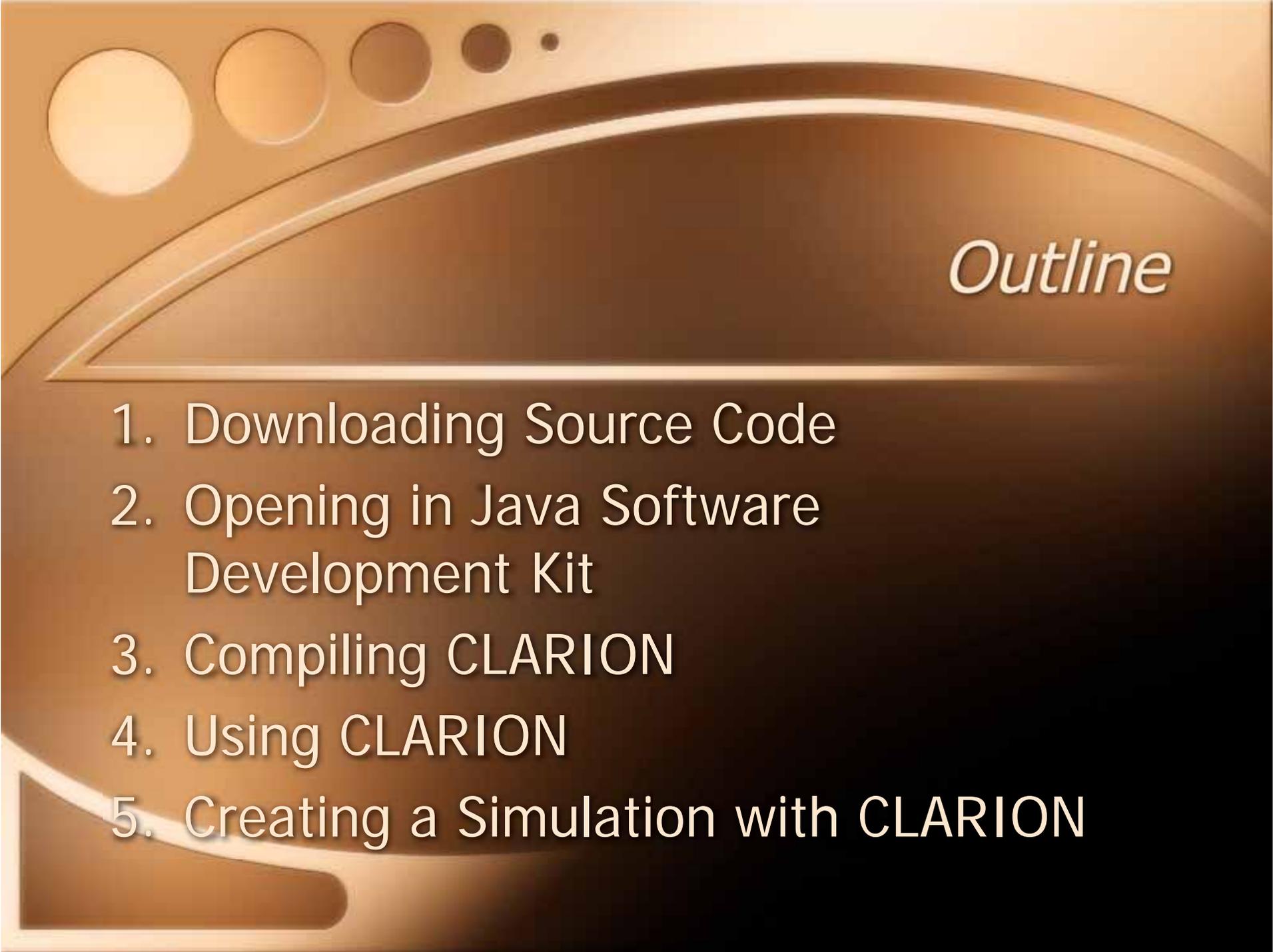


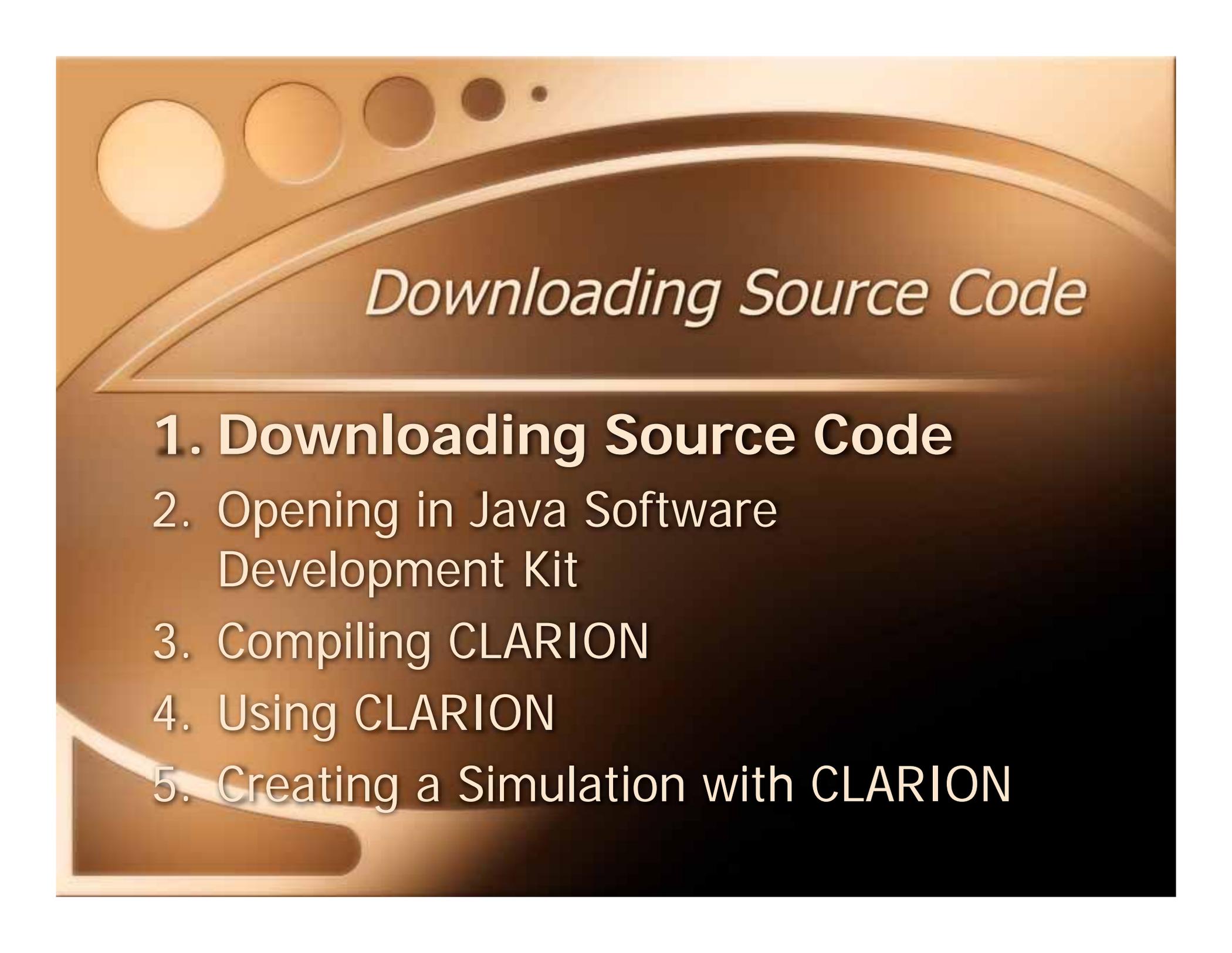
The CLARION Java Package

Nicholas Wilson, Sebastien Helie, Ron Sun
Cognitive Science, Rensselaer Polytechnic Institute



Outline

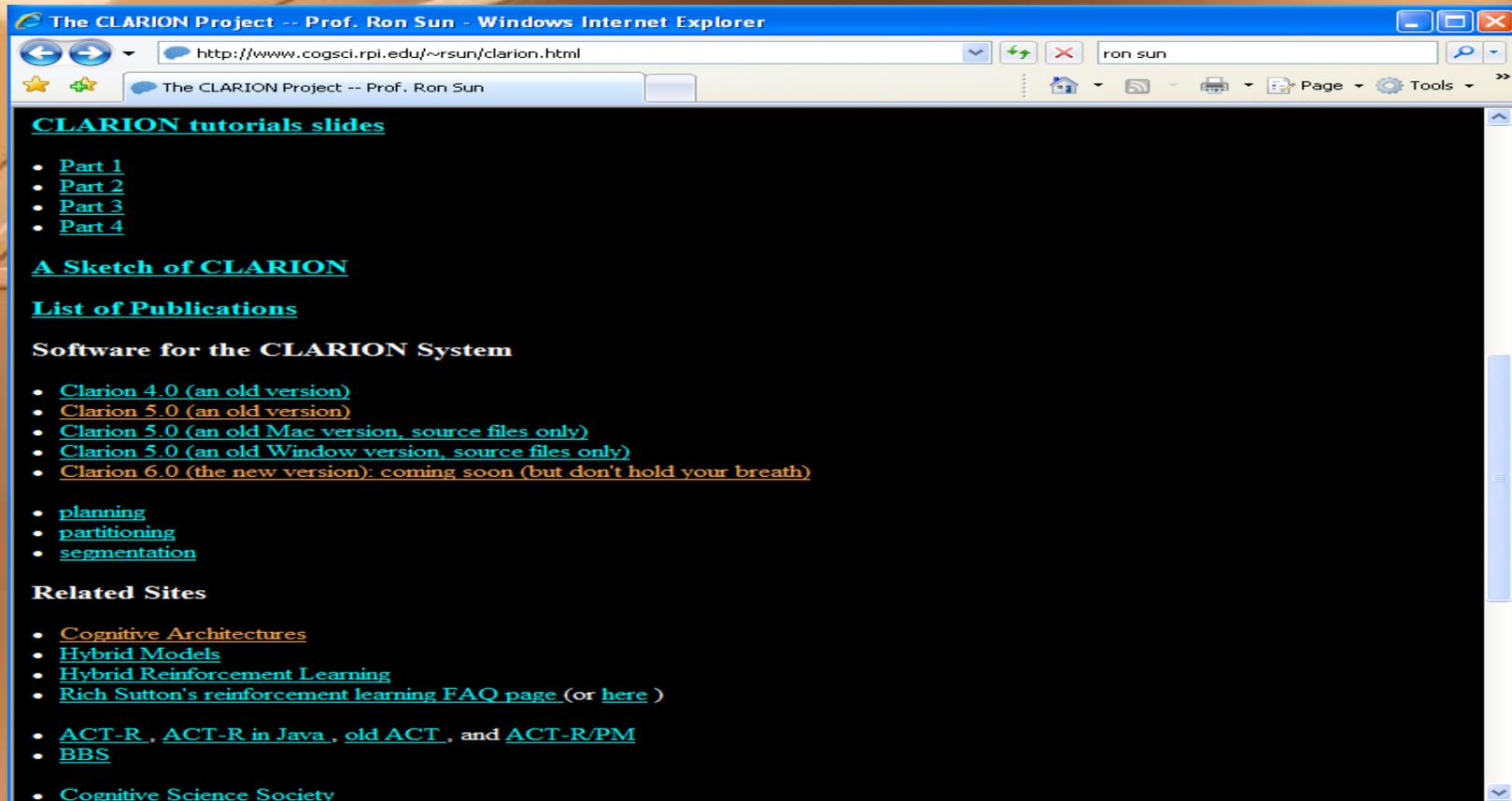
1. Downloading Source Code
2. Opening in Java Software Development Kit
3. Compiling CLARION
4. Using CLARION
5. Creating a Simulation with CLARION



Downloading Source Code

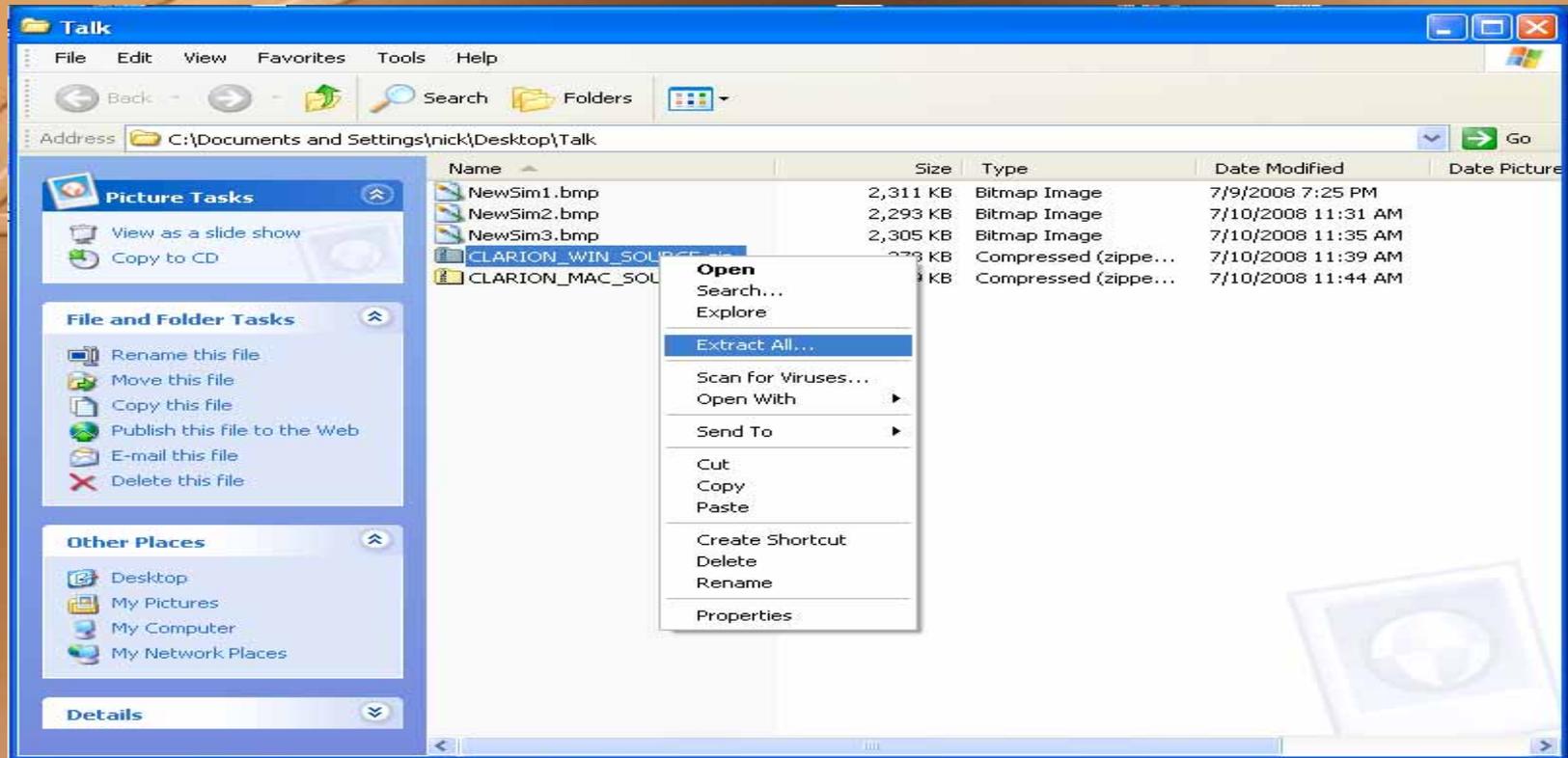
- 1. Downloading Source Code**
2. Opening in Java Software Development Kit
3. Compiling CLARION
4. Using CLARION
5. Creating a Simulation with CLARION

Downloading Source Code



- Go to <http://www.cogsci.rpi.edu/~rsun/clarion.html>
- Click on the link below to download the zip file containing the source code
 - "Clarion 5.0 (an old *** version, source files only)"

Downloading Source Code

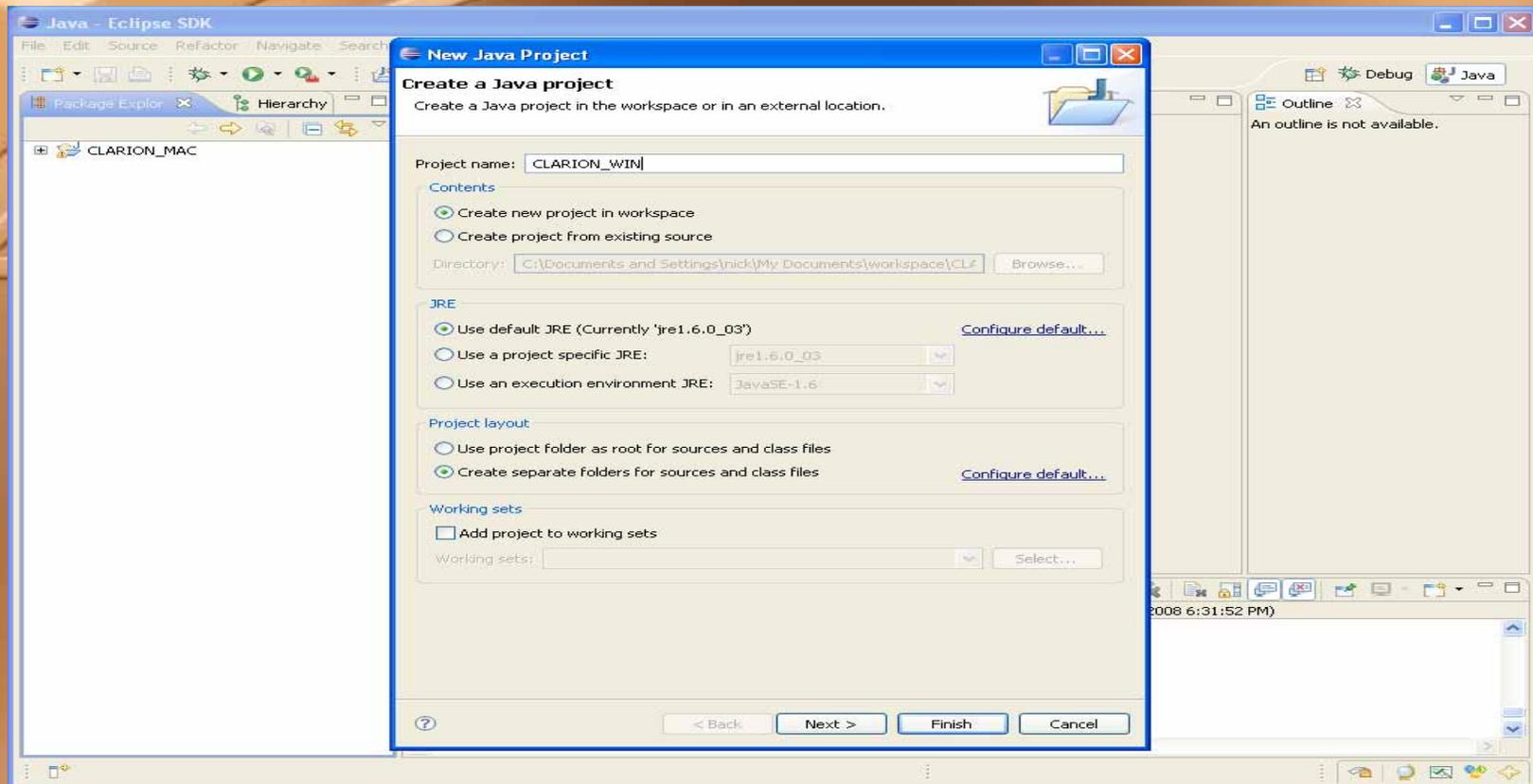


- Extract the zip file

Opening in Java SDK

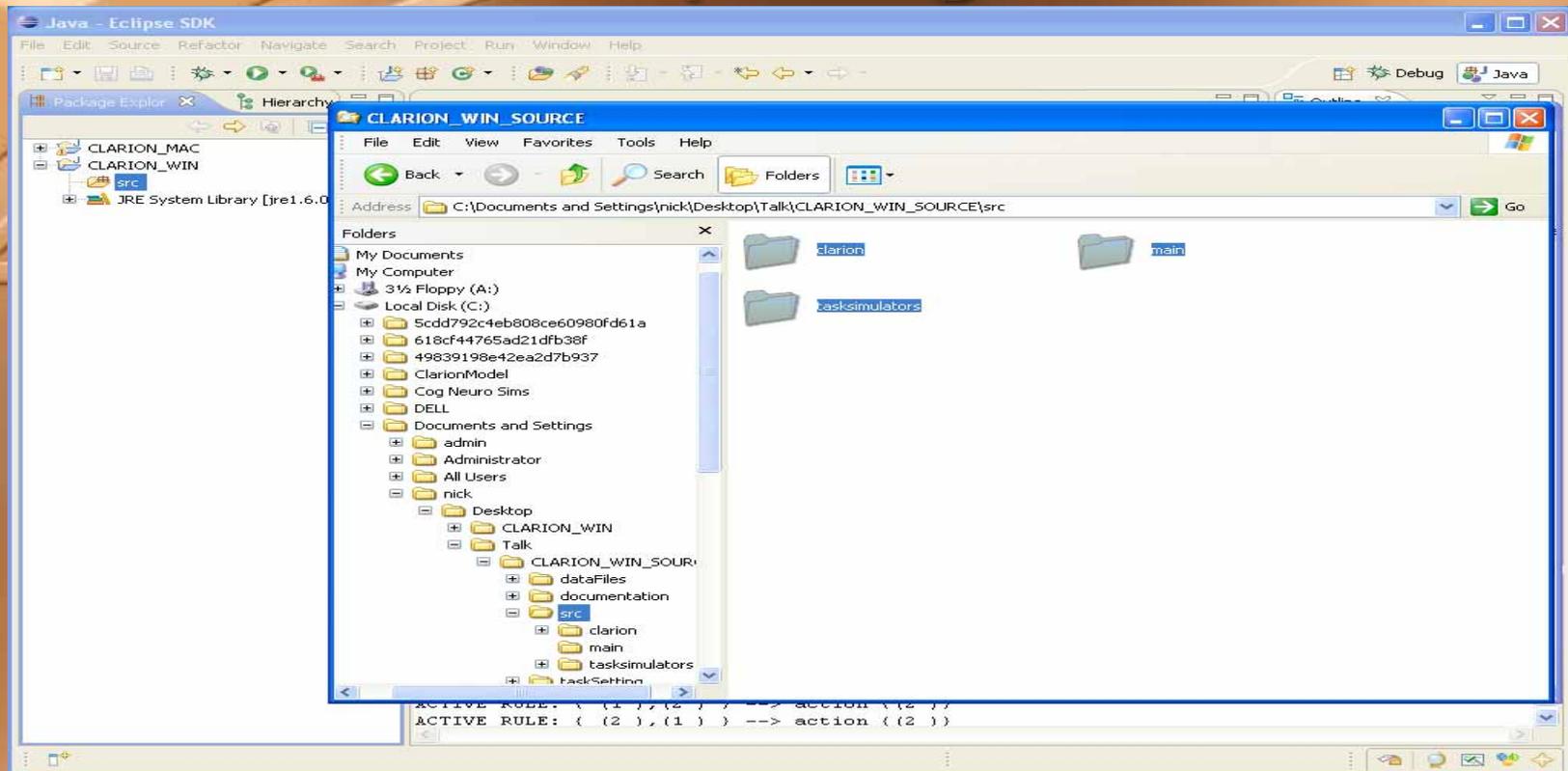
1. Downloading Source Code
- 2. Opening in Java Software Development Kit**
3. Compiling CLARION
4. Using CLARION
5. Creating a Simulation with CLARION

Opening in Java SDK



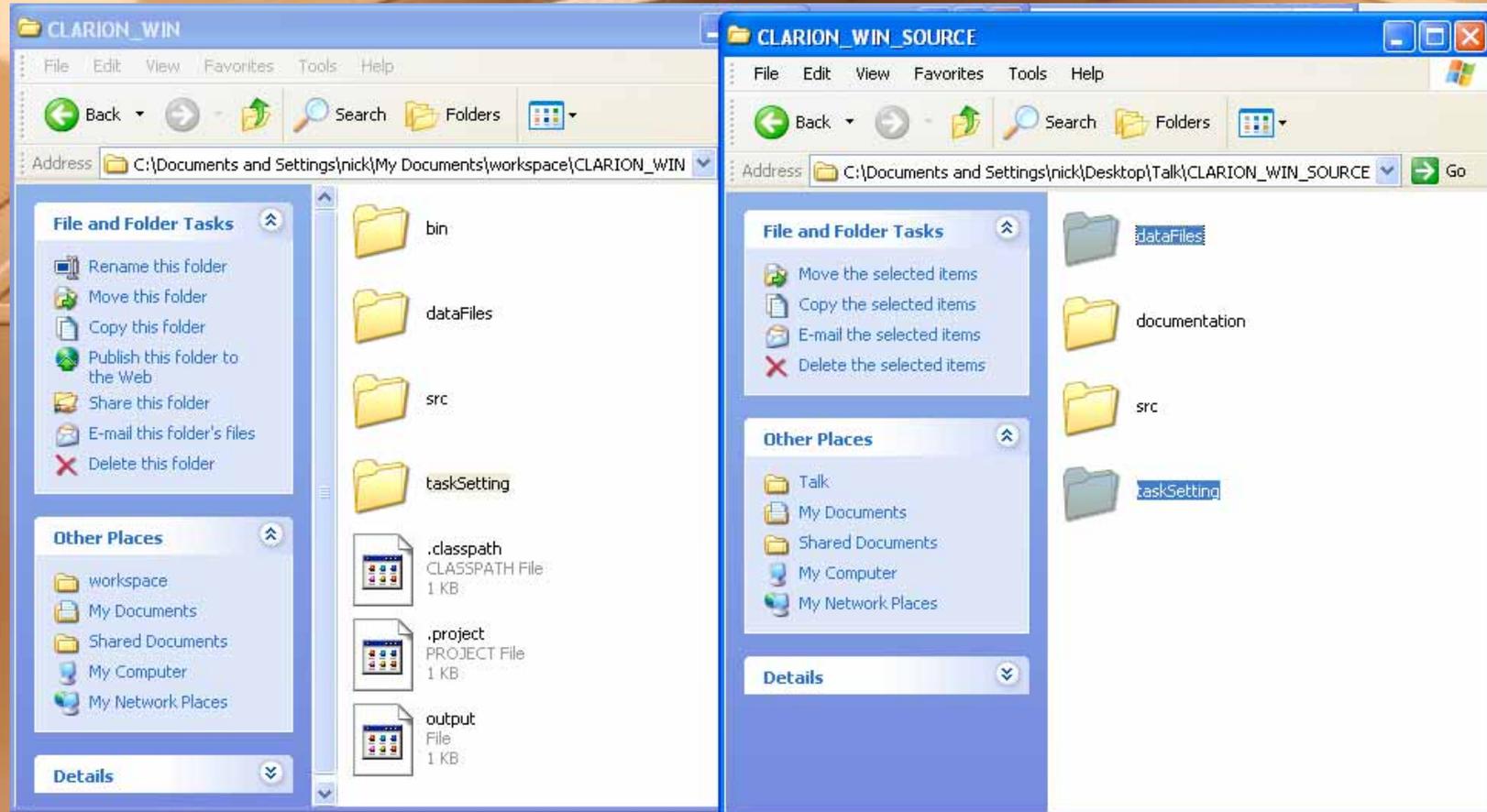
- Open the Java Development Kit (e.g. Eclipse)
- Create a new Java Project (CLARION_WIN)

Opening in Java SDK



- Open the unzipped folder labeled "src"
- Highlight the folders in the "src" folder and drag them into the SDK under the newly created project in a subfolder labeled "src"

Opening in SDK

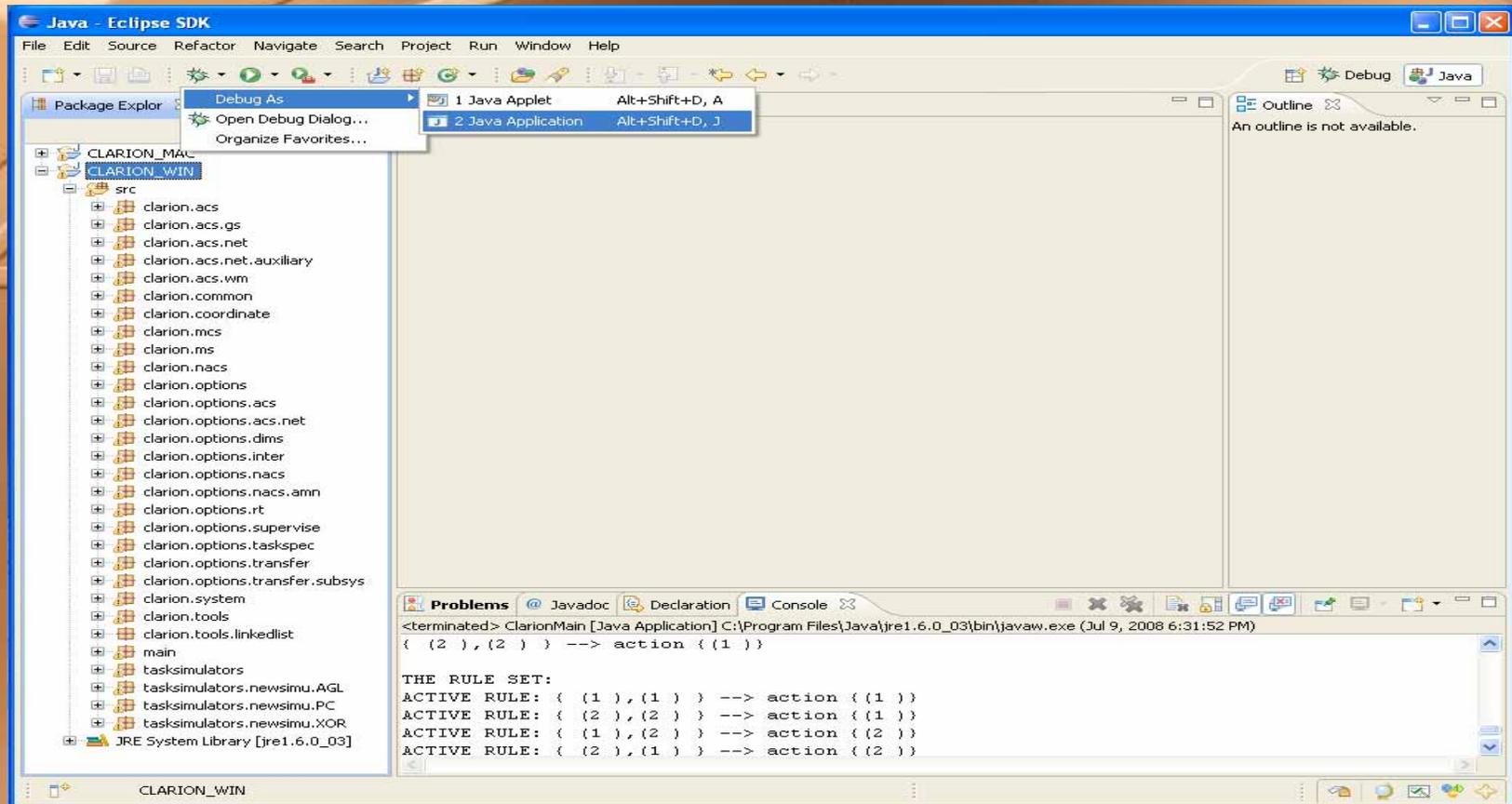


- Open the project folder (CLARION_WIN) located in the workspace folder (most likely in "My Documents")
- Copy the unzipped folders "dataFiles" and "taskSetting" to the project folder

Compiling CLARION

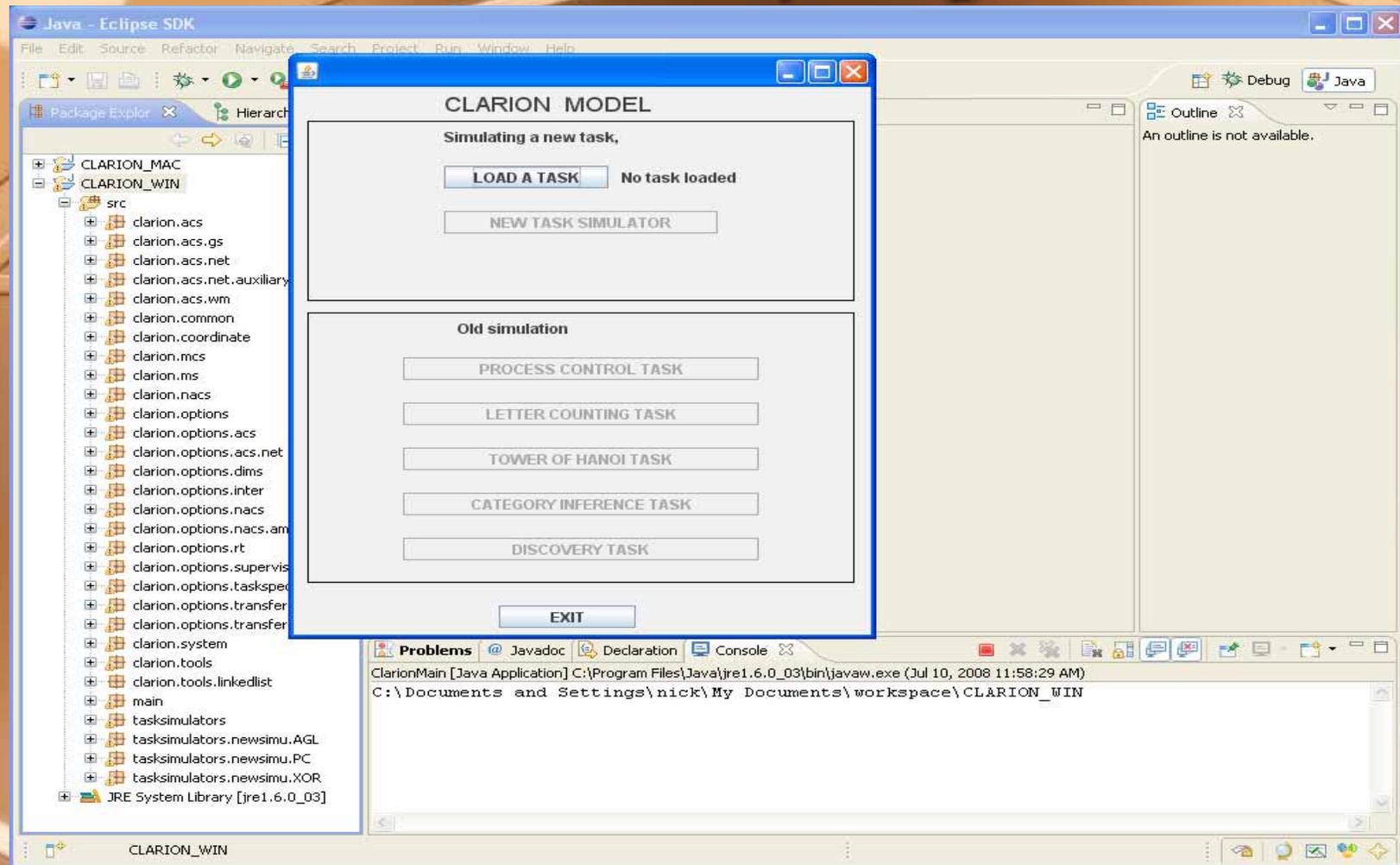
1. Downloading Source Code
2. Opening in Java Software Development Kit
- 3. Compiling CLARION**
4. Using CLARION
5. Creating a Simulation with CLARION

Compiling CLARION



- Click the arrow next to the debug button
- Choose "Debug As"
- Select "Java Application"

Compiling CLARION



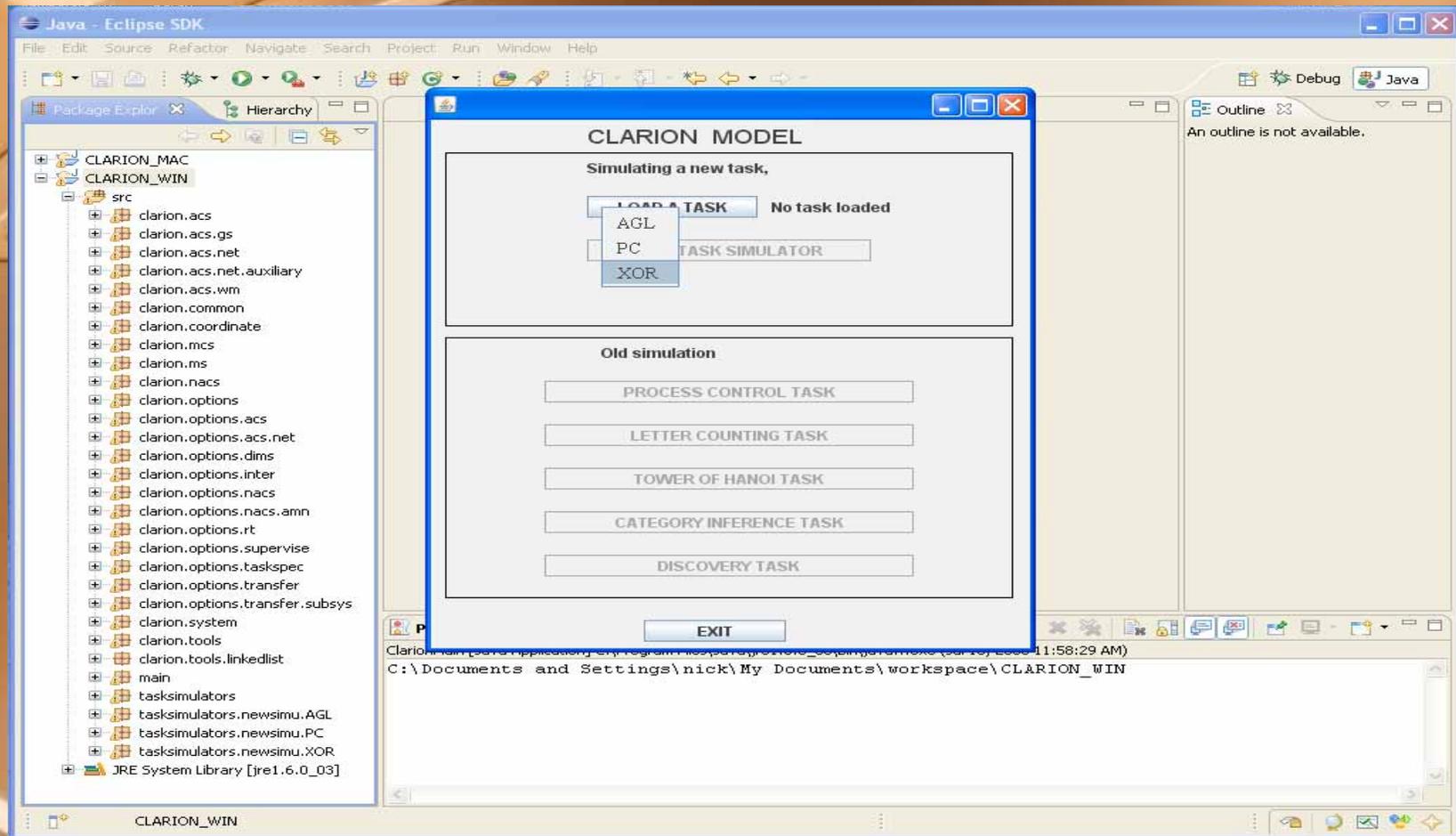
- The CLARION Model window will appear when the project has compiled



Using CLARION

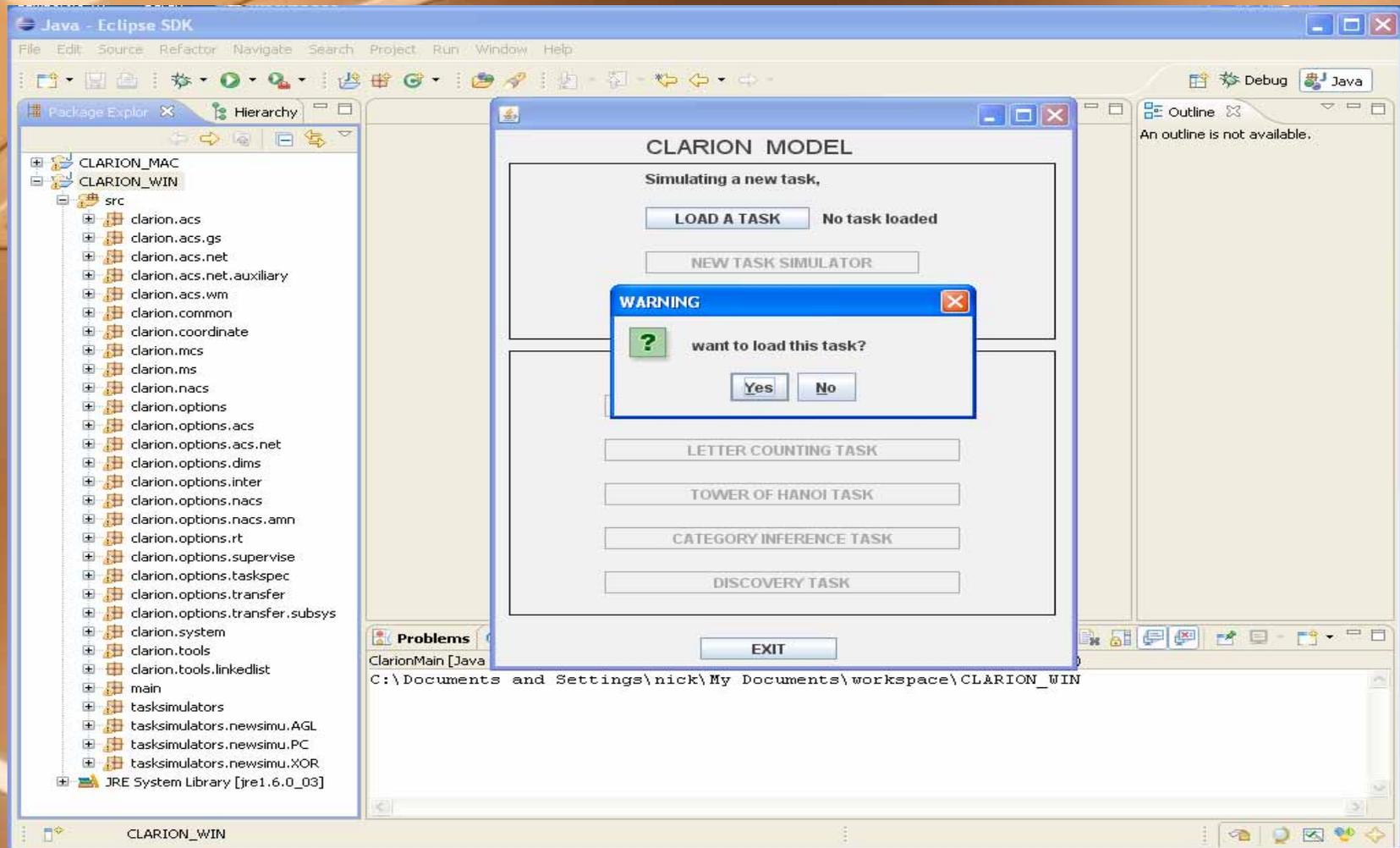
1. Downloading Source Code
2. Opening in Java Software Development Kit
3. Compiling CLARION
- 4. Using CLARION**
5. Creating a Simulation with CLARION

Using CLARION



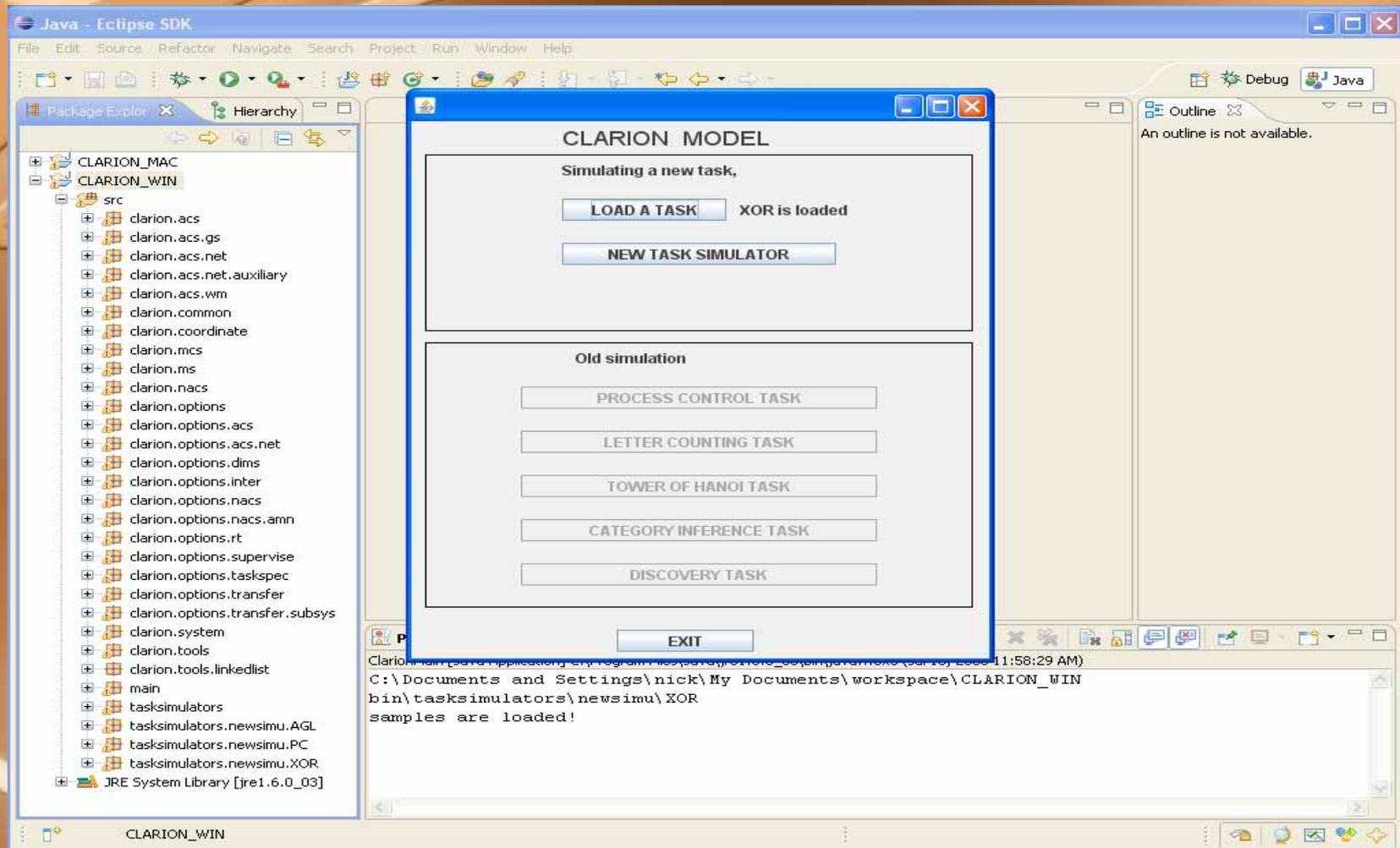
- Click on "LOAD A TASK"
- Select a task from the drop-down menu

Using CLARION



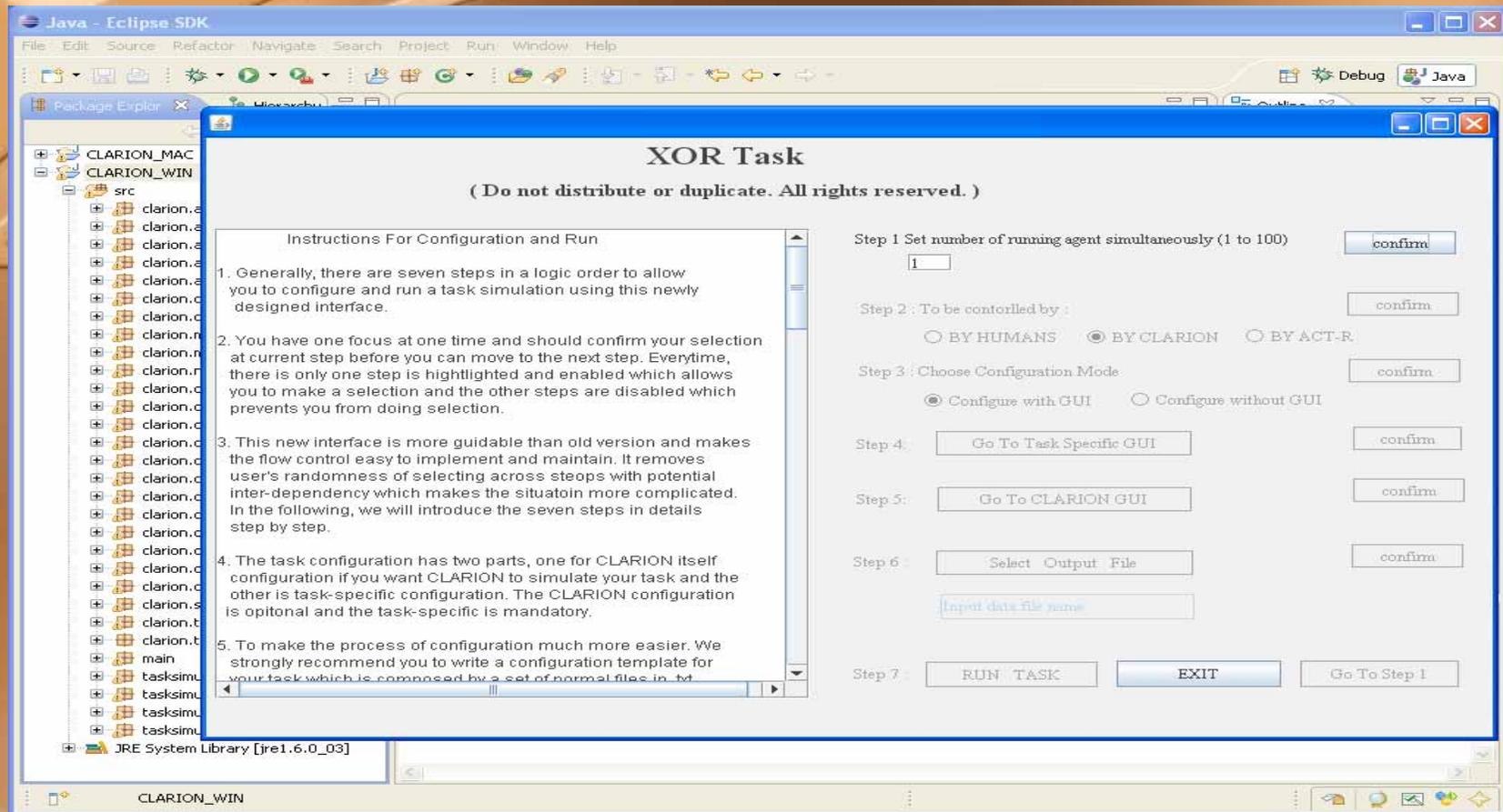
- A load task confirmation popup will appear; click "Yes"

Using CLARION



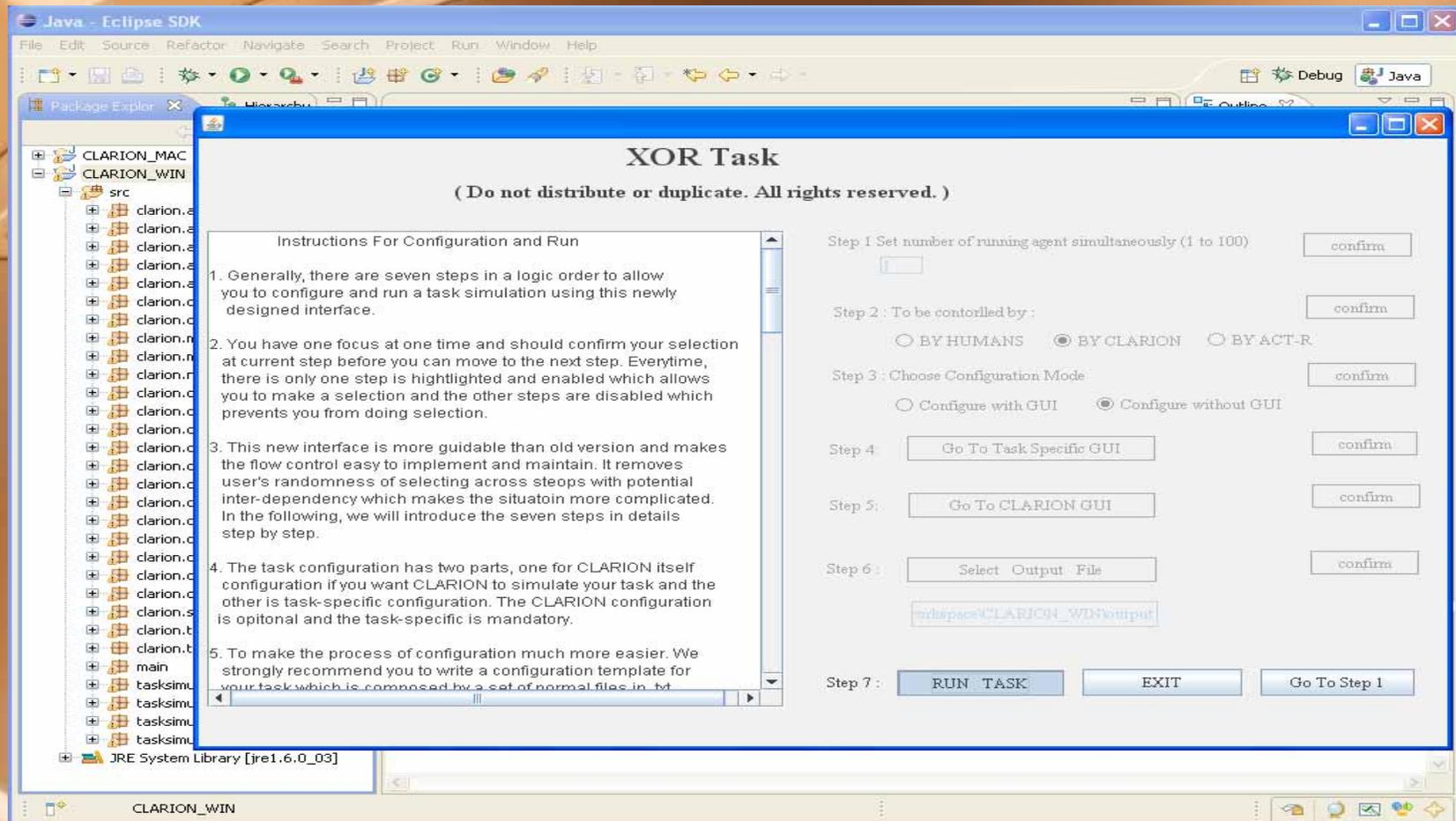
- After the task has loaded, click “NEW TASK SIMULATOR”

Using CLARION



- The task simulator setup window will appear
- Push the “confirm” button to advance to the next step
- Steps 4 - 6 have their own configuration windows. See the documentation folder for details on using the configuration windows

Using CLARION



- The preloaded tasks are already configured so "Configure without GUI" can be selected in step 3
- Once all steps have been confirmed, select "RUN TASK"

Using CLARION

The screenshot shows the 'XOR Task' configuration window in a Java IDE. The window title is 'XOR Task' and it contains a warning message: '(Do not distribute or duplicate. All rights reserved.)'. The main area is titled 'Instructions For Configuration and Run' and contains five numbered instructions. To the right, there are seven steps for configuration, each with a 'confirm' button. A 'WARNING' popup window is overlaid on the main window, containing a question mark icon and the text: 'All options should be unchanged during task running, start a new task?'. Below the warning are 'Yes' and 'No' buttons. At the bottom of the main window, there are buttons for 'RUN TASK', 'EXIT', and 'Go To Step 1'. The IDE's status bar at the bottom shows 'CLARION_WIN' and a console window with the following output:

```
ClarionMain [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (Jul 10, 2008 12:04:42 PM)
C:\Documents and Settings\nick\My Documents\workspace\CLARION_WIN
bin\tasksimulators\newsimu\XOR
samples are loaded!
```

- A warning popup will appear; click "Yes"

Using CLARION

XOR Task
(Do not distribute or duplicate. All rights reserved.)

Instructions For Configuration and Run

- Generally, there are seven steps in a logic order to allow you to configure and run a task simulation using this newly designed interface.
- You have one focus at one time and should confirm your selection at current step before you can move to the next step. Everytime, there is only one step is highlighted and enabled which allows you to make a selection and the other steps are disabled which prevents you from doing selection.
- This new interface is more guidable than old version and makes the flow control easy to implement and maintain. It removes user's randomness of selecting across steeps with potential inter-dependency which makes the situatoin more complicated. In the following, we will introduce the seven steps in details step by step.
- The task configuration has two parts, one for CLARION itself configuration if you want CLARION to simulate your task and the other is task-specific configuration. The CLARION configuration is opitonal and the task-specific is mandatory.
- To make the process of configuration much more easier. We strongly recommend you to write a configuration template for your task which is composed by a set of normal files in .txt

Step 1: Set number of running agent simultaneously (1 to 100) confirm

Step 2: To be controlled by: BY HUMANS BY CLARION BY ACT-R confirm

Step 3: Choose Configuration Mode Configure with GUI Configure without GUI confirm

Step 4: confirm

Step 5: confirm

Step 6: confirm

Step 7:

```
Match con
( { 2 }, { 2 }
Match bot
( { 2 }, { 2 } } --> action { { 1 } }
THE RULE SET:
ACTIVE RULE: { { 2 }, { 2 } } --> action { { 1 } }
ACTIVE RULE: { { 2 }, { 1 } } --> action { { 2 } }
ACTIVE RULE: { { 1 }, { 1 } } --> action { { 1 } }
ACTIVE RULE: { { 1 }, { 2 } } --> action { { 2 } }
```

- All simulation output (including results) will appear in the console window of the Java SDK
- The results will also be output to the output file specified in step 6

Using CLARION: without GUI

```
Java - CLARION_WIN/src/main/ClarionMain.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer Hierarchy
CLARION_MAC
CLARION_WIN
src
  clarion.acs
  clarion.acs.gs
  clarion.acs.net
  clarion.acs.net.auxiliary
  clarion.acs.wm
  clarion.common
  clarion.coordinate
  clarion.mcs
  clarion.ms
  clarion.nacs
  clarion.options
  clarion.options.acs
  clarion.options.acs.net
  clarion.options.dims
  clarion.options.inter
  clarion.options.nacs
  clarion.options.nacs.amn
  clarion.options.rt
  clarion.options.supervise
  clarion.options.taskspec
  clarion.options.transfer
  clarion.options.transfer.subsys
  clarion.system
  clarion.tools
  clarion.tools.linkedlist
  main
    ClarionMain.java
  tasksimulators
  tasksimulators.newsimu.AGL
  tasksimulators.newsimu.PC
  tasksimulators.newsimu.XOR
  JRE System Library [jre1.6.0_03]

ClarionMain.java
//newTaskPanel.add( clarionButton );
oldTaskPanel.add( oldTaskLabel );
this.getContentPane().add( capLabel, null );
this.getContentPane().add( newTaskPanel, null );
this.getContentPane().add( oldTaskPanel, null );
this.getContentPane().add( exitButton, null );
validate();
setEnabled( true );
setVisible( true );
// for old task simulations.
//setModelParaSet();
//setTaskParaSet();
}

/** the entry of the whole system to start. */
public static void main( String args[] )
{
  ClarionMain aDemo = new ClarionMain();
}

/** Search current simulated tasks.
 * @param names the array to store the simulated task names. */
public int searchCurrentTasks( Vector names )
{
  String path = Global.getTaskDir();
  File file = new File( path );
  int len = file.list().length;
  for( int i = 0; i < len; i++ )

```

Outline

- simuType : int
- loadTaskName : SI
- newTaskNames : \
- thisFrame : JFram
- capLabel : JLabel
- oldTaskPanel : JPe
- oldTaskLabel : JLa
- oldTaskButtons : :
- newTaskPanel : JF
- newTaskLabel : JL
- loadLabel : JLabel
- loadButton : JButt
- newTaskButton : :
- loadTaskMenu : JF
- loadTaskItem : JM
- exitButton : JButt
- newTask : Task
- taskClarion : Task
- oldSimulators : Sim
- newSimulator : Ne
- ClarionMain() : Ne
- ClarionMain()
- initGui()
- new ActionLi:
- new ActionLi:
- new ActionLi:
- main(String[])

Problems @ Javadoc Declaration Console Debug

ClarionMain [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (Jul 10, 2008 1:50:12 PM)

```
THE RULE SET:
ACTIVE RULE: { ( 2 ), ( 2 ) } --> action { ( 1 ) }
ACTIVE RULE: { ( 2 ), ( 1 ) } --> action { ( 2 ) }
ACTIVE RULE: { ( 1 ), ( 1 ) } --> action { ( 1 ) }
ACTIVE RULE: { ( 1 ), ( 2 ) } --> action { ( 2 ) }
```

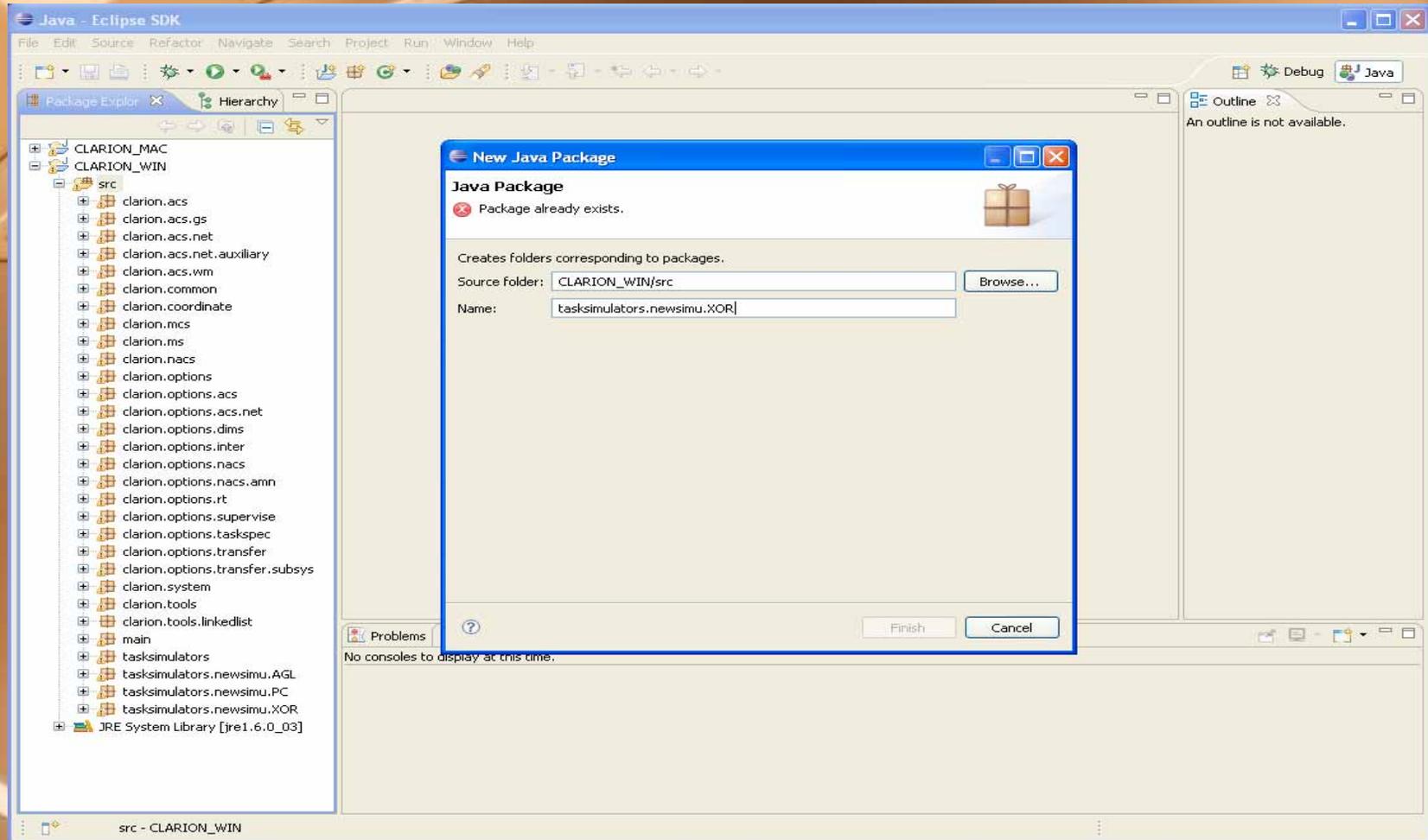
- The "main" method is located in ClarionMain.java
- The java implementation of the CLARION subsystems can be implemented without the GUI by simply commenting out the ClarionMain instantiation in the main method and replacing it with your own simulation class instantiation



Creating a Simulation with CLARION

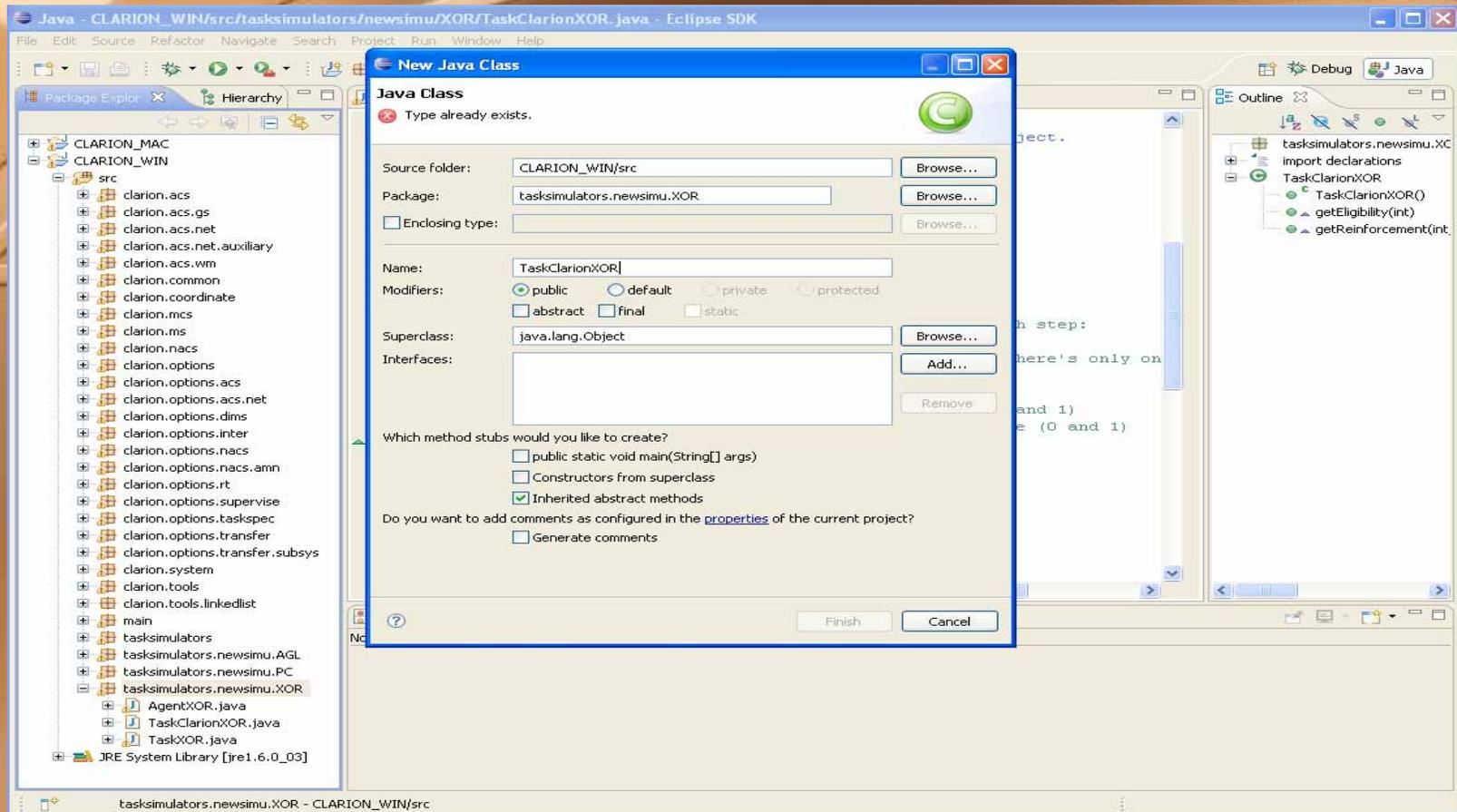
1. Downloading Source Code
2. Opening in Java Software Development Kit
3. Compiling CLARION
4. Using CLARION
- 5. Creating a Simulation with
CLARION**

Creating a Simulation with CLARION



- Create a new Java package under "src" in the "CLARION_WIN" project
- Name the package "tasksimulators.newtasksimu. XOR "

Creating a Simulation with CLARION



- Create the following classes:
 - Agent_XOR_.java
 - TaskClarion_XOR_.java
 - Task_XOR_.java

Creating a Simulation with CLARION

```
Java - CLARION_WIN/src/tasksimulators/newsimu/XOR/TaskClarionXOR.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer | Hierarchy | AgentXOR.java | TaskClarionXOR.java | TaskXOR.java | Outline | Debug | Java

CLARION_MAC
├── CLARION_WIN
│   └── src
│       ├── clarion.acs
│       ├── clarion.acs.gs
│       ├── clarion.acs.net
│       ├── clarion.acs.net.auxiliary
│       ├── clarion.acs.wm
│       ├── clarion.common
│       ├── clarion.coordinate
│       ├── clarion.mcs
│       ├── clarion.ms
│       ├── clarion.nacs
│       ├── clarion.options
│       ├── clarion.options.acs
│       ├── clarion.options.acs.net
│       ├── clarion.options.dims
│       ├── clarion.options.inter
│       ├── clarion.options.nacs
│       ├── clarion.options.nacs.amn
│       ├── clarion.options.rt
│       ├── clarion.options.supervise
│       ├── clarion.options.taskspec
│       ├── clarion.options.transfer
│       ├── clarion.options.transfer.subsys
│       ├── clarion.system
│       ├── clarion.tools
│       ├── clarion.tools.linkedlist
│       ├── main
│       ├── tasksimulators
│       ├── tasksimulators.newsimu.AGL
│       ├── tasksimulators.newsimu.PC
│       └── tasksimulators.newsimu.XOR
│           ├── AgentXOR.java
│           ├── TaskClarionXOR.java
│           └── TaskXOR.java
└── JRE System Library [jre1.6.0_03]

* <li> Usage of this class. <br>
* This class is instantiated inside a <tt>TaskAgent</tt> object.
* </ul>
* @author Xi Zhang
*/

public class TaskClarionXOR extends TaskClarion{

    public TaskClarionXOR ()
    {
    }

    // Determines which of the IDN networks is eligible at each step:
    // * The outermost array stores all the eligibility data.
    // * The middle array (there is only one) indicates that there's only on
    // * The inner arrays indicate, respectively,
    //   - That for each possible input dimension (0 or 1)
    //   - Both values of the first dimension are eligible (0 and 1)
    //   - And both values of the second dimension are eligible (0 and 1)

    public short[][] [] getEligibility( int netIdx )
    {
        short[][] [] eligCondition = {
            {
                {0, 1},
                {1, 1},
                {1, 1}
            },
            . . .
        }
    }
}

tasksimulators.newsimu.XC
import declarations
TaskClarionXOR
├── TaskClarionXOR()
├── getEligibility(int)
└── getReinforcement(int)

Problems | Javadoc | Declaration | Console | Debug
No consoles to display at this time.

Writable | Smart Insert | 2 : 1
```

- Each of the newly created classes extend from other classes in the CLARION model (e.g. TaskClarionXOR extends TaskClarion)

Creating a Simulation with CLARION

```
Java - CLARION_WIN/src/clarion/system/TaskClarion.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer Hierarchy Task.java SimuTask.java TaskAgent.java TaskClarion.java
CLARION_MAC
CLARION_WIN
src
  clarion.acs
  clarion.acs.gs
  clarion.acs.net
  clarion.acs.net.auxiliary
  clarion.acs.wm
  clarion.common
  clarion.coordinate
  clarion.mcs
  clarion.ms
  clarion.nacs
  clarion.options
  clarion.options.acs
  clarion.options.acs.net
  clarion.options.dims
  clarion.options.inter
  clarion.options.nacs
  clarion.options.nacs.amn
  clarion.options.rt
  clarion.options.supervise
  clarion.options.taskspec
  clarion.options.transfer
  clarion.options.transfer.subsys
  clarion.system
    Clarion.java
    ClarionClassLoader.java
    Global.java
    SimuTask.java
    Task.java
    TaskAgent.java
    TaskClarion.java
  clarion.tools
  clarion.tools.linkedlist
  main
  tasksimulators

TaskClarion.java
//===== the followings need be override by user's own codes. =====
/** Returns the goals. The returned value is in the format of 2-dimension
 * The first dimension indexes a specific goal and the second dimension i
 * specific goal dimension. We assume that a goal is composed by an array
 * each corresponding to a specific goal dimension.
 * @return the goal set. */
public short[][] getGoals()
{
    return (short[][] )null;
}

/** Returns the goal dimension-value info. The returned value is in the fo
 * dimensional array each slot storing the corresponding dimension value
 * @return the goal dimension-value info */
public short[] getGoalDimDVs()
{
    return (short[])null;
}

/** bottom-up rectification process described in the Chapter on ACS in tut
 */
public void combineByBottomUpRectification()
{
}

/** top-down guidance process described in the Chapter on ACS in tutorial.
 */
public void combineByTopDownGuidance()
{
}

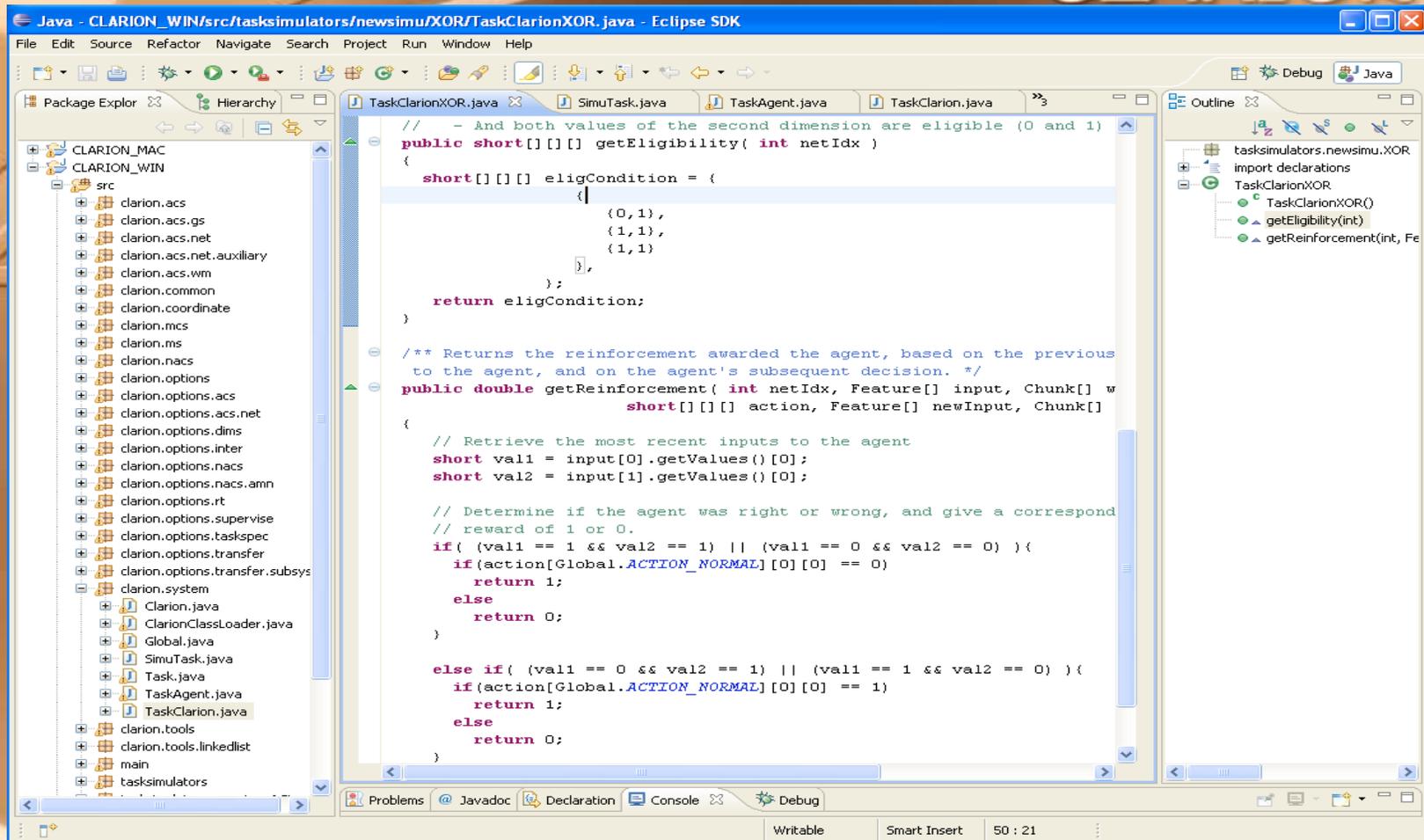
Outline
clarion.system
import declarations
TaskClarion
  SUB_SYS_NUM : ir
  ACS : int
  NACS : int
  RT : int
  MCS : int
  ACS_SAMPLE_NUM
  NACS_SAMPLE_NUM
  RT_SAMPLE_NUM
  MCS_SAMPLE_NUM
  GOAL : int
  GOAL_DIM : int
  BUR : int
  TDG : int
  ELIG : int
  REIN : int
  RER_WGT_C : int
  RER_WGT_R : int
  RER_UTL : int
  RER_POS : int
  RER_IG : int
  IRL_WGT_C : int
  IRL_WGT_R : int
  IRL_UTL : int

Problems Javadoc Declaration Console Debug
No consoles to display at this time.

Writable Smart Insert 1 : 1
```

- The parent classes have specified methods that need to be overwritten with simulation specific information

Creating a Simulation with CLARION



```
Java - CLARION_WIN/src/tasksimulators/newsimu/XOR/TaskClarionXOR.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explor Hierarchy TaskClarionXOR.java SimuTask.java TaskAgent.java TaskClarion.java
// - And both values of the second dimension are eligible (0 and 1)
public short[][][] getEligibility( int netIdx )
{
    short[][][] eligCondition = {
        {
            {0,1},
            {1,1},
            {1,1}
        },
    };
    return eligCondition;
}

/** Returns the reinforcement awarded the agent, based on the previous
to the agent, and on the agent's subsequent decision. */
public double getReinforcement( int netIdx, Feature[] input, Chunk[] w
    short[][][] action, Feature[] newInput, Chunk[]

    // Retrieve the most recent inputs to the agent
    short val1 = input[0].getValues()[0];
    short val2 = input[1].getValues()[0];

    // Determine if the agent was right or wrong, and give a correspond
    // reward of 1 or 0.
    if( (val1 == 1 && val2 == 1) || (val1 == 0 && val2 == 0) ){
        if(action[Global.ACTION_NORMAL][0][0] == 0)
            return 1;
        else
            return 0;
    }

    else if( (val1 == 0 && val2 == 1) || (val1 == 1 && val2 == 0) ){
        if(action[Global.ACTION_NORMAL][0][0] == 1)
            return 1;
        else
            return 0;
    }
}

tasksimulators.newsimu.XOR
import declarations
TaskClarionXOR
TaskClarionXOR()
getEligibility(int)
getReinforcement(int, Fe
```

- Above is an example of the overwritten methods in TaskClarionXOR.java
- Note that not all methods in the parent class need to be overwritten, only methods needed to run the task

Creating a Simulation with CLARION

```
Java - CLARION_WIN/src/clarion/system/TaskAgent.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer Hierarchy AgentXOR.java TaskClarionXOR.java TaskAgent.java TaskClarion.java
CLARION_MAC
CLARION_WIN
src
  clarion.acs
  clarion.acs.gs
  clarion.acs.net
  clarion.acs.net.auxiliary
  clarion.acs.wm
  clarion.common
  clarion.coordinate
  clarion.mcs
  clarion.ms
  clarion.nacs
  clarion.options
  clarion.options.acs
  clarion.options.acs.net
  clarion.options.dims
  clarion.options.inter
  clarion.options.nacs
  clarion.options.nacs.amn
  clarion.options.rt
  clarion.options.supervise
  clarion.options.taskspec
  clarion.options.transfer
  clarion.options.transfer.subsys
  clarion.system
    Clarion.java
    ClarionClassLoader.java
    Global.java
    SimuTask.java
    Task.java
    TaskAgent.java
    TaskClarion.java
  clarion.tools
  clarion.tools.linkedlist
  main
  tasksimulators

public void setLearningTest( boolean flag )
{
    learningTestOn = flag;
}

/** save the current into the specified data file. */
public void saveOutput( String results )
{
    // dataFile.saveStringData( results );
}

/** perform the chosen action.
 * @param netIdx the ACS network index.
 * @param action the selected actions by ACS networks. */
public void performAction( int netIdx, short[] [] [] action )
{
    if( global.isNacsOn() )
        global.getNACS().performAction( action );
}

/** Returns the output of current step.
 * @param arr the array to be filled in with current output. */
abstract public void getOutput( Object arr );
/** Determines which AMN will be used given current action.
 * @param netIdx the ACS network index.
 * @param action the selected actions by ACS networks.
 * @return the AMN network index. */
abstract public int determineAmnNet( int netIdx, short[] [] [] action )
/** send the retrieval results from NACS to ACS components.
 * @param netIdx the ACS network index.
 * @param results retrieval results from NACS */
abstract public void setRetrievalResults( int netIdx, GKSCChunk[] resu
/** simulate a subject performance by step. */
abstract public void run();
}
```

Outline: clarion.system, import declarations, TaskAgent, global: Global, clarion: Clarion, taskClarion: TaskClarion, curSensoryInput: Featur, curNacsInput: Feature[], nacsResults: Feature[], desiredOutput: Feature[], reinforcements: double[], activeAmnIdx: int, learningTestOn: boolean, TaskAgent(), TaskAgent(Global), reinit(), isInLearningTest(), getActiveAmnIdx(), getRerRules(int), getIrrRules(int), getFixRules(int), getCurSensoryInput(), getCurNacsInput(), getCurDesiredOutput(), setLearningTest(boolean), saveOutput(String), performAction(int, short[], short[], short[]), getOutput(Object), determineAmnNet(int, short[], short[], short[]), setRetrievalResults(int, GKSCChunk[]), run()

- Some of the methods in the parent class are abstract and must be overwritten in the child class

Creating a Simulation with CLARION

The screenshot shows the Eclipse IDE with the following components:

- Package Explorer:** Shows a project structure with packages like `clarion.acs`, `clarion.common`, and `clarion.system`.
- Code Editor:** Displays the implementation of `AgentXOR.java`. The visible code includes:

```
public void setSensoryInput(short[] inputs)
{
    // Each input dimension is encoded as an array of values
    short[] values1 = { inputs[0] };
    short[] values2 = { inputs[1] };
    // Sets the input (see clarion.common/Feature.java for the declara
    // of the Feature() constructor). Some notes on this constructor:
    // * Each feature is referenced by its name (1st param)
    // * 'Type' will normally be 0 (2nd param)
    // * 'Order' will be 0 for 1st feature, 1 for second, etc. (3rd pa
    // * 'Length' is the number of values in each dimension (4th param
    curSensoryInput[0] = new Feature( global.getFeatureNames() [Global.
    (short)Global.FT_INPUT, (short)0
    (short)Global.FT_INPUT, (short)1

    // is the chosen action is an action for controlling NACS?
    public boolean isControlNacsAction( int netIdx, short[] chosenAction
    {
        return false;
    }

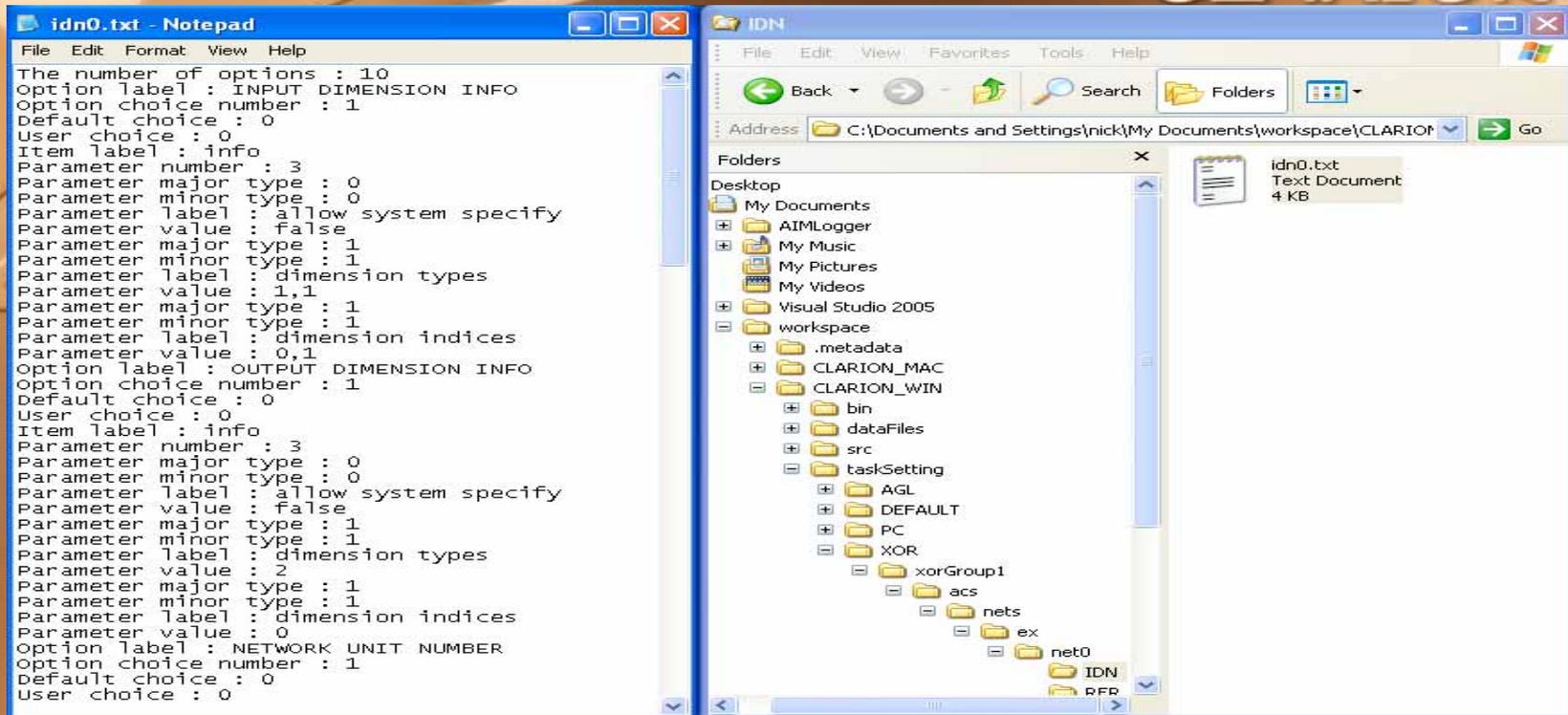
    // Gets the output of current step. Since there is only one "action"
    // (output) in the XOR task, we set the first value of the array acco
    public void getOutput( Object arr )
    {
        ((int[])arr)[0] = lastResult;
    }

    // determine which AMN will be used given current action.
    public int determineAmnNet( int netIdx, short[][][] action ) {
        return 0;
    }

    // send the retrieval results to ACS components.
    public void setRetrievalResults( int netIdx, GKSCChunk[] results ) {}
```
- Outline:** Shows the class hierarchy for `AgentXOR`, including methods like `setSensoryInput`, `isControlNacsAction`, `getOutput`, `determineAmnNet`, `setRetrievalResults`, and `run`.

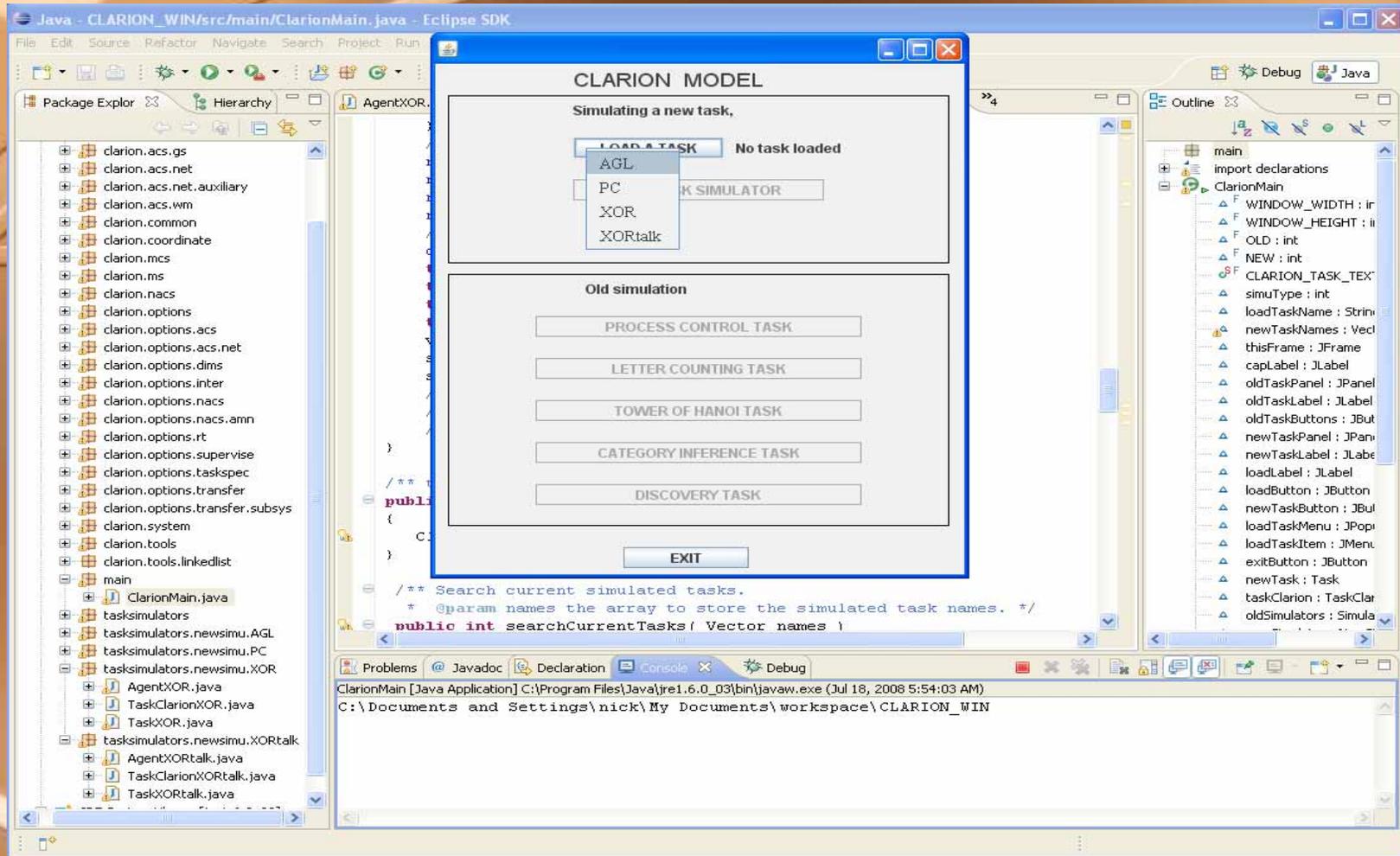
- Above is an example of the abstract methods from `TaskAgent.java` being overwritten in `AgentXOR.java`

Creating a Simulation with CLARION



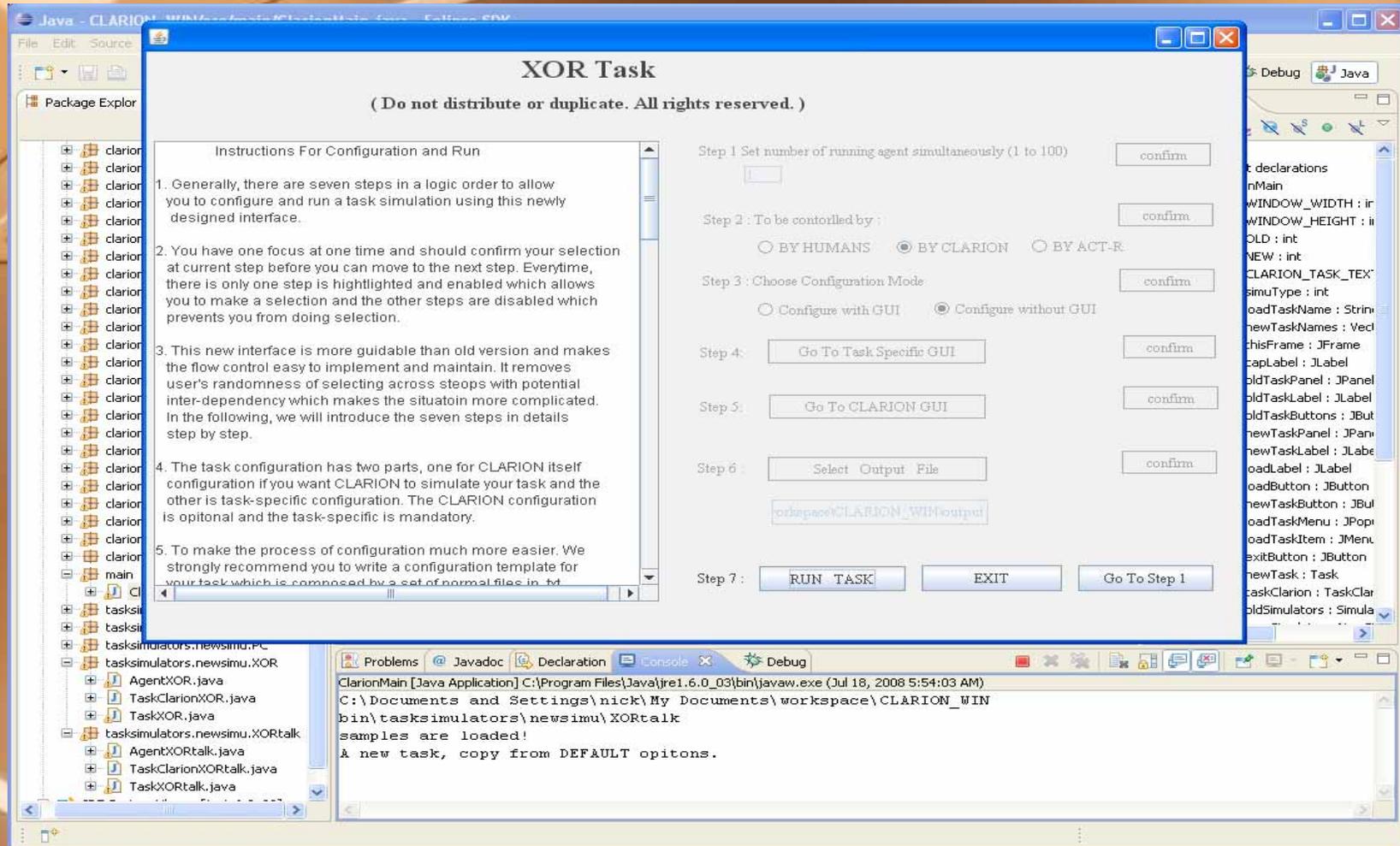
- There are several options for setting the parameters in CLARION (see the appendix of the tutorial for a list of the parameters and their default settings):
 - Manually configuring the settings in the various text files located under “taskSetting” (demonstrated above, advanced)
 - Using the GUI (see documentation for further details, buggy)
 - In code via direct manipulation of variables located in the “Global” class (best method)

Creating a Simulation with CLARION



- After the simulation is written, compile and run the project
- The new simulation should appear in the drop-down menu

Creating a Simulation with CLARION



XOR Task
(Do not distribute or duplicate. All rights reserved.)

Instructions For Configuration and Run

1. Generally, there are seven steps in a logic order to allow you to configure and run a task simulation using this newly designed interface.
2. You have one focus at one time and should confirm your selection at current step before you can move to the next step. Everytime, there is only one step is highlighted and enabled which allows you to make a selection and the other steps are disabled which prevents you from doing selection.
3. This new interface is more guidable than old version and makes the flow control easy to implement and maintain. It removes user's randomness of selecting across steps with potential inter-dependency which makes the situation more complicated. In the following, we will introduce the seven steps in details step by step.
4. The task configuration has two parts, one for CLARION itself configuration if you want CLARION to simulate your task and the other is task-specific configuration. The CLARION configuration is optional and the task-specific is mandatory.
5. To make the process of configuration much more easier. We strongly recommend you to write a configuration template for your task which is composed by a set of normal files in .td

Step 1: Set number of running agent simultaneously (1 to 100) confirm

Step 2: To be controlled by :
 BY HUMANS BY CLARION BY ACT-R confirm

Step 3: Choose Configuration Mode
 Configure with GUI Configure without GUI confirm

Step 4: confirm

Step 5: confirm

Step 6: confirm

Step 7:

```
ClarionMain [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (Jul 18, 2008 5:54:03 AM)
C:\Documents and Settings\nick\My Documents\workspace\CLARION_WIN
bin\tasksimulators\newsimu\XORTalk
samples are loaded!
A new task, copy from DEFAULT options.
```

- If methods 1 or 3 are used to set the CLARION parameters, select "Configure without GUI" in step 3.

The CLARION Java Package

These slides as well as further documentation can be found in the Java packages located on the website.

<http://www.cogsci.rpi.edu/~rsun/clarion.html>

Thank You
Questions?