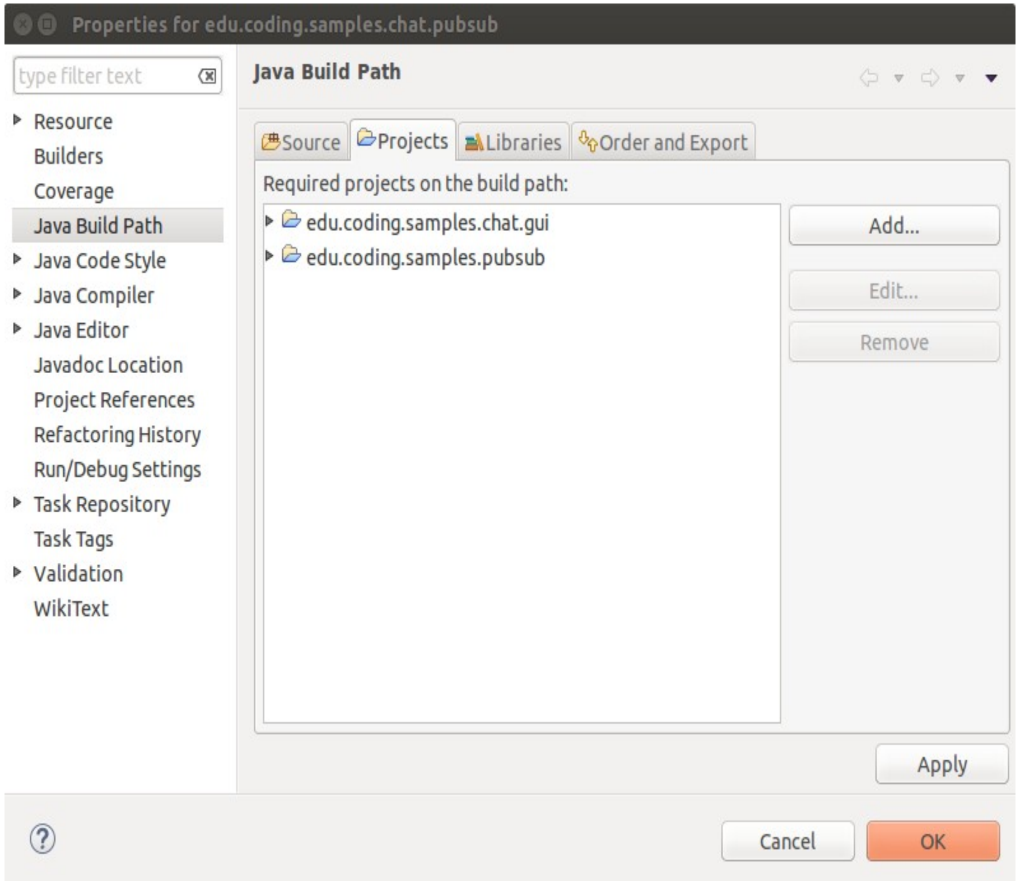
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
Type Hierarchy
Outline
JUnit

- edu.coding.samples.chat.gui
- edu.coding.samples.chat.pubsub
- edu.coding.samples.pubsub
  - api
  - impl
  - tests
  - START_HERE
  - samples
  - JRE System Library [JavaSE-1.7]
  - JUnit 4
  - README
  - README-INSTALL
- edu.coding.samples.pubsub.posix
- edu.coding.samples.sockets
- edu.coding.samples.worms
- edu.coding.stream.basics

Where the compiler generates class files

Eclipse Workspace – Java Project Setup

- Projects
  - Can depend on each others
  - Cycles are possible (advanced option)

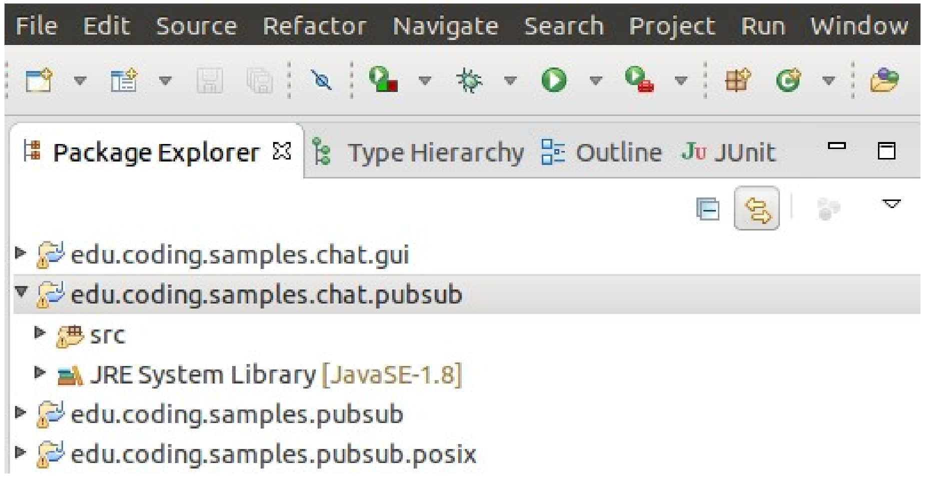


Properties for edu.coding.samples.chat.pubsub

Java Build Path

Required projects on the build path:

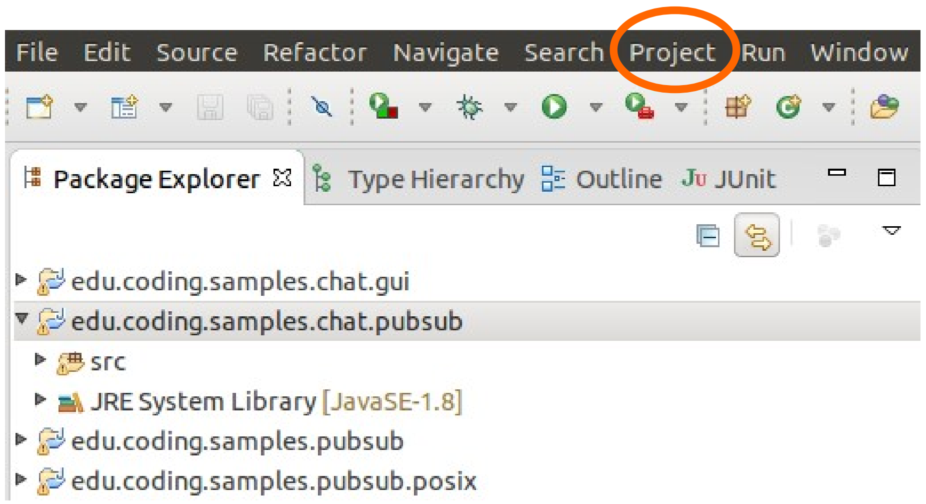
- edu.coding.samples.chat.gui
- edu.coding.samples.pubsub

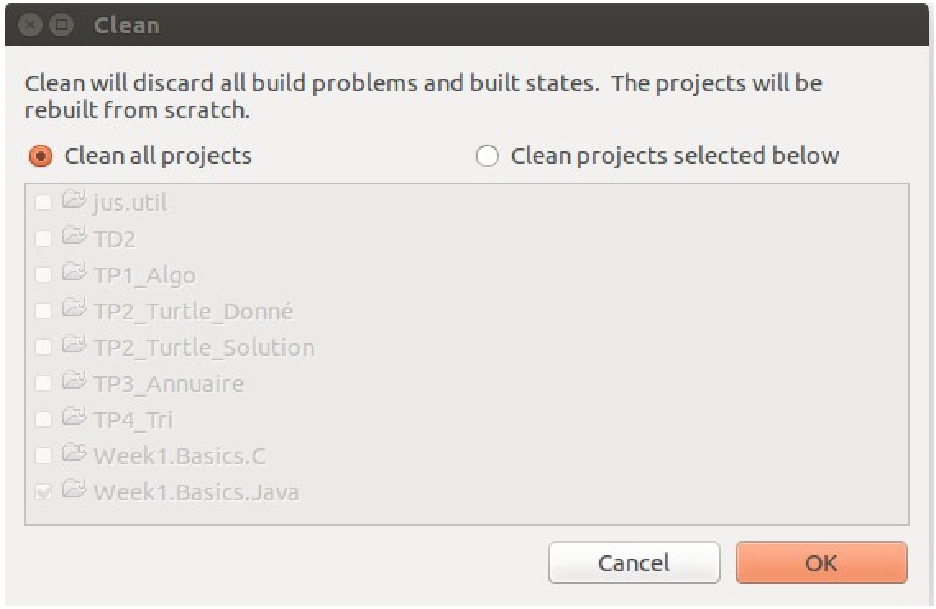


- edu.coding.samples.chat.gui
- edu.coding.samples.chat.pubsub
  - src
  - JRE System Library [JavaSE-1.8]
- edu.coding.samples.pubsub
- edu.coding.samples.pubsub.posix

Eclipse Workspace – Java Project Build

- Project Menu
  - Automatic build is the default
  - Very accurate/efficient in Java
- Project Clean
  - Sometimes cleaning is necessary
  - Alt-P → Clean


File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
Type Hierarchy
Outline
JUnit
edu.coding.samples.chat.gui
edu.coding.samples.chat.pubsub
src
JRE System Library [JavaSE-1.8]
edu.coding.samples.pubsub
edu.coding.samples.pubsub.posix

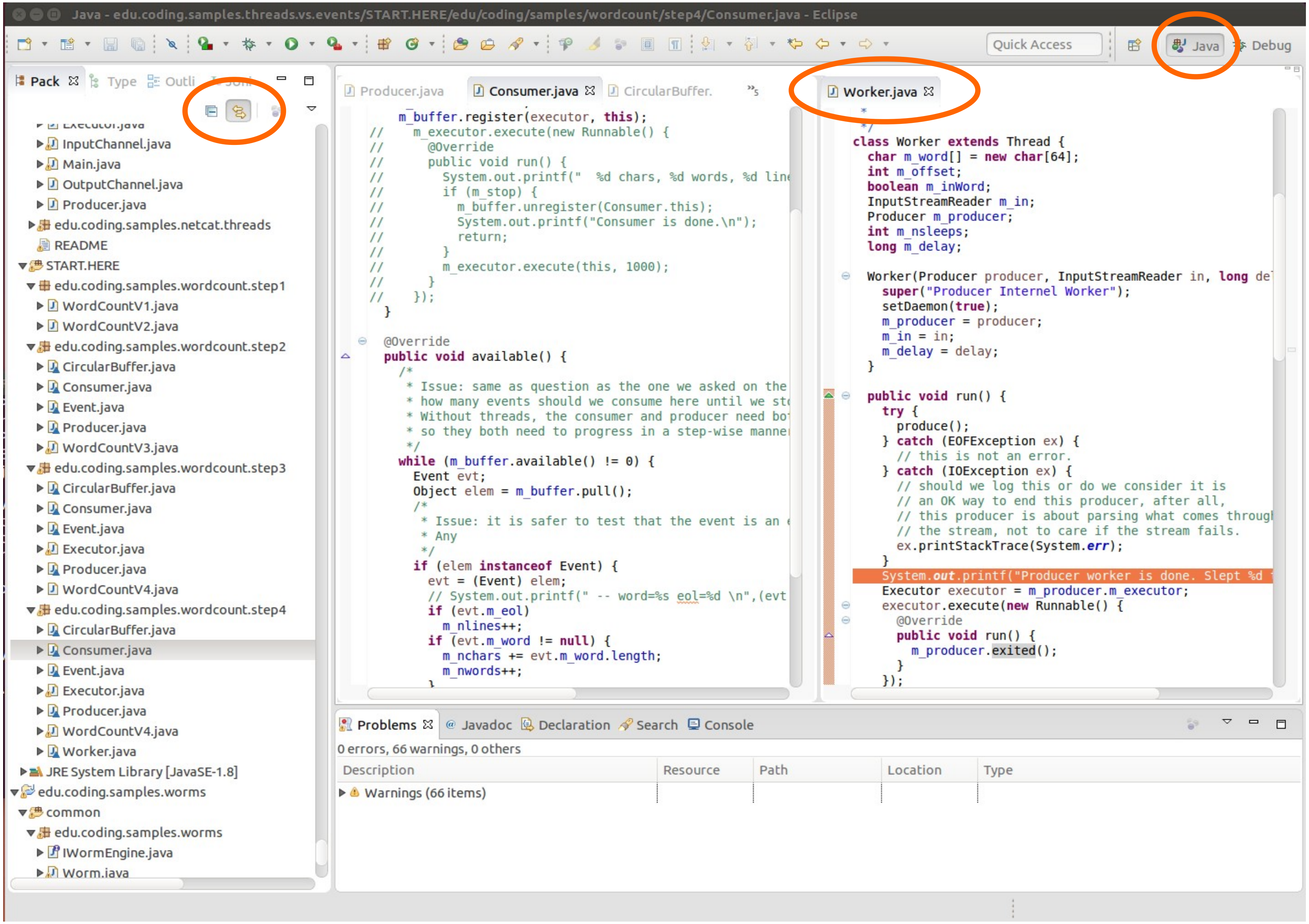

Clean

Clean will discard all build problems and built states. The projects will be rebuilt from scratch.

[✓ Clean all projects] Clean projects selected below

- jus.util
- TD2
- TP1_Algo
- TP2_Turtle_Donné
- TP2_Turtle_Solution
- TP3_Annuaire
- TP4_Tri
- Week1.Basics.C
- ✓ Week1.Basics.Java

Eclipse – Java Coding



```
Executor.java
InputChannel.java
Main.java
OutputChannel.java
Producer.java
edu.coding.samples.netcat.threads
README
START.HERE
  edu.coding.samples.wordcount.step1
    WordCountV1.java
    WordCountV2.java
  edu.coding.samples.wordcount.step2
    CircularBuffer.java
    Consumer.java
    Event.java
    Producer.java
    WordCountV3.java
  edu.coding.samples.wordcount.step3
    CircularBuffer.java
    Consumer.java
    Event.java
    Executor.java
    Producer.java
    WordCountV4.java
  edu.coding.samples.wordcount.step4
    CircularBuffer.java
    Consumer.java
    Event.java
    Executor.java
    Producer.java
    WordCountV4.java
    Worker.java
JRE System Library [JavaSE-1.8]
edu.coding.samples.worms
  common
    edu.coding.samples.worms
      IWormEngine.java
      Worm.java
```

```
    m_buffer.register(executor, this);
//    m_executor.execute(new Runnable() {
//      @Override
//      public void run() {
//        System.out.printf("  %d chars, %d words, %d line
//        if (m_stop) {
//          m_buffer.unregister(Consumer.this);
//          System.out.printf("Consumer is done.\n");
//          return;
//        }
//        m_executor.execute(this, 1000);
//      }
//    });
  }

  @Override
  public void available() {
    /*
     * Issue: same as question as the one we asked on the
     * how many events should we consume here until we st
     * Without threads, the consumer and producer need bo
     * so they both need to progress in a step-wise manne
     */
    while (m_buffer.available() != 0) {
      Event evt;
      Object elem = m_buffer.pull();
      /*
       * Issue: it is safer to test that the event is an e
       * Any
       */
      if (elem instanceof Event) {
        evt = (Event) elem;
        // System.out.printf(" -- word=%s eol=%d \n",(evt
        if (evt.m_eol)
          m_nlines++;
        if (evt.m_word != null) {
          m_nchars += evt.m_word.length;
          m_nwords++;
        }
```

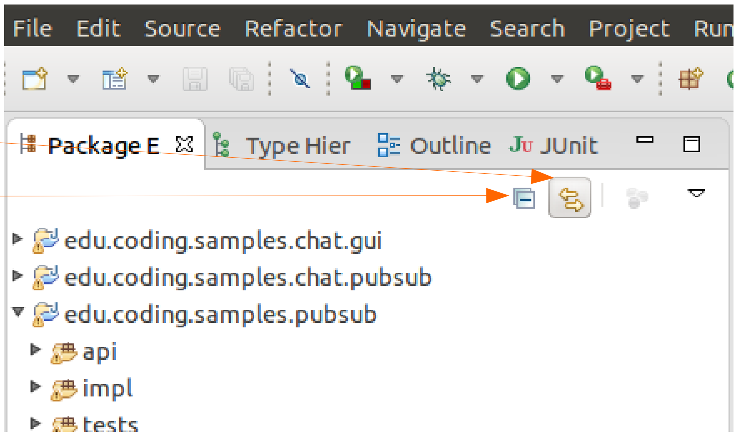
```
 *
 */
class Worker extends Thread {
  char m_word[] = new char[64];
  int m_offset;
  boolean m_inWord;
  InputStreamReader m_in;
  Producer m_producer;
  int m_nsleeps;
  long m_delay;

  Worker(Producer producer, InputStreamReader in, long de
    super("Producer Internel Worker");
    setDaemon(true);
    m_producer = producer;
    m_in = in;
    m_delay = delay;
  }

  public void run() {
    try {
      produce();
    } catch (EOFException ex) {
      // this is not an error.
    } catch (IOException ex) {
      // should we log this or do we consider it is
      // an OK way to end this producer, after all,
      // this producer is about parsing what comes throug
      // the stream, not to care if the stream fails.
      ex.printStackTrace(System.err);
    }
    System.out.printf("Producer worker is done. Slept %d 
    Executor executor = m_producer.m_executor;
    executor.execute(new Runnable() {
      @Override
      public void run() {
        m_producer.exited();
      }
    });
```

0 errors, 66 warnings, 0 others

Description	Resource	Path	Location	Type
Warnings (66 items)				

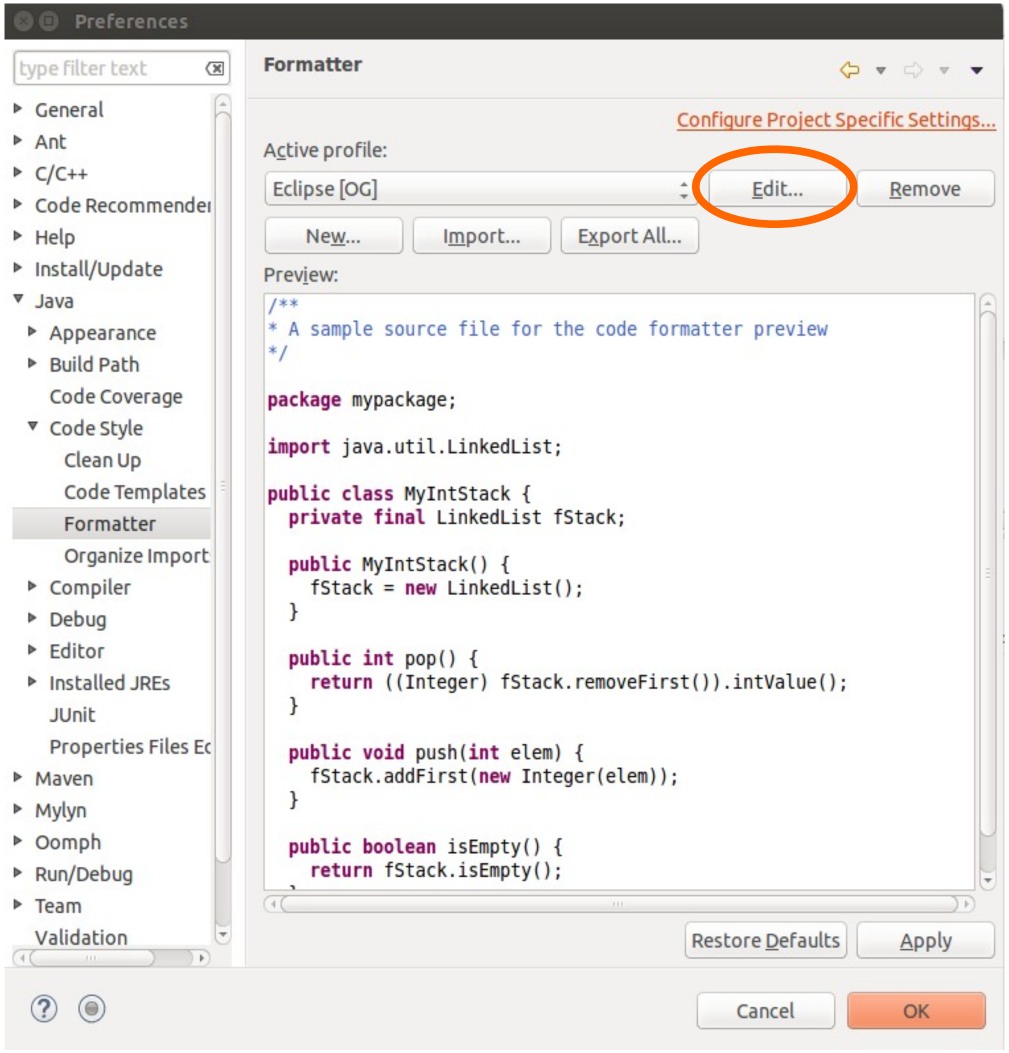
Eclipse – Java Coding

- Java Editing
  - Multi-editors (drag-and-drop, sort of a grid layout)
  - Split editors (Alt-W-E or Ctr-_ Ctr-{})
  - Tab management
    - Close current tab (Ctr-W)
    - Close all tabs (Ctr-Shift-W)
    - Change tab (Ctr-PgUp, Ctr-PgDown)
    - Choose tab (Ctr-E + regular expression)
- Navigator
  - Synchronized navigation
  - Collapse all



Eclipse – Java Coding

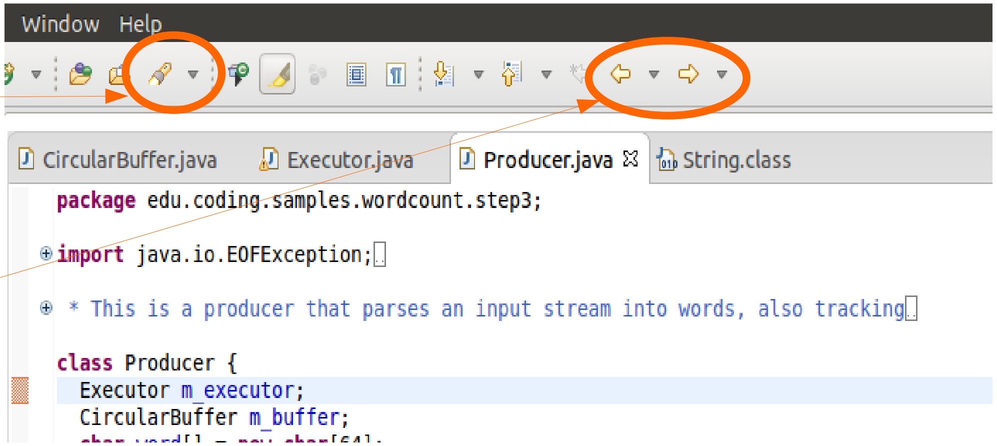
- Java Editing
  - Setup your preferences (Alt-W-P)
    - Edit your Java Formatter profile
      - Java → Code Style → Formatter
      - No tabs, 2 spaces
      - Comment reformattting (disable/enable)
      - Choose a coding style
  - Pretty printing
    - Pretty printing (**Ctr-Shift-F**)
    - Organize imports (**Ctrl-Shift-O**)
- Export/Import preferences
  - Across your workspaces
  - Across team members



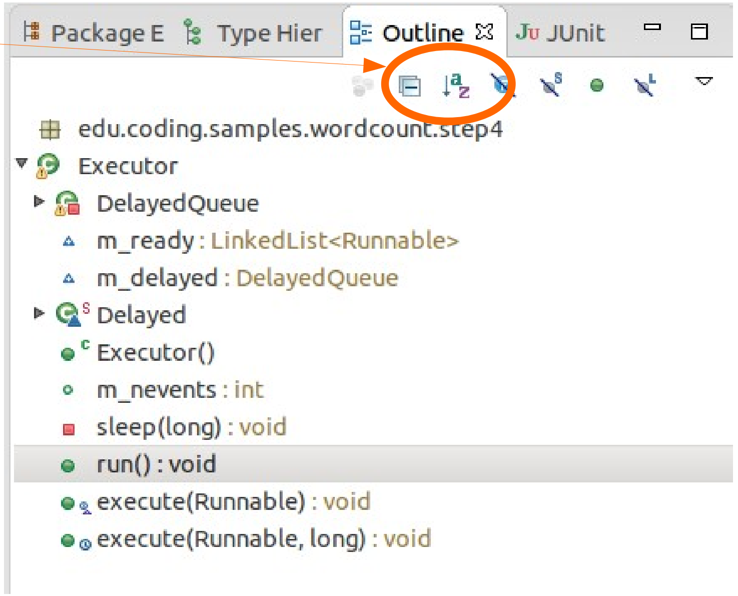
```
/**
* A sample source file for the code formatter preview
*/

package mypackage;

import java.util.LinkedList;

public class MyIntStack {
  private final LinkedList fStack;

  public MyIntStack() {
    fStack = new LinkedList();
  }

  public int pop() {
    return ((Integer) fStack.removeFirst()).intValue();
  }

  public void push(int elem) {
    fStack.addFirst(new Integer(elem));
  }

  public boolean isEmpty() {
    return fStack.isEmpty();
```

Eclipse – Java Coding

- Java code navigation
  - Search engine
  - Class search (**Ctlr-Shift-T + regexp**)
  - Select and hit F3
  - Use arrows with drop-down
  - References (right-click or Ctr-Shift-G)
- Outline
  - Outline view (can be sorted a-z)
  - Outline popup (**Ctlr-o + regexp**)



```
package edu.coding.samples.wordcount.step3;

import java.io.EOFException;

 * This is a producer that parses an input stream into words, also tracking

class Producer {
  Executor m_executor;
  CircularBuffer m_buffer;
```



```
edu.coding.samples.wordcount.step4
Executor
  DelayedQueue
  m_ready : LinkedList<Runnable>
  m_delayed : DelayedQueue
  Delayed
  Executor()
  m_nevents : int
  sleep(long) : void
  run() : void
  execute(Runnable) : void
  execute(Runnable, long) : void
```

Eclipse – Java Debugging

- Eclipse has a fantastic debugger for Java
  - Use it – You will love it
  - Really well integrated withint the Eclipse environment
- What is a debugger for?
  - A debugger is a tool to watch the execution of a program
  - It allows you to **single step** through the execution
    - Single stepping means to execute one instruction/line at a time
  - It allows you to set **breakpoints** and run the program until it hits a breakpoint
    - A breakpoint is a point in your program where the execution will stop
  - It allows you to **introspect** your program
    - When the execution is suspended
    - The debugger shows you the call stack, local variables, etc.

Eclipse – Java Debugging

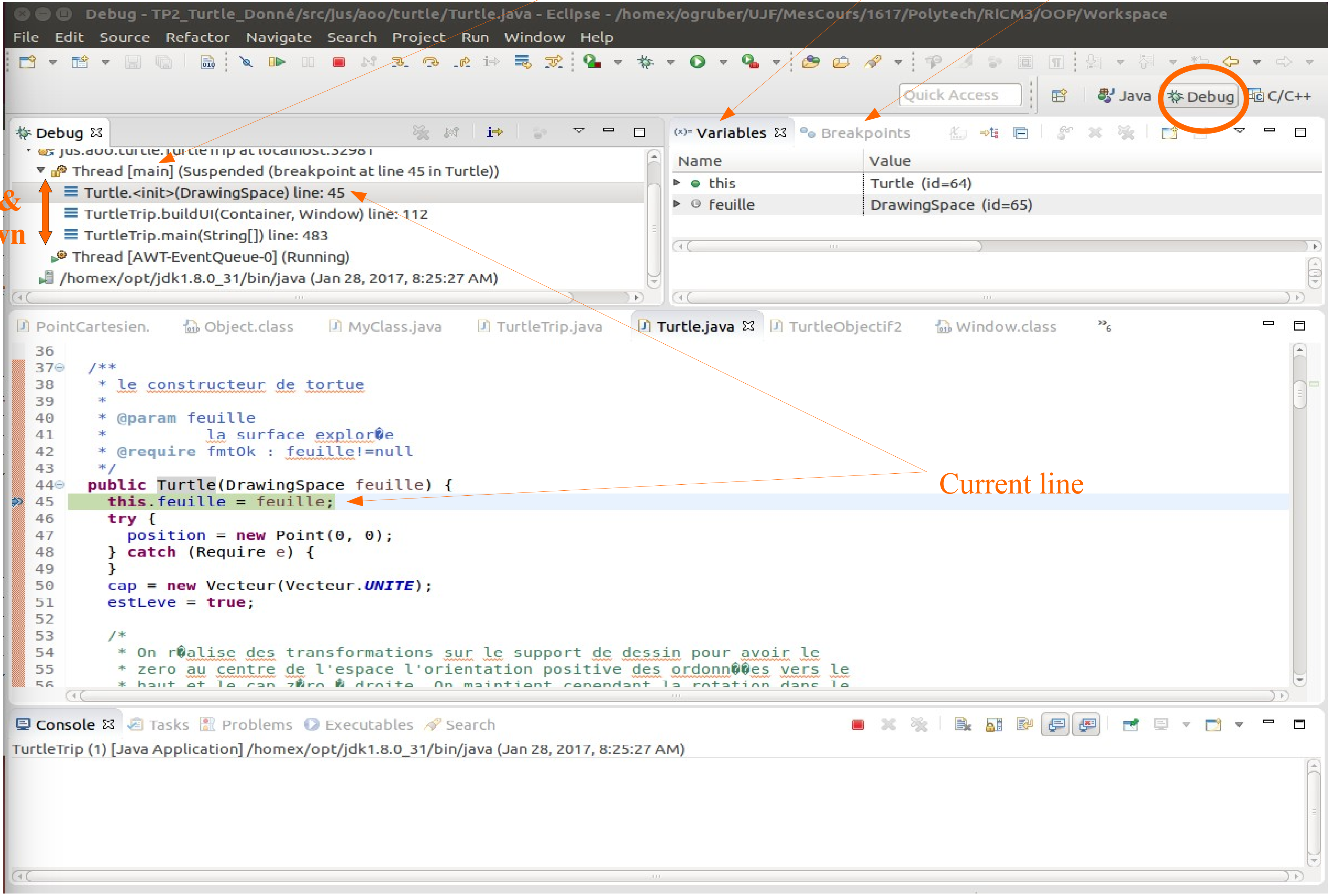
- Java perspective vs Debug perspective
  - Java perspective is to develop Java program
  - Debug perspective is to debug programs
- How to get started
  - Set a breakpoint on the first line of the main method
    - In the Package Explorer, scroll to your class and then to its main method
    - Double-click on the left edge of the editor at the first line of the main method
  - Then launch the execution
    - In the Package Explorer, scroll to your class and then to its main method
    - Right-click on the class with the static main method
    - Select Debug-As → Java Application ◀

The first time, Eclipse may ask you
If you want to switch to the Debug perspective
Click remember my decision and says yes.

**WHILE DEVELOPING,
ALWAYS EXECUTE
UNDER THE DEBUGGER!**

You never know when a bug will show up...

Eclipse – Java Debugging



Debug - TP2_Turtle_Donné/src/jus/aoo/turtle/Turtle.java - Eclipse - /homex/ogruber/UJF/MesCours/1617/Polytech/RICM3/OOP/Workspace

```
jus.aoo.turtle.TurtleTrip at localhost:32981
  Thread [main] (Suspended (breakpoint at line 45 in Turtle))
    Turtle.<init>(DrawingSpace) line: 45
    TurtleTrip.buildUI(Container, Window) line: 112
    TurtleTrip.main(String[]) line: 483
  Thread [AWT-EventQueue-0] (Running)
/homex/opt/jdk1.8.0_31/bin/java (Jan 28, 2017, 8:25:27 AM)
```

Name	Value
this	Turtle (id=64)
feuille	DrawingSpace (id=65)

```
36
37    /**
38     * le constructeur de tortue
39     *
40     * @param feuille
41     *            la surface explor�e
42     * @require fmtOk : feuille!=null
43     */
44    public Turtle(DrawingSpace feuille) {
45      this.feuille = feuille;
46      try {
47        position = new Point(0, 0);
48      } catch (Require e) {
49      }
50      cap = new Vecteur(Vecteur.UNITE);
51      estLeve = true;
52
53      /*
54       * On r�alise des transformations sur le support de dessin pour avoir le
55       * zero au centre de l'espace l'orientation positive des ordonn��es vers le
56       * haut et le cap z�ro � droite. On maintient cependant la rotation dans le
```

TurtleTrip (1) [Java Application] /homex/opt/jdk1.8.0_31/bin/java (Jan 28, 2017, 8:25:27 AM)

Eclipse – Java Debugging

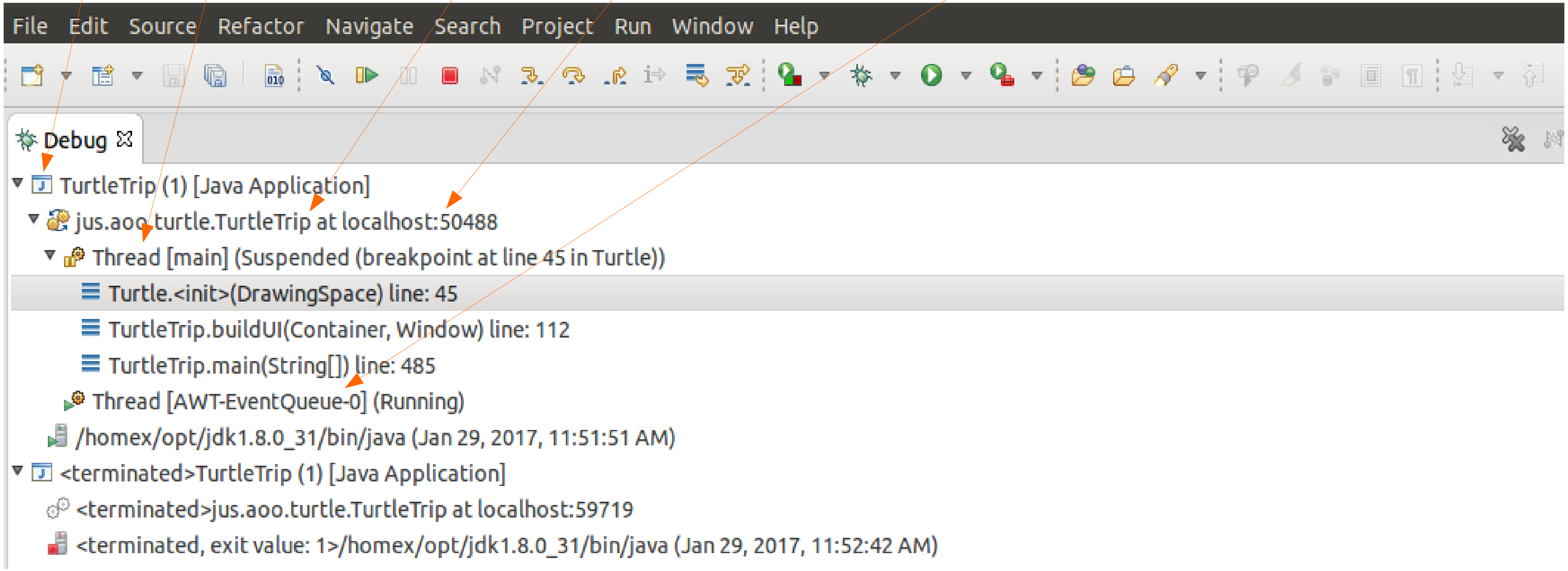
One Java Application under debug...

Launched class: jus.aoo.turtle.TurtleTrip

Main thread (flow of execution)

Local debug (localhost:50488)

GUI Thread (AWT)


File Edit Source Refactor Navigate Search Project Run Window Help
Debug

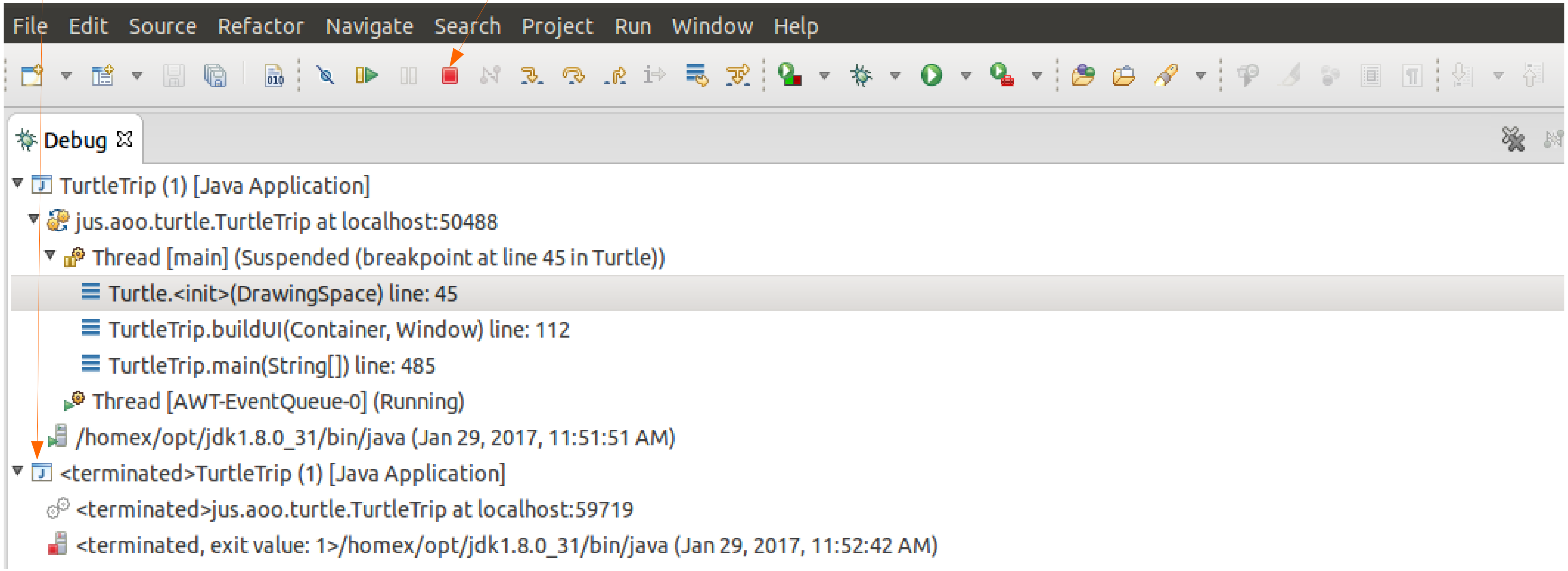
```
TurtleTrip (1) [Java Application]
  jus.aoo.turtle.TurtleTrip at localhost:50488
    Thread [main] (Suspended (breakpoint at line 45 in Turtle))
      Turtle.<init>(DrawingSpace) line: 45
      TurtleTrip.buildUI(Container, Window) line: 112
      TurtleTrip.main(String[]) line: 485
    Thread [AWT-EventQueue-0] (Running)
  /homex/opt/jdk1.8.0_31/bin/java (Jan 29, 2017, 11:51:51 AM)
<terminated>TurtleTrip (1) [Java Application]
  <terminated>jus.aoo.turtle.TurtleTrip at localhost:59719
  <terminated, exit value: 1>/homex/opt/jdk1.8.0_31/bin/java (Jan 29, 2017, 11:52:42 AM)
```

Eclipse – Java Debugging

Terminated debug session

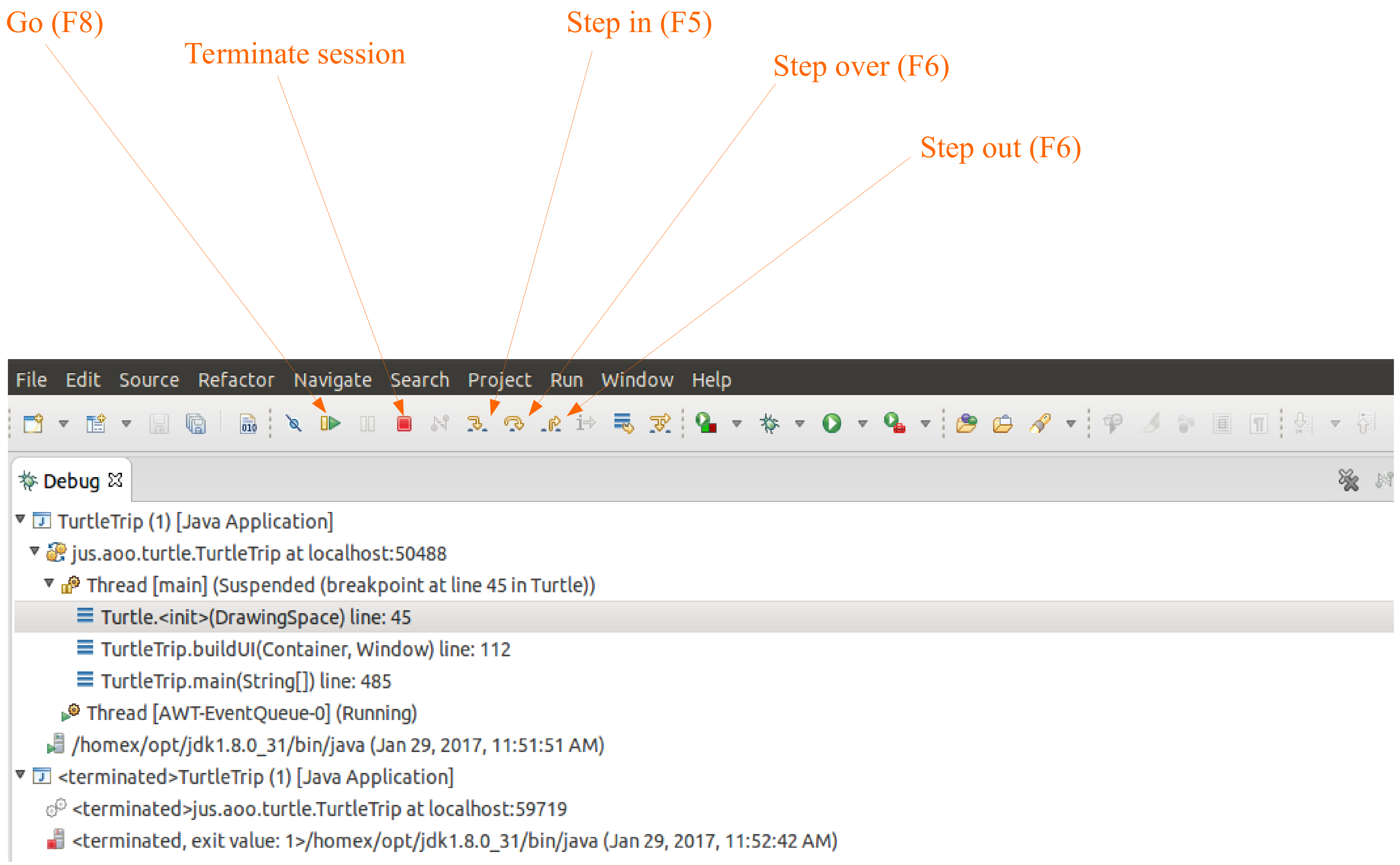
One can use multiple debug sessions at the same time in Eclipse

Make sure you terminate all sessions that you no longer follow



```
TurtleTrip (1) [Java Application]
  jus.aoo.turtle.TurtleTrip at localhost:50488
    Thread [main] (Suspended (breakpoint at line 45 in Turtle))
      Turtle.<init>(DrawingSpace) line: 45
      TurtleTrip.buildUI(Container, Window) line: 112
      TurtleTrip.main(String[]) line: 485
    Thread [AWT-EventQueue-0] (Running)
  /homex/opt/jdk1.8.0_31/bin/java (Jan 29, 2017, 11:51:51 AM)
<terminated>TurtleTrip (1) [Java Application]
  <terminated>jus.aoo.turtle.TurtleTrip at localhost:59719
  <terminated, exit value: 1>/homex/opt/jdk1.8.0_31/bin/java (Jan 29, 2017, 11:52:42 AM)
```

Eclipse – Java Stepping



```
TurtleTrip (1) [Java Application]
  jus.aoo.turtle.TurtleTrip at localhost:50488
    Thread [main] (Suspended (breakpoint at line 45 in Turtle))
      Turtle.<init>(DrawingSpace) line: 45
      TurtleTrip.buildUI(Container, Window) line: 112
      TurtleTrip.main(String[]) line: 485
    Thread [AWT-EventQueue-0] (Running)
  /homex/opt/jdk1.8.0_31/bin/java (Jan 29, 2017, 11:51:51 AM)
<terminated>TurtleTrip (1) [Java Application]
  <terminated>jus.aoo.turtle.TurtleTrip at localhost:59719
  <terminated, exit value: 1>/homex/opt/jdk1.8.0_31/bin/java (Jan 29, 2017, 11:52:42 AM)
```

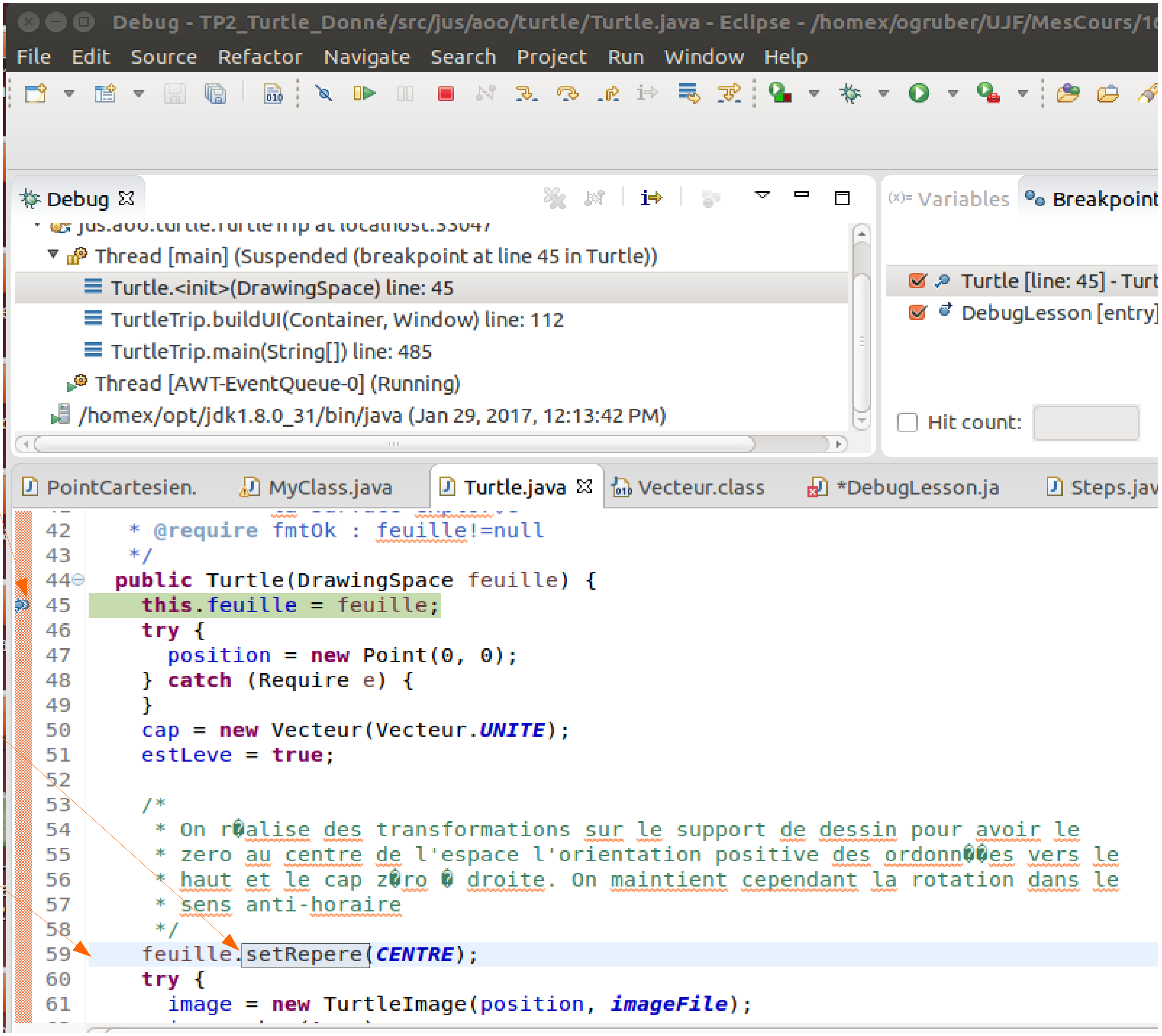
Eclipse – Java Stepping

Set breakpoints by double-clicks on the left edge of the window

Top tricks:

Run to selection (Ctlr-F5)

Run to line (Ctlr-R)

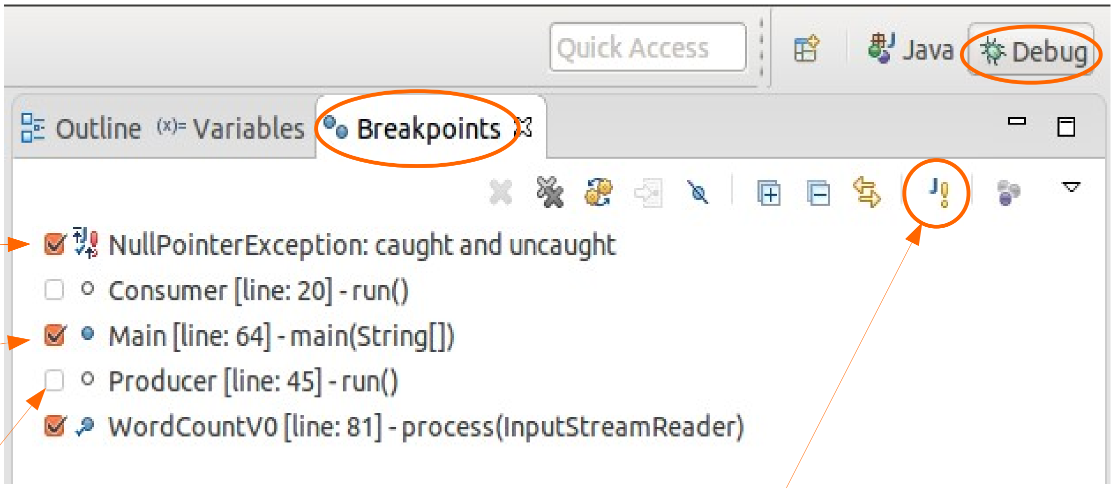

Debug - TP2_Turtle_Donné/src/jus/aoo/turtle/Turtle.java - Eclipse - /homex/ogruber/UJF/MesCours/1
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
Variables
Breakpoint
Turtle [line: 45] - Turt
DebugLesson [entry]
Hit count:
PointCartesien.
MyClass.java
Turtle.java
Vecteur.class
*DebugLesson.ja
Steps.jav

```
Thread [main] (Suspended (breakpoint at line 45 in Turtle))
    Turtle.<init>(DrawingSpace) line: 45
    TurtleTrip.buildUI(Container, Window) line: 112
    TurtleTrip.main(String[]) line: 485
Thread [AWT-EventQueue-0] (Running)
/homex/opt/jdk1.8.0_31/bin/java (Jan 29, 2017, 12:13:42 PM)
```

```java
   * @require fmtOk : feuille!=null
   */
  public Turtle(DrawingSpace feuille) {
    this.feuille = feuille;
    try {
      position = new Point(0, 0);
    } catch (Require e) {
    }
    cap = new Vecteur(Vecteur.UNITE);
    estLeve = true;

    /*
     * On r�alise des transformations sur le support de dessin pour avoir le
     * zero au centre de l'espace l'orientation positive des ordonn��es vers le
     * haut et le cap z�ro � droite. On maintient cependant la rotation dans le
     * sens anti-horaire
     */
    feuille.setRepere(CENTRE);
    try {
      image = new TurtleImage(position, imageFile);
```

Eclipse – Java Stepping and Breakpoints

- Single stepping summary
  - F5 (step into) Ctlr-F5 (step into selection)
  - F6 (step over)
  - F7 (step out)
  - F8 (go)
  - Ctlr-R (run to line)
- Breakpoints
  - On thrown exception
    - Caught or Uncaught
  - On a given line



- ☑ NullPointerException: caught and uncaught
- ☐ Consumer [line: 20] - run()
- ☑ Main [line: 64] - main(String[])
- ☐ Producer [line: 45] - run()
- ☑ WordCountV0 [line: 81] - process(InputStreamReader)

Active/Inactive breakpoints

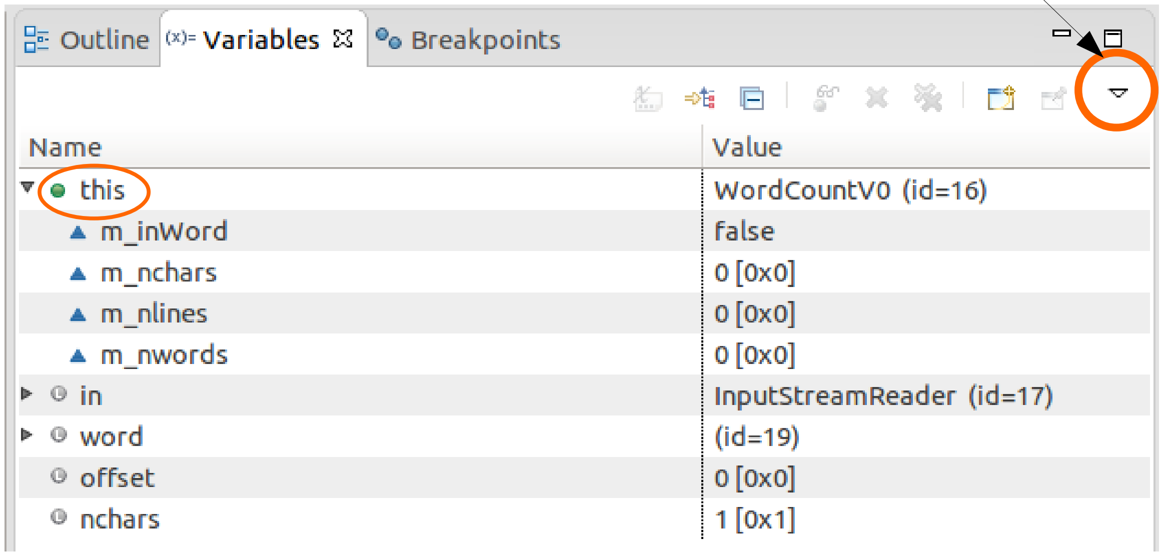
To add a breakpoint
for an exception

Note: to change your key bindings
Alt-w → Preferences
General → Editor → Keys

Eclipse – Java Debugging

- Introspecting variables via the Variables View
  - Receiver (this)
  - Method arguments
  - Local variables
- Values
  - In decimal/hexadecimal
  - Walk through objects
  - Calling Object.toString()

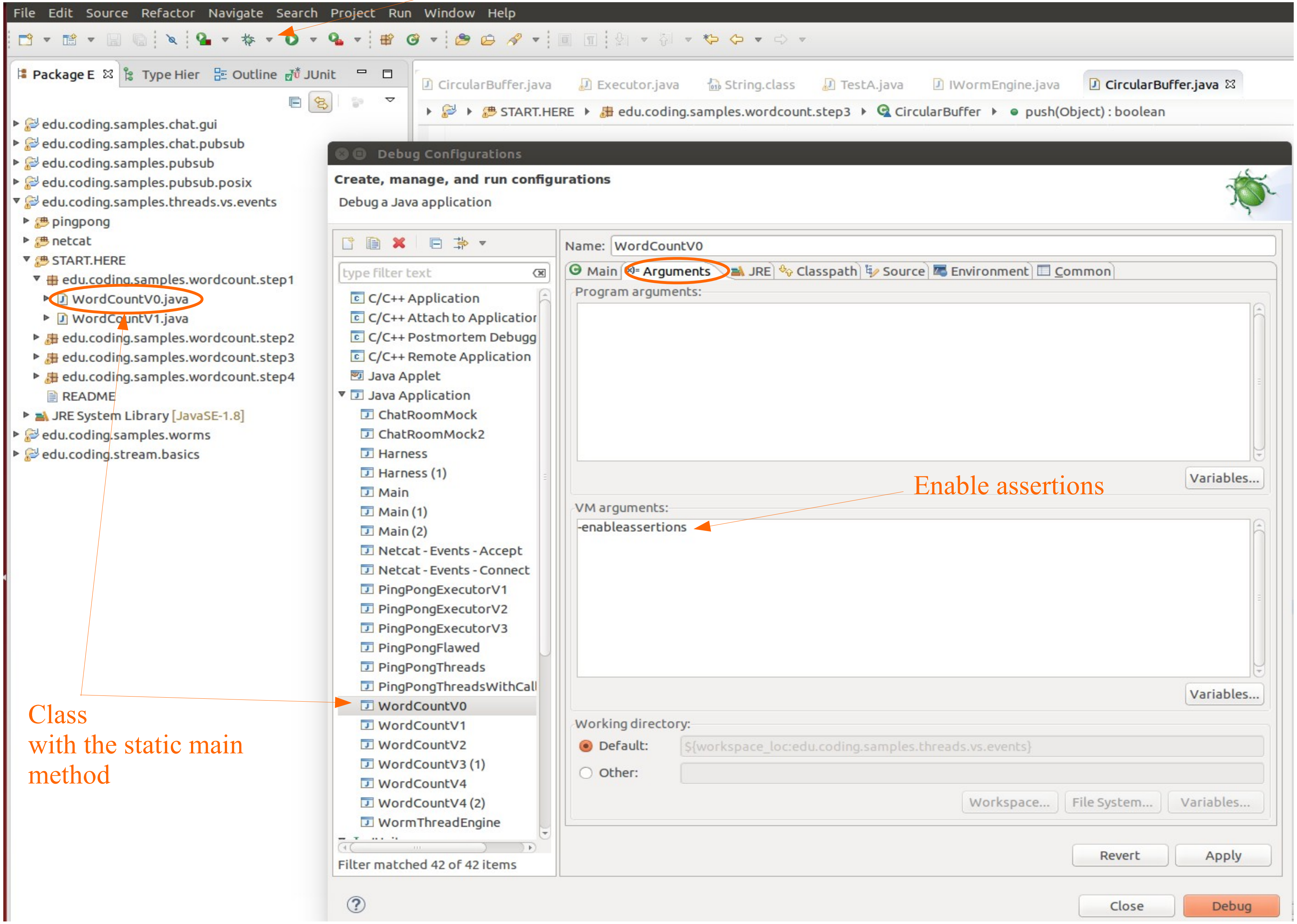
Note: there is a bug in the variable tab sometimes
You are left with only one column
Drop-down menu → layout → select columns
Select a new column and all will come back



Name	Value
this	WordCountV0 (id=16)
m_inWord	false
m_nchars	0 [0x0]
m_nlines	0 [0x0]
m_nwords	0 [0x0]
in	InputStreamReader (id=17)
word	(id=19)
offset	0 [0x0]
nchars	1 [0x1]

Debug Configurations

Drop-down combo → Debug configurations



Name: WordCountV0

Program arguments:

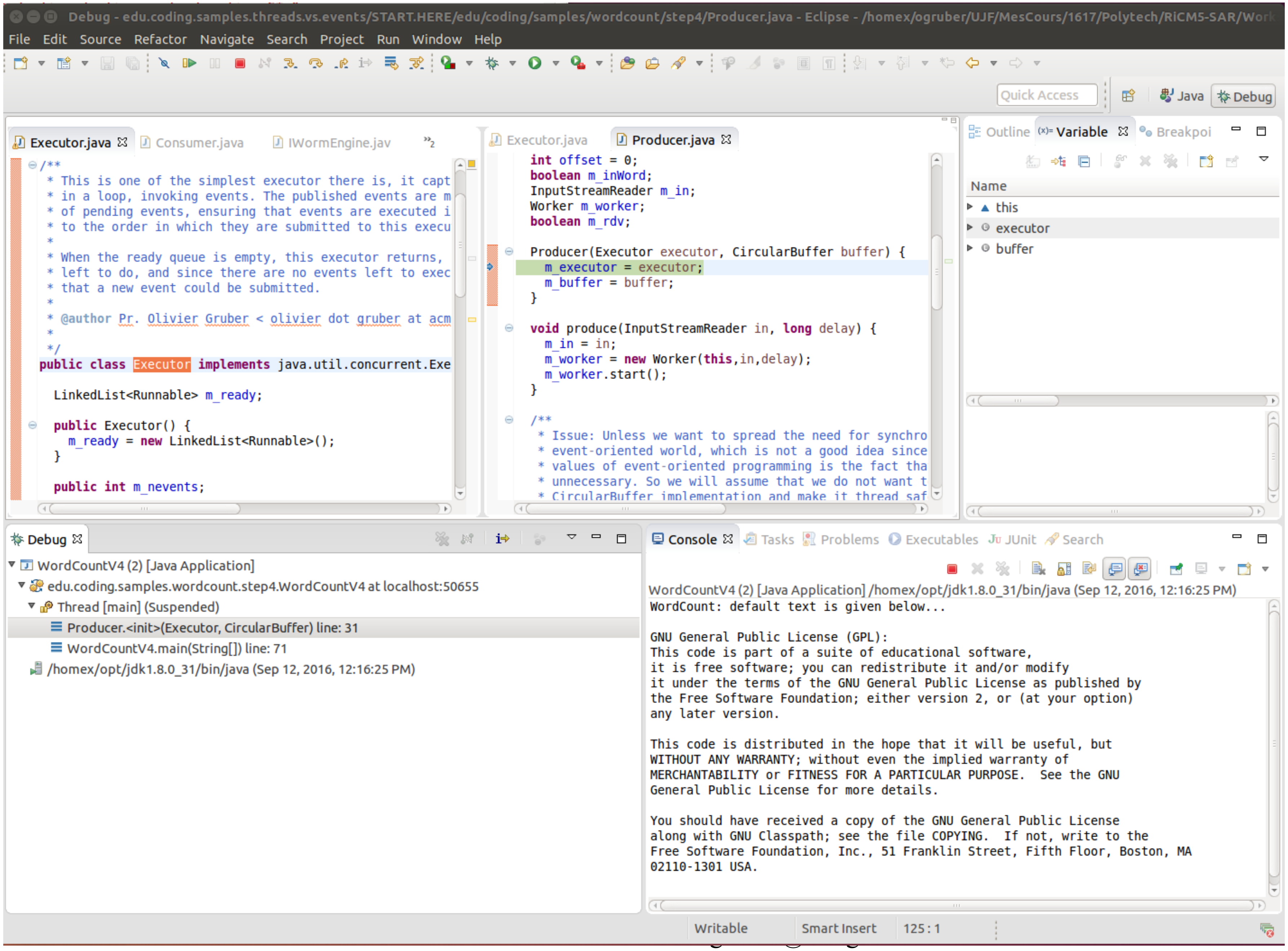
VM arguments: -enableassertions

Working directory:
[✓ Default:] ${workspace_loc:edu.coding.samples.threads.vs.events}
Other:

Enable assertions

Class
with the static main
method

Eclipse – Java Debugging – Use your own layout!



Debug - edu.coding.samples.threads.vs.events/START.HERE/edu/coding/samples/wordcount/step4/Producer.java - Eclipse - /homex/ogruber/UJF/MesCours/1617/Polytech/RICM5-SAR/Wor

```
/**
 * This is one of the simplest executor there is, it capt
 * in a loop, invoking events. The published events are m
 * of pending events, ensuring that events are executed i
 * to the order in which they are submitted to this execu
 *
 * When the ready queue is empty, this executor returns,
 * left to do, and since there are no events left to exec
 * that a new event could be submitted.
 *
 * @author Pr. Olivier Gruber < olivier dot gruber at acm
 *
 */
public class Executor implements java.util.concurrent.Exe

  LinkedList<Runnable> m_ready;

  public Executor() {
    m_ready = new LinkedList<Runnable>();
  }

  public int m_nevents;
```

```
int offset = 0;
boolean m_inWord;
InputStreamReader m_in;
Worker m_worker;
boolean m_rdv;

Producer(Executor executor, CircularBuffer buffer) {
  m_executor = executor;
  m_buffer = buffer;
}

void produce(InputStreamReader in, long delay) {
  m_in = in;
  m_worker = new Worker(this,in,delay);
  m_worker.start();
}

/**
 * Issue: Unless we want to spread the need for synchro
 * event-oriented world, which is not a good idea since
 * values of event-oriented programming is the fact tha
 * unnecessary. So we will assume that we do not want t
 * CircularBuffer implementation and make it thread saf
```

Name
this
executor
buffer

```
WordCountV4 (2) [Java Application]
  edu.coding.samples.wordcount.step4.WordCountV4 at localhost:50655
    Thread [main] (Suspended)
      Producer.<init>(Executor, CircularBuffer) line: 31
      WordCountV4.main(String[]) line: 71
  /homex/opt/jdk1.8.0_31/bin/java (Sep 12, 2016, 12:16:25 PM)
```

```
WordCountV4 (2) [Java Application] /homex/opt/jdk1.8.0_31/bin/java (Sep 12, 2016, 12:16:25 PM)
WordCount: default text is given below...

GNU General Public License (GPL):
This code is part of a suite of educational software,
it is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2, or (at your option)
any later version.

This code is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
General Public License for more details.

You should have received a copy of the GNU General Public License
along with GNU Classpath; see the file COPYING.  If not, write to the
Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
02110-1301 USA.
```

Eclipse – Java Unit Testing

- JUnit has direct support in Eclipse
  - See JUnit/Eclipse documentation
  - See given examples
- Core points
  - Add JUnit (project properties → Java Build Path → Add library)
  - You can run/debug as JUnit a Java package or a Java source folder (right-click → debug as)

WHILE JUNIT TESTING, YOU CAN EXECUTE UNDER THE DEBUGGER ALSO!

Trust me, you will have to debug both your tests and your application...

http://junit.sourceforge.net/doc/cookbook/cookbook.htm
http://help.eclipse.org/mars/index.jsp?topic=%2Forg.eclipse.jdt.doc.user%2FgettingStarted%2Fqs-junit.htm

Eclipse – Java Unit Testing with Coverage

- Cobertura
  - Runs your JUnit tests gathering coverage information
  - See the edu.coding.testing project
  - See the cobertura.sh scripts
- Coverage is important
  - You will be surprised at how even complex tests cover so few lines...
  - Most of the times though, 100% is not achievable

Even at 100%, you probably haven't tested all situations

Testing is just hard and necessary.

Regression testing is very important for evolving software.

Eclipse – Java Performance Tuning

- Golden rule
  - Performance numbers are just numbers, the interpretation is everything
- Java Visual VM (jvisualvm)
  - It is your health monitor for your Java Runtime Environments (JREs)
  - It is part of the Oracle JDK (in the bin directory, with java)
- Why bother with Java Visual VM?
  - Machines are extremely fast → Hide performance problems
- JVisualVM – main features
  - CPU usage, showing garbage collection overhead
  - Heap evolution, showing memory usage
  - Showing threads, loaded classes

Eclipse – Java Performance Tuning

- Golden rule
  - Performance numbers are just numbers, the interpretation is everything
- Java Visual VM (jvisualvm)
  - It is your health monitor for your Java Runtime Environments (JREs)
  - It is part of the Oracle JDK (in the bin directory, with java)
- Small Java test harness
  - See edu.coding.testing project → edu.coding.perfs package
  - A simple harness that supports running multiple benchmarks
    - With warmup or not
    - Computes averaged elapsed time over several execution (default is 10)
    - With forced garbage collection in between runs (or not)
    - With basic support for hprof
  - Yourkit (https://www.yourkit.com/)
    - Seems a pretty complete and advanced profiling framework for Java and DotNet

Conclusion

Hope this talk helped you.

Happy coding...

Eclipse – Setup

- Windows/Mac users...
  - You are on your own with Google
- Linux users
  - My setup is like this
  - Because of GTK issues, on Ubuntu 14.04, with Eclipse Mars

```
$ more ~/bin/eclipse
#!/bin/sh
export JAVA_HOME=/homex/opt/jdk1.8.0_31/
PATH=$PATH:$JAVA_HOME/bin
export SWT_GTK3=0
export UBUNTU_MENUPROXY=0
/homex/opt/eclipse-mars/eclipse -showlocation $*
```

Eclipse – Setup

- Eclipse.ini

```
$ more /homex/opt/eclipse-mars/eclipse.ini
-startup
plugins/org.eclipse.equinox.launcher_1.3.100.v20150511-1540.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.gtk.linux.x86_64_1.1.300.v20150602-1417
-product
org.eclipse.epp.package.java.product
--launcher.defaultAction
openFile
-showsplash
org.eclipse.platform
--launcher.XXMaxPermSize
256m
--launcher.defaultAction
openFile
--launcher.appendVmargs
-vmargs
-Dosgi.requiredJavaVersion=1.7
-XX:MaxPermSize=256m
-Xms256m
-Xmx1024m
```

Eclipse Workbench

- Each Perspective

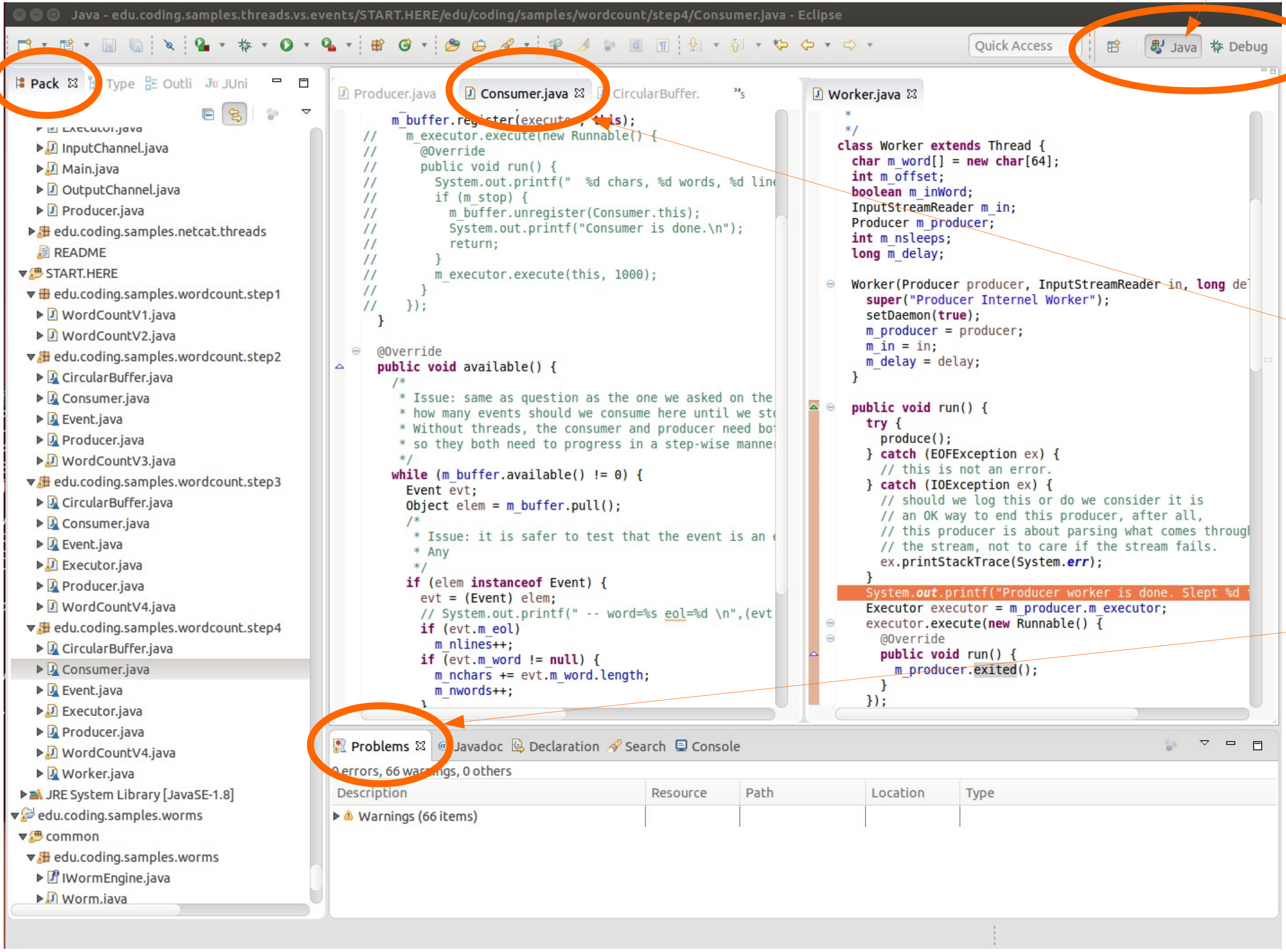
Current perspective (Java)

Package Explorer

Perspective switcher

An "Editor"

A view "Problems"



```
Executor.java
InputChannel.java
Main.java
OutputChannel.java
Producer.java
edu.coding.samples.netcat.threads
README
START.HERE
  edu.coding.samples.wordcount.step1
    WordCountV1.java
    WordCountV2.java
  edu.coding.samples.wordcount.step2
    CircularBuffer.java
    Consumer.java
    Event.java
    Producer.java
    WordCountV3.java
  edu.coding.samples.wordcount.step3
    CircularBuffer.java
    Consumer.java
    Event.java
    Executor.java
    Producer.java
    WordCountV4.java
  edu.coding.samples.wordcount.step4
    CircularBuffer.java
    Consumer.java
    Event.java
    Executor.java
    Producer.java
    WordCountV4.java
    Worker.java
JRE System Library [JavaSE-1.8]
edu.coding.samples.worms
  common
    edu.coding.samples.worms
      IWormEngine.java
      Worm.java
```

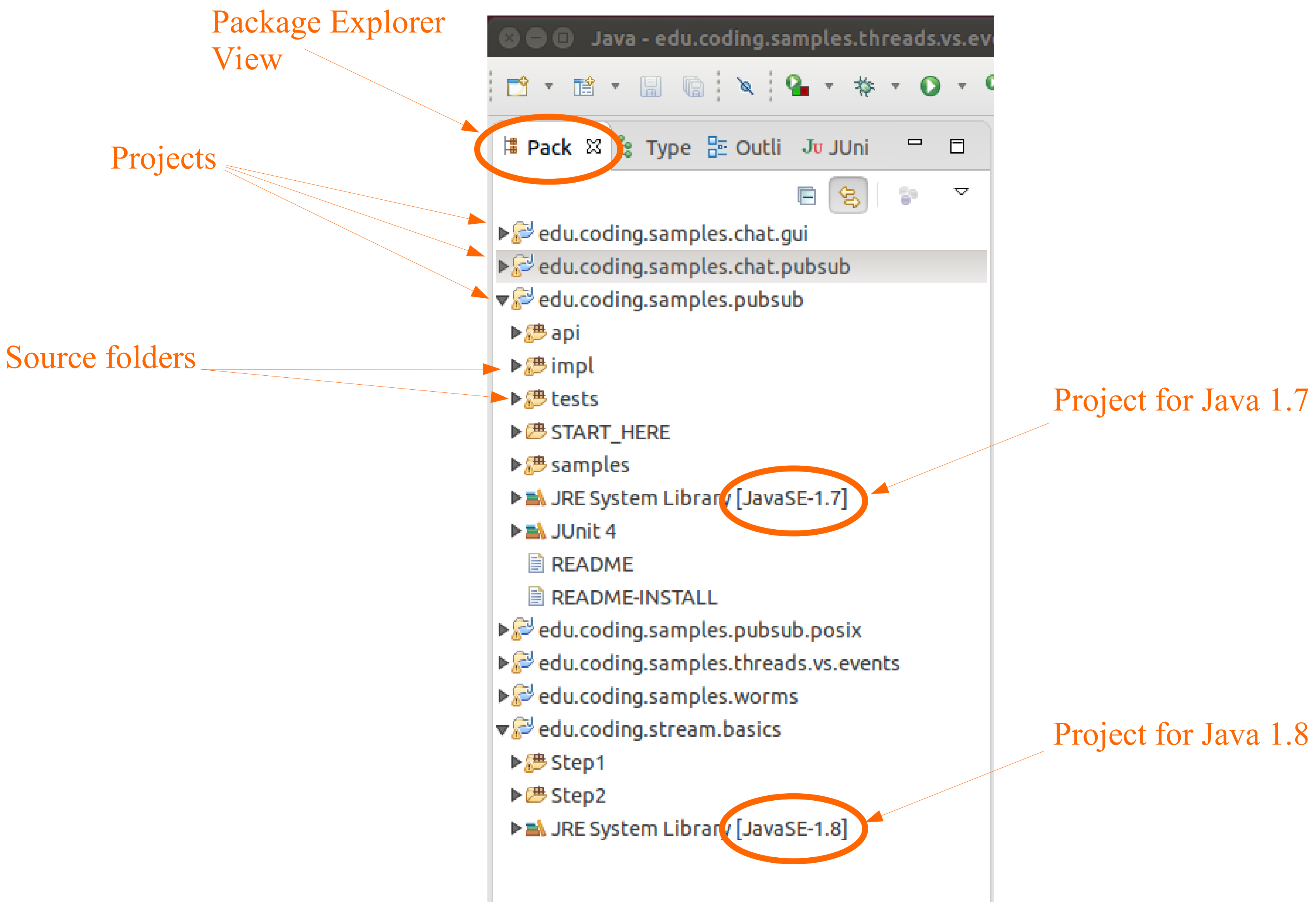
```java
    m_buffer.register(executor, this);
//    m_executor.execute(new Runnable() {
//      @Override
//      public void run() {
//        System.out.printf("  %d chars, %d words, %d line
//        if (m_stop) {
//          m_buffer.unregister(Consumer.this);
//          System.out.printf("Consumer is done.\n");
//          return;
//        }
//        m_executor.execute(this, 1000);
//      }
//    });
  }

  @Override
  public void available() {
    /*
     * Issue: same as question as the one we asked on the
     * how many events should we consume here until we sto
     * Without threads, the consumer and producer need bot
     * so they both need to progress in a step-wise manne
     */
    while (m_buffer.available() != 0) {
      Event evt;
      Object elem = m_buffer.pull();
      /*
       * Issue: it is safer to test that the event is an e
       * Any
       */
      if (elem instanceof Event) {
        evt = (Event) elem;
        // System.out.printf(" -- word=%s eol=%d \n",(evt
        if (evt.m_eol)
          m_nlines++;
        if (evt.m_word != null) {
          m_nchars += evt.m_word.length;
          m_nwords++;
        }
```

```java
 *
 */
class Worker extends Thread {
  char m_word[] = new char[64];
  int m_offset;
  boolean m_inWord;
  InputStreamReader m_in;
  Producer m_producer;
  int m_nsleeps;
  long m_delay;

  Worker(Producer producer, InputStreamReader in, long de
    super("Producer Internel Worker");
    setDaemon(true);
    m_producer = producer;
    m_in = in;
    m_delay = delay;
  }

  public void run() {
    try {
      produce();
    } catch (EOFException ex) {
      // this is not an error.
    } catch (IOException ex) {
      // should we log this or do we consider it is
      // an OK way to end this producer, after all,
      // this producer is about parsing what comes throug
      // the stream, not to care if the stream fails.
      ex.printStackTrace(System.err);
    }
    System.out.printf("Producer worker is done. Slept %d 
    Executor executor = m_producer.m_executor;
    executor.execute(new Runnable() {
      @Override
      public void run() {
        m_producer.exited();
      }
    });
```

0 errors, 66 warnings, 0 others

Description	Resource	Path	Location	Type
Warnings (66 items)				

Eclipse Workspace



edu.coding.samples.chat.gui
edu.coding.samples.chat.pubsub
edu.coding.samples.pubsub
api
impl
tests
START_HERE
samples
JRE System Library [JavaSE-1.7]
JUnit 4
README
README-INSTALL
edu.coding.samples.pubsub.posix
edu.coding.samples.threads.vs.events
edu.coding.samples.worms
edu.coding.stream.basics
Step1
Step2
JRE System Library [JavaSE-1.8]

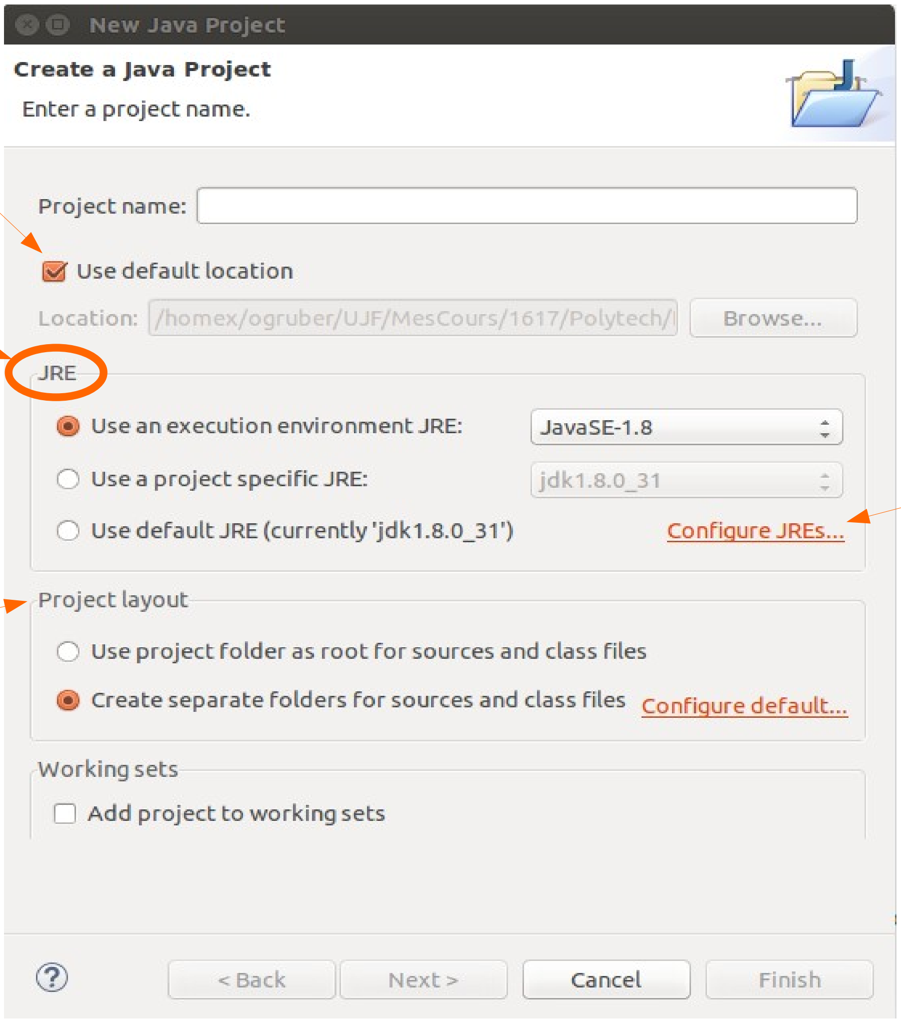
Eclipse Workspace – New Java Project

Default location in the Worskpace directory

Choose a JRE

Choose a layout

You might have to configure the JRE you want



New Java Project

Create a Java Project

Enter a project name.

Project name:

[✓] Use default location

Location: /homex/ogruber/UJF/MesCours/1617/Polytech/

JRE

(●) Use an execution environment JRE: JavaSE-1.8

( ) Use a project specific JRE: jdk1.8.0_31

( ) Use default JRE (currently 'jdk1.8.0_31')

Project layout

( ) Use project folder as root for sources and class files

(●) Create separate folders for sources and class files

Working sets

[ ] Add project to working sets

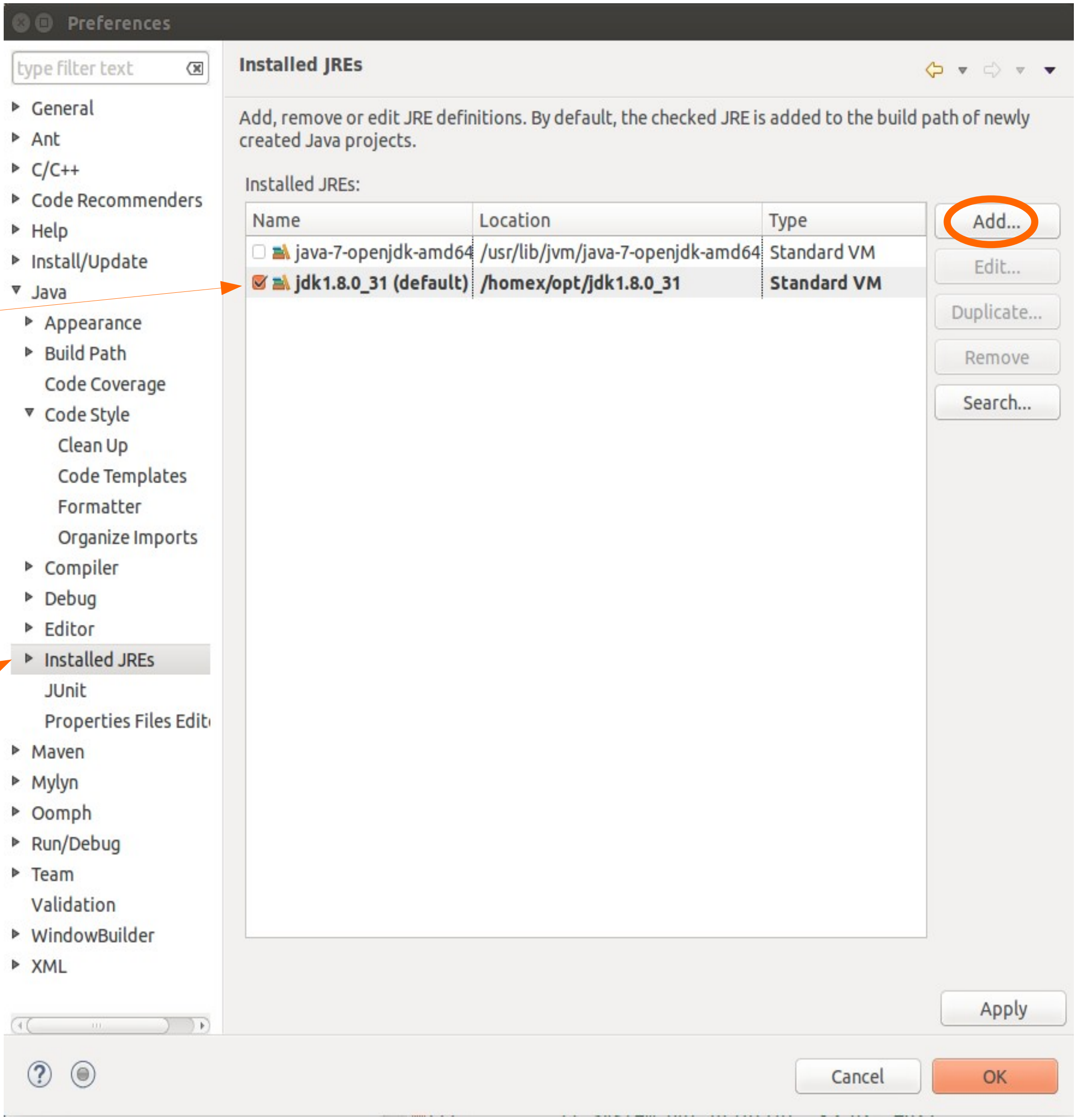
Eclipse Workspace – Java Runtime Environment Setup

- Configuring JREs
  - Installed JREs and JDKs
  - **Always install a JDK**
  - **From Oracle website**
  - Make it the default

To get this dialog, use menus

Window → Preferences

Drill down to
Java
Installed JREs



Installed JREs

Add, remove or edit JRE definitions. By default, the checked JRE is added to the build path of newly created Java projects.

Installed JREs:

	Name	Location	Type
☐	java-7-openjdk-amd64	/usr/lib/jvm/java-7-openjdk-amd64	Standard VM
☑	**jdk1.8.0_31 (default)**	**/homex/opt/jdk1.8.0_31**	**Standard VM**

Eclipse Workspace – Java Project Setup

Select the Java Build Path

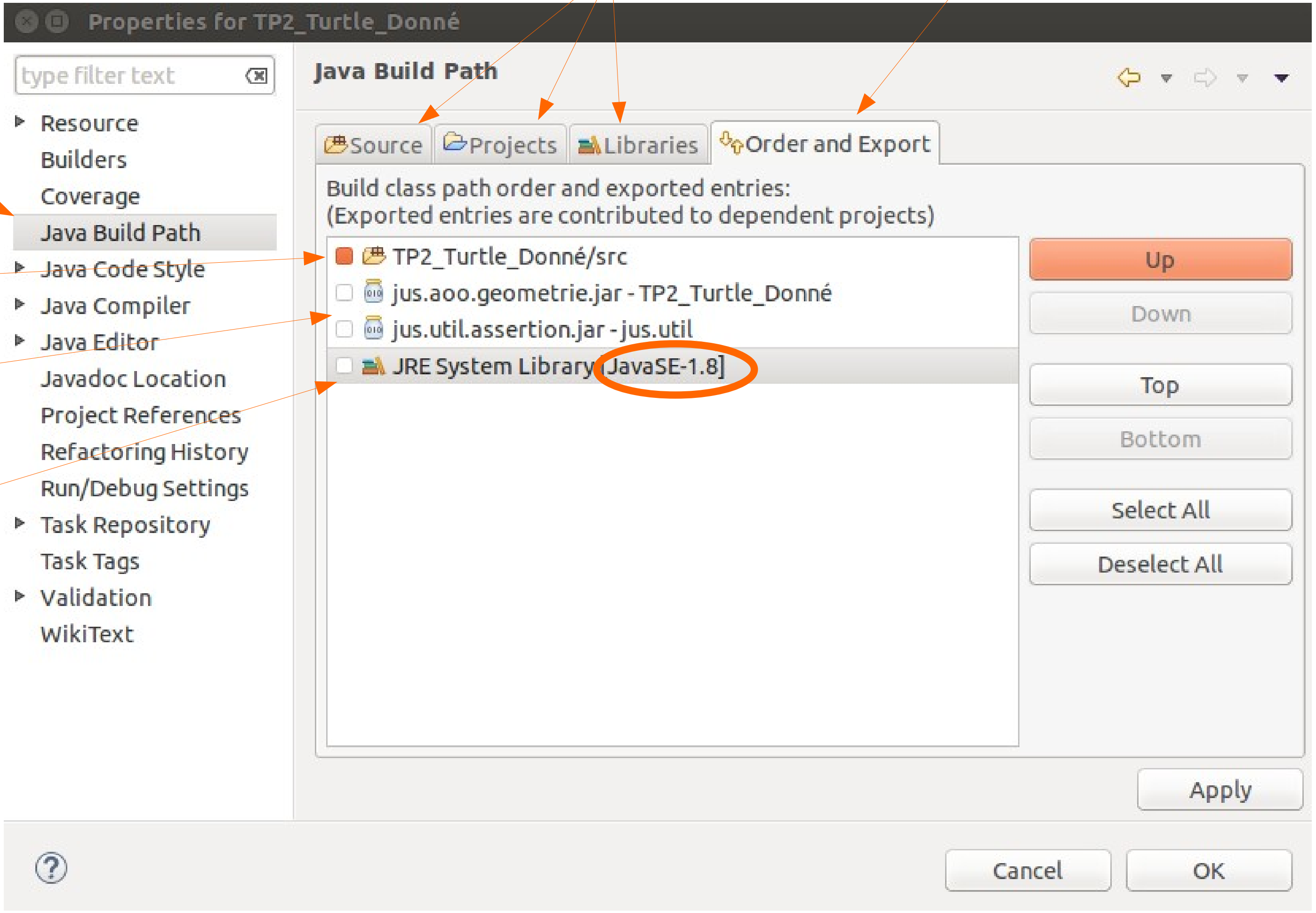
Sources

Libraries

Compatible
JRE System Library

Edit tabs

Summary



Properties for TP2_Turtle_Donné

Build class path order and exported entries:
(Exported entries are contributed to dependent projects)

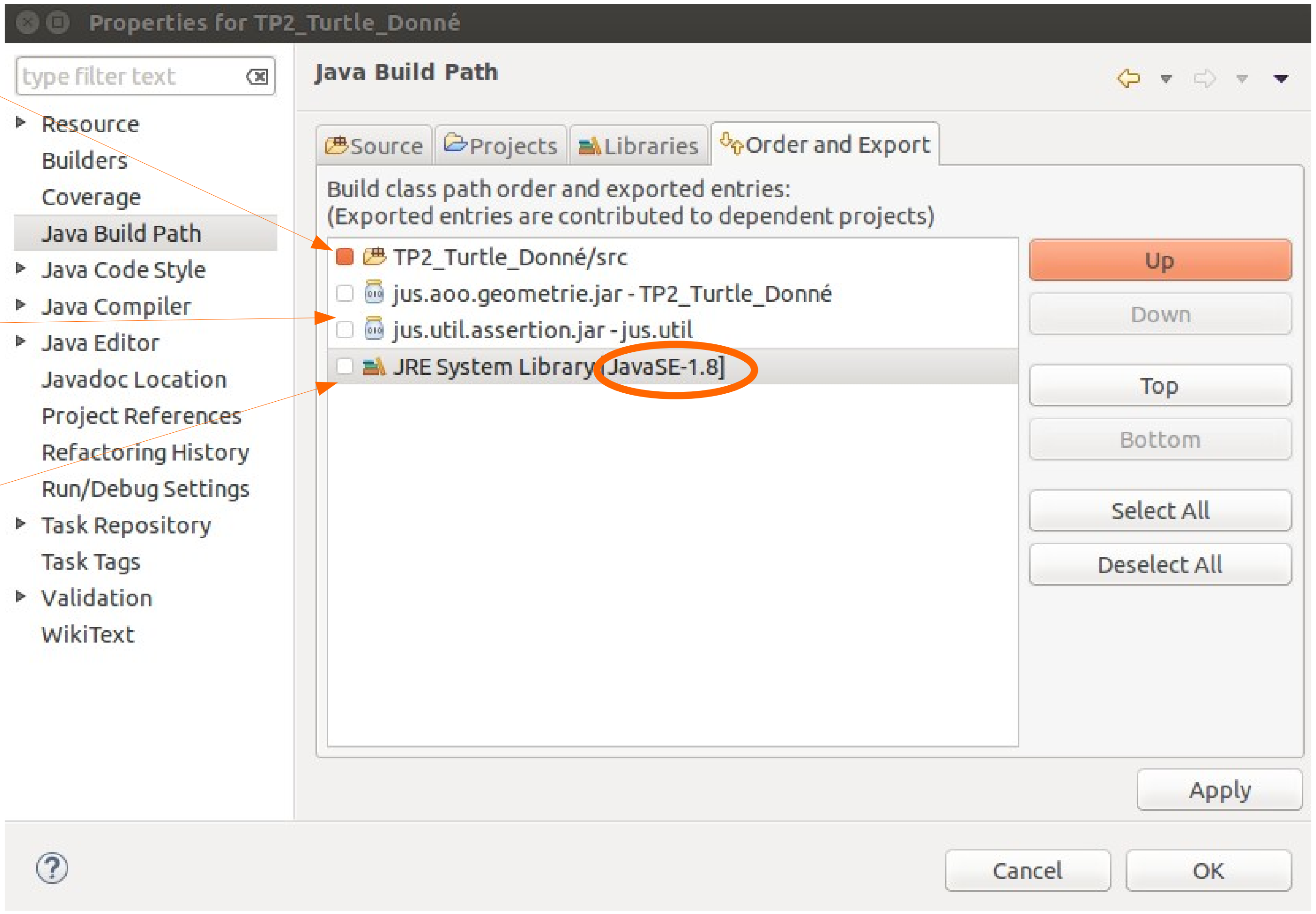
- TP2_Turtle_Donné/src
- jus.aoo.geometrie.jar - TP2_Turtle_Donné
- jus.util.assertion.jar - jus.util
- JRE System Library [JavaSE-1.8]

Eclipse Workspace – Java Project Setup

Exported source folder

Unexported libraries

Unexported
JRE System Library


Properties for TP2_Turtle_Donné

Java Build Path

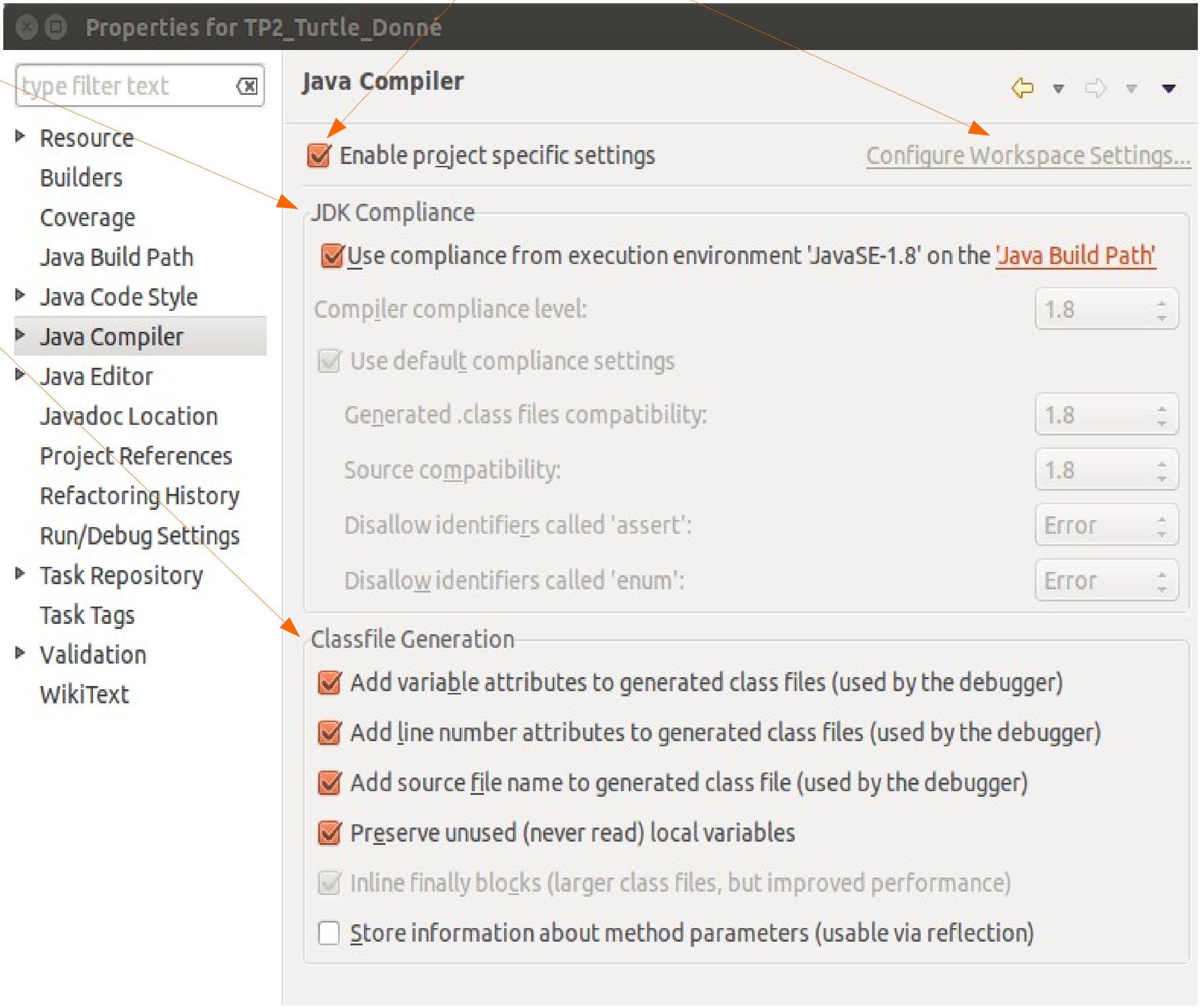
Build class path order and exported entries:
(Exported entries are contributed to dependent projects)

- [✓] TP2_Turtle_Donné/src
- [ ] jus.aoo.geometrie.jar - TP2_Turtle_Donné
- [ ] jus.util.assertion.jar - jus.util
- [ ] JRE System Library [JavaSE-1.8]

Eclipse Workspace – Java Project Setup

- Setup the Java Compiler
  - Choose JDK compliance
  - Select debug infos

Two levels of settings



Java Compiler

☑ Enable project specific settings

JDK Compliance

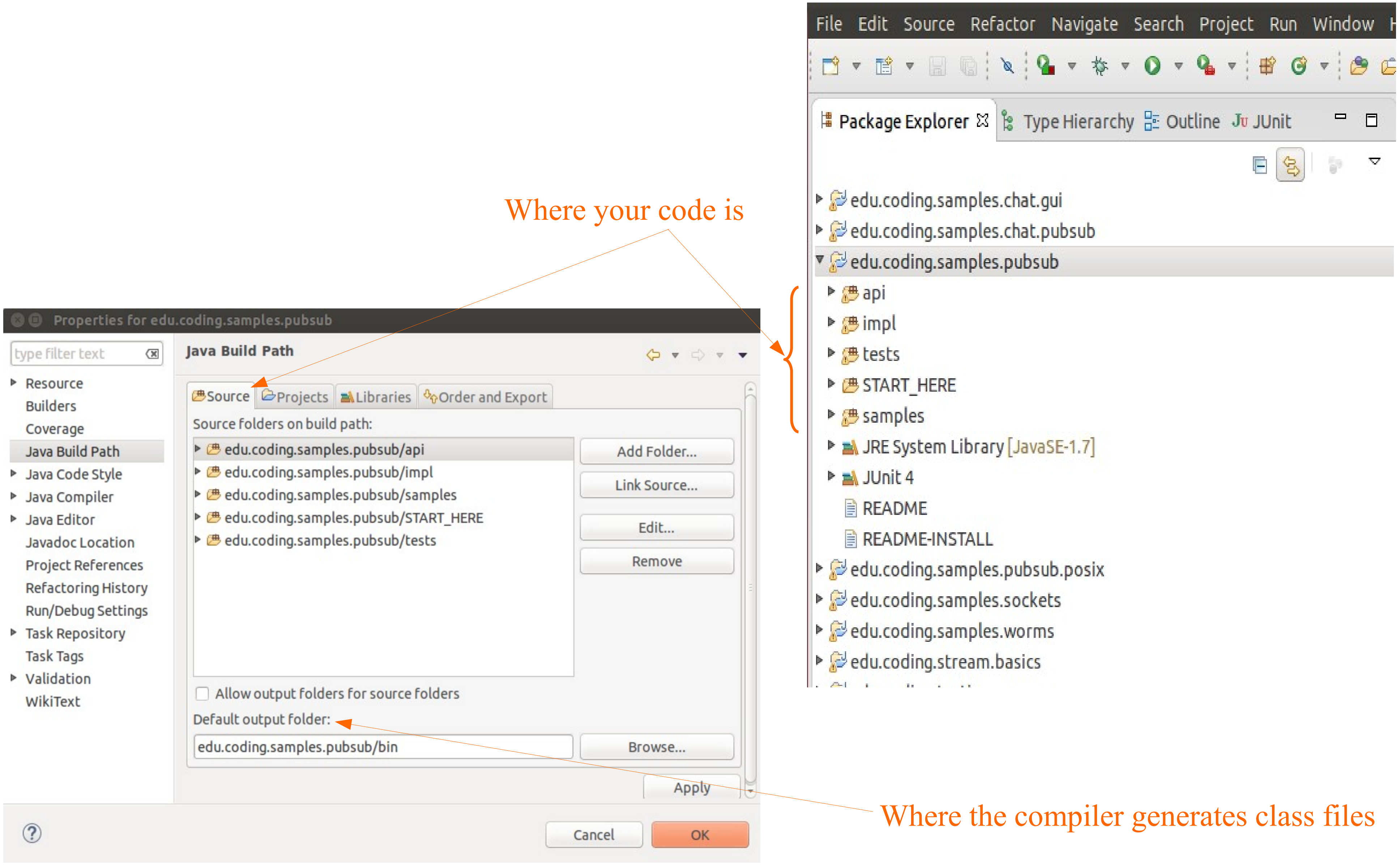
☑ Use compliance from execution environment 'JavaSE-1.8' on the 'Java Build Path'

Compiler compliance level:	1.8
☑ Use default compliance settings	
Generated .class files compatibility:	1.8
Source compatibility:	1.8
Disallow identifiers called 'assert':	Error
Disallow identifiers called 'enum':	Error

Classfile Generation

☑ Add variable attributes to generated class files (used by the debugger)
☑ Add line number attributes to generated class files (used by the debugger)
☑ Add source file name to generated class file (used by the debugger)
☑ Preserve unused (never read) local variables
☑ Inline finally blocks (larger class files, but improved performance)
☐ Store information about method parameters (usable via reflection)

Eclipse Workspace – Java Project Setup



Properties for edu.coding.samples.pubsub

Java Build Path

Source | Projects | Libraries | Order and Export

Source folders on build path:

- edu.coding.samples.pubsub/api
- edu.coding.samples.pubsub/impl
- edu.coding.samples.pubsub/samples
- edu.coding.samples.pubsub/START_HERE
- edu.coding.samples.pubsub/tests

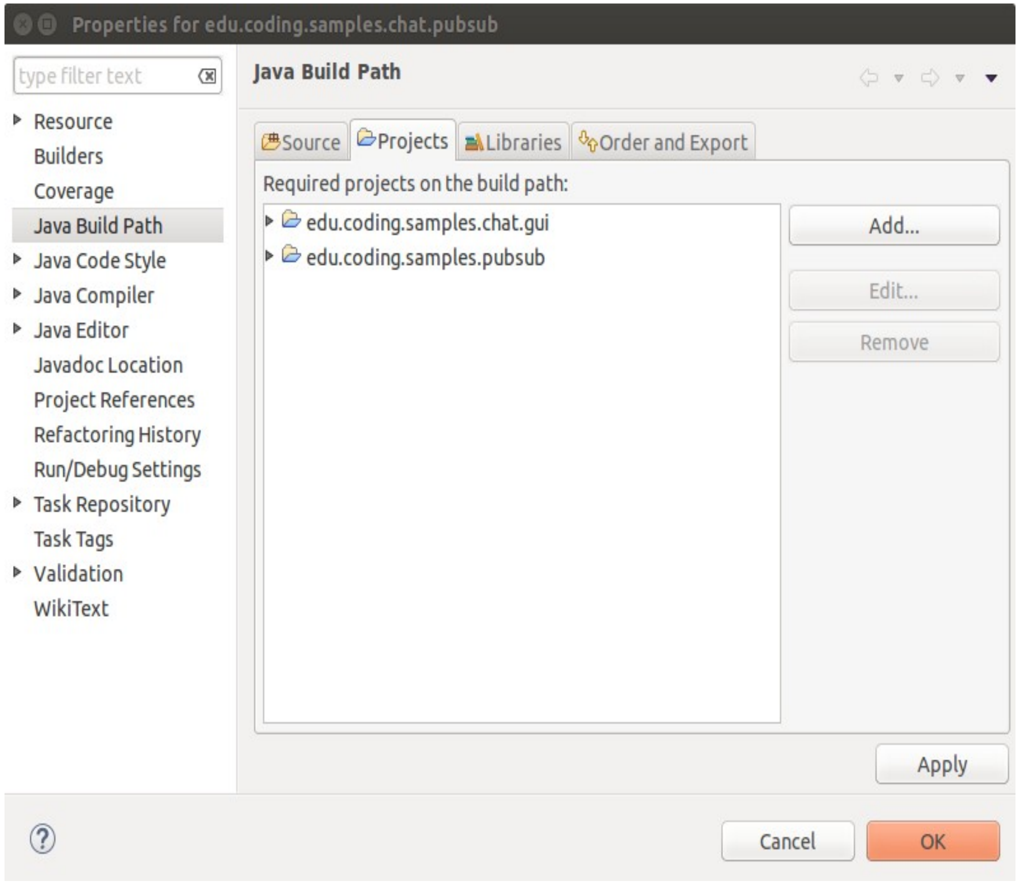
Allow output folders for source folders

Default output folder: edu.coding.samples.pubsub/bin

Package Explorer

- edu.coding.samples.chat.gui
- edu.coding.samples.chat.pubsub
- edu.coding.samples.pubsub
  - api
  - impl
  - tests
  - START_HERE
  - samples
  - JRE System Library [JavaSE-1.7]
  - JUnit 4
  - README
  - README-INSTALL
- edu.coding.samples.pubsub.posix
- edu.coding.samples.sockets
- edu.coding.samples.worms
- edu.coding.stream.basics

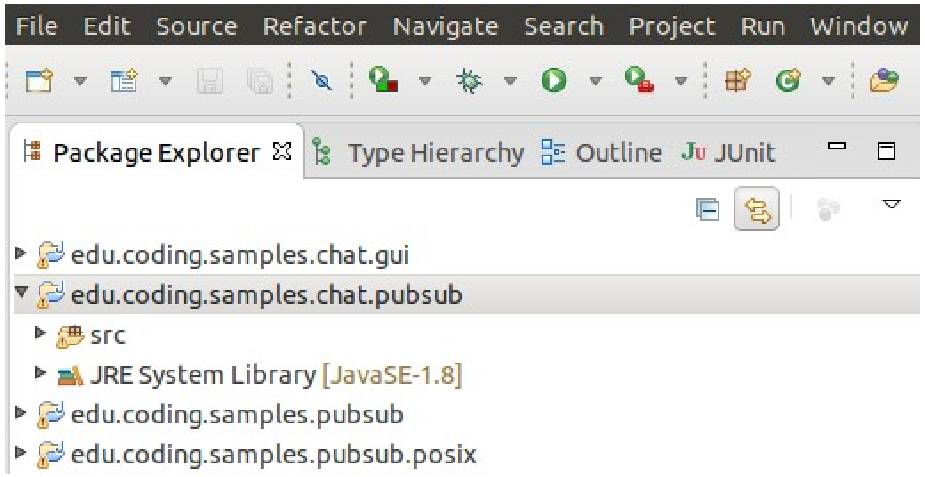
Eclipse Workspace – Java Project Setup


Properties for edu.coding.samples.chat.pubsub

Java Build Path

Source | Projects | Libraries | Order and Export

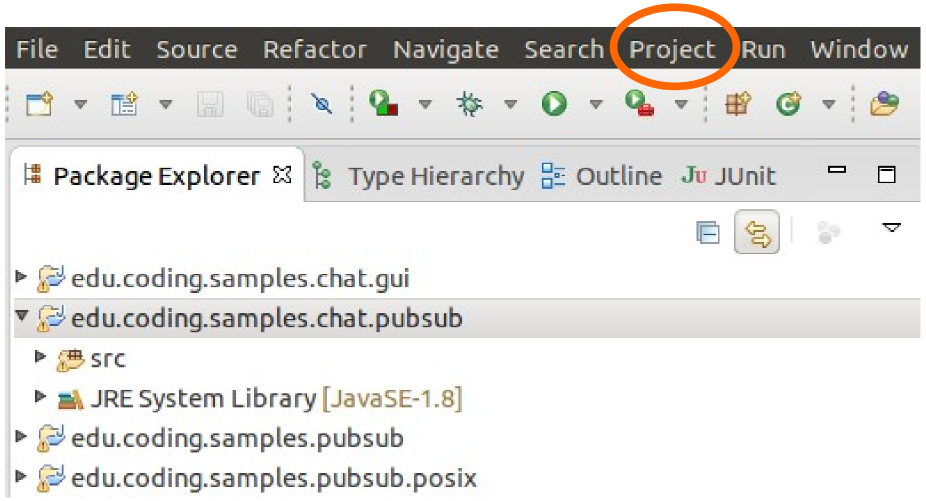
Required projects on the build path:
- edu.coding.samples.chat.gui
- edu.coding.samples.pubsub

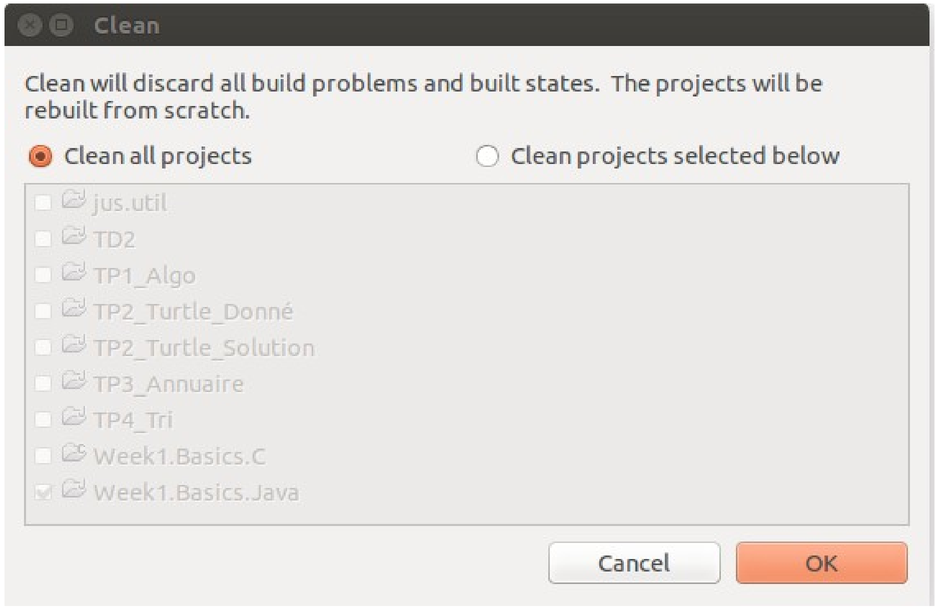

File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
Type Hierarchy
Outline
JUnit

- edu.coding.samples.chat.gui
- edu.coding.samples.chat.pubsub
  - src
  - JRE System Library [JavaSE-1.8]
- edu.coding.samples.pubsub
- edu.coding.samples.pubsub.posix

Eclipse Workspace – Java Project Build

- Project Menu
  - Automatic build is the default
  - Very accurate/efficient in Java
- Project Clean
  - Sometimes cleaning is necessary
  - Alt-P → Clean


File
Edit
Source
Refactor
Navigate
Search
Project
Run
Window
Package Explorer
Type Hierarchy
Outline
JUnit
edu.coding.samples.chat.gui
edu.coding.samples.chat.pubsub
src
JRE System Library [JavaSE-1.8]
edu.coding.samples.pubsub
edu.coding.samples.pubsub.posix


Clean

Clean will discard all build problems and built states. The projects will be rebuilt from scratch.

[✓ Clean all projects] Clean projects selected below

- jus.util
- TD2
- TP1_Algo
- TP2_Turtle_Donné
- TP2_Turtle_Solution
- TP3_Annuaire
- TP4_Tri
- Week1.Basics.C
- Week1.Basics.Java