Java Programming I

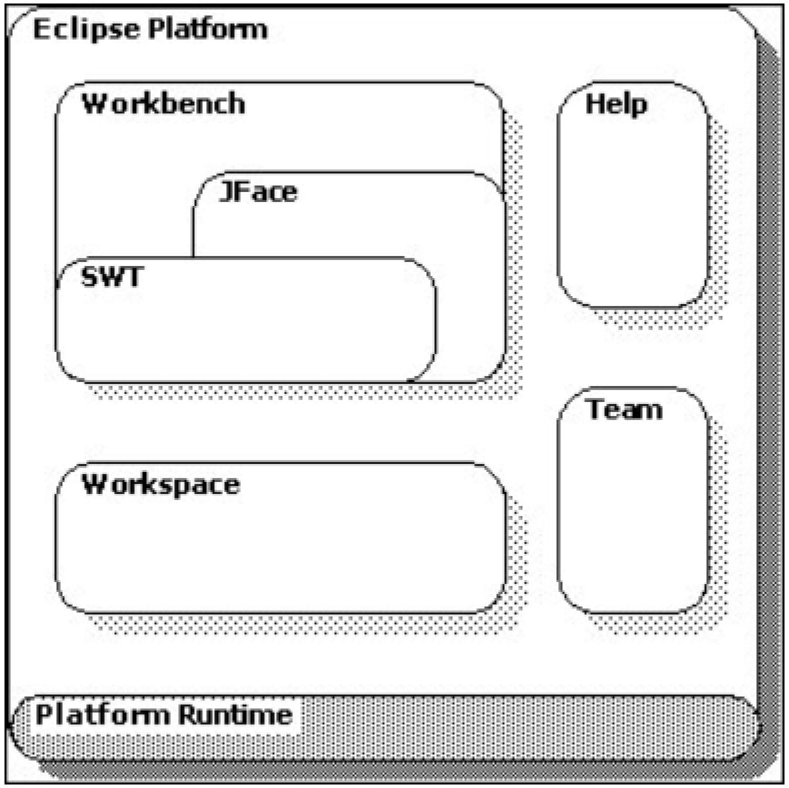
Introduction to Eclipse

Contents

Features

- Eclipse is an integrated development environment (IDE) used in computer programming, and is the most widely used Java IDE.
- It contains a base workspace and an extensible plug-in system for customizing the environment.
- Eclipse is written mostly in Java and its primary use is for developing Java applications.
- It may also be used to develop applications in other programming languages via plug-ins.

Eclipse platform

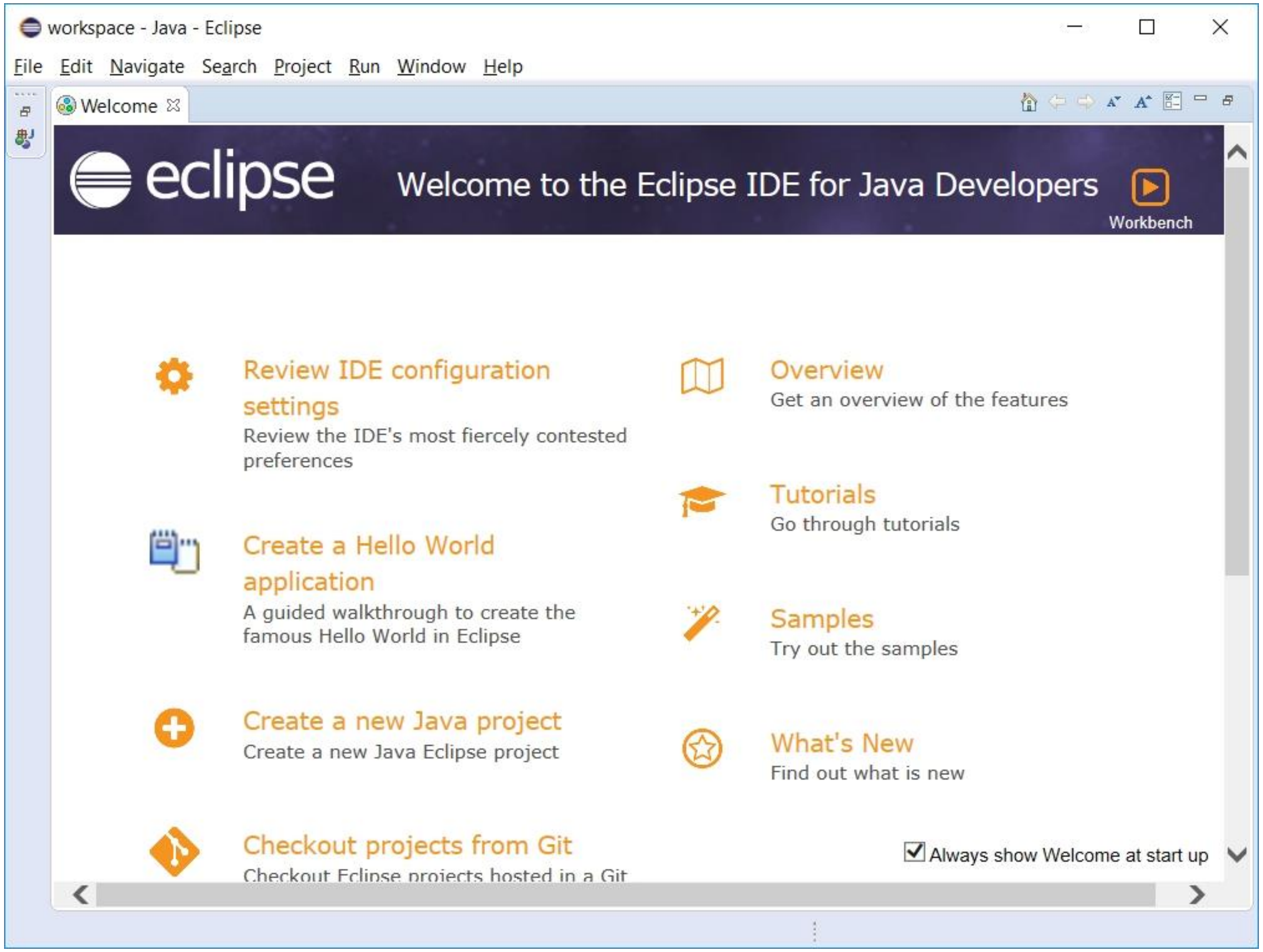


- The term Workbench refers to the desktop development environment. The Workbench aims to achieve seamless tool integration and controlled openness by providing a common paradigm for the creation, management, and navigation of workspace resources.

Eclipse platform

- Each Workbench window contains one or more perspectives.
- Perspectives contain views and editors and control what appears in certain menus and tool bars.
- More than one Workbench window can exist on the desktop at any given time.

Welcome window


workspace - Java - Eclipse

File Edit Navigate Search Project Run Window Help

Welcome

eclipse Welcome to the Eclipse IDE for Java Developers

Workbench

Review IDE configuration settings
Review the IDE's most fiercely contested preferences

Create a Hello World application
A guided walkthrough to create the famous Hello World in Eclipse

Create a new Java project
Create a new Java Eclipse project

Checkout projects from Git
Checkout Eclipse projects hosted in a Git

Overview
Get an overview of the features

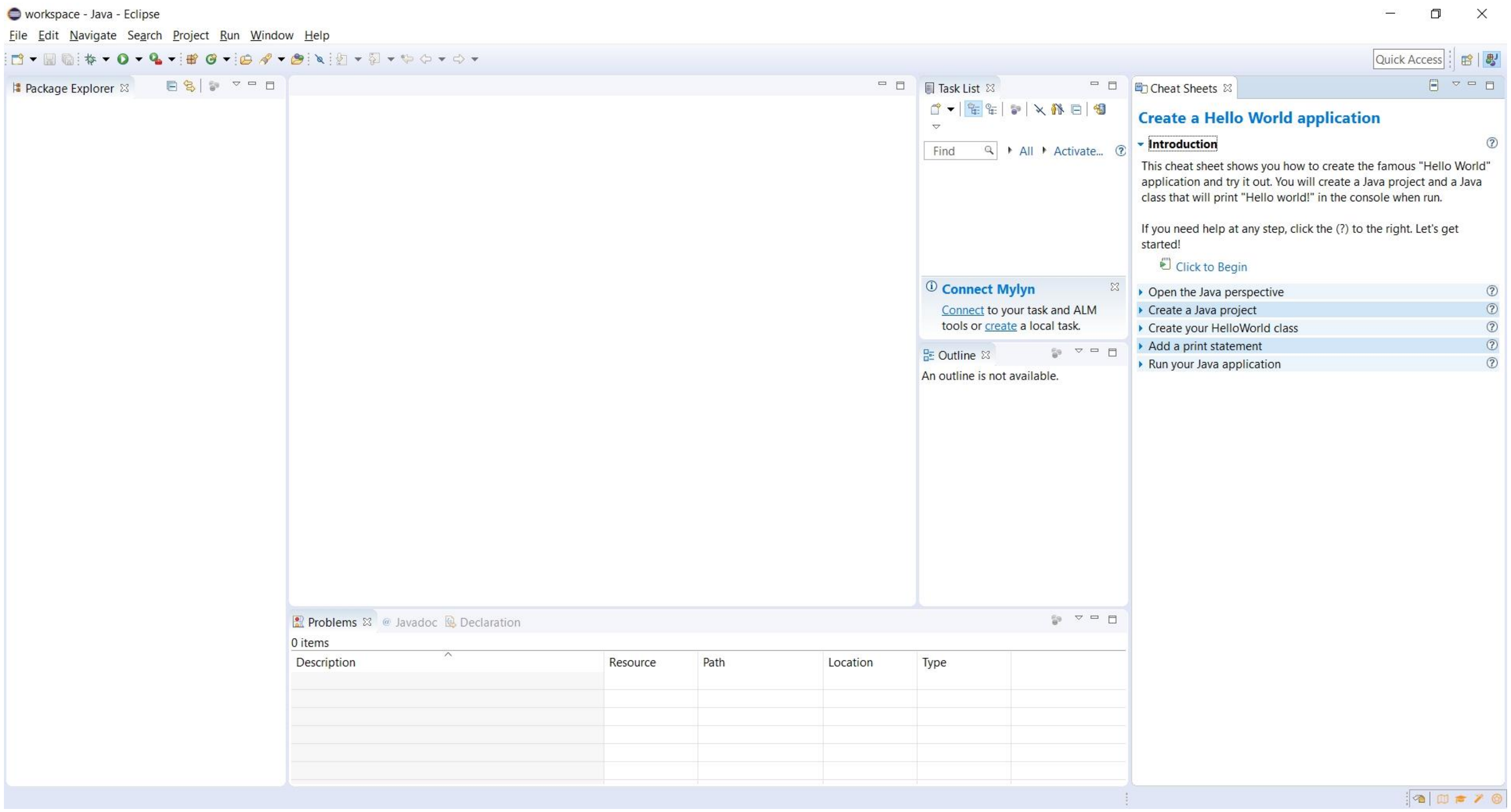
Tutorials
Go through tutorials

Samples
Try out the samples

What's New
Find out what is new

☑ Always show Welcome at start up

Hello World App



Create a Hello World application

Introduction

This cheat sheet shows you how to create the famous "Hello World" application and try it out. You will create a Java project and a Java class that will print "Hello world!" in the console when run.

If you need help at any step, click the (?) to the right. Let's get started!

Click to Begin

- Open the Java perspective
- Create a Java project
- Create your HelloWorld class
- Add a print statement
- Run your Java application

Connect Mylyn

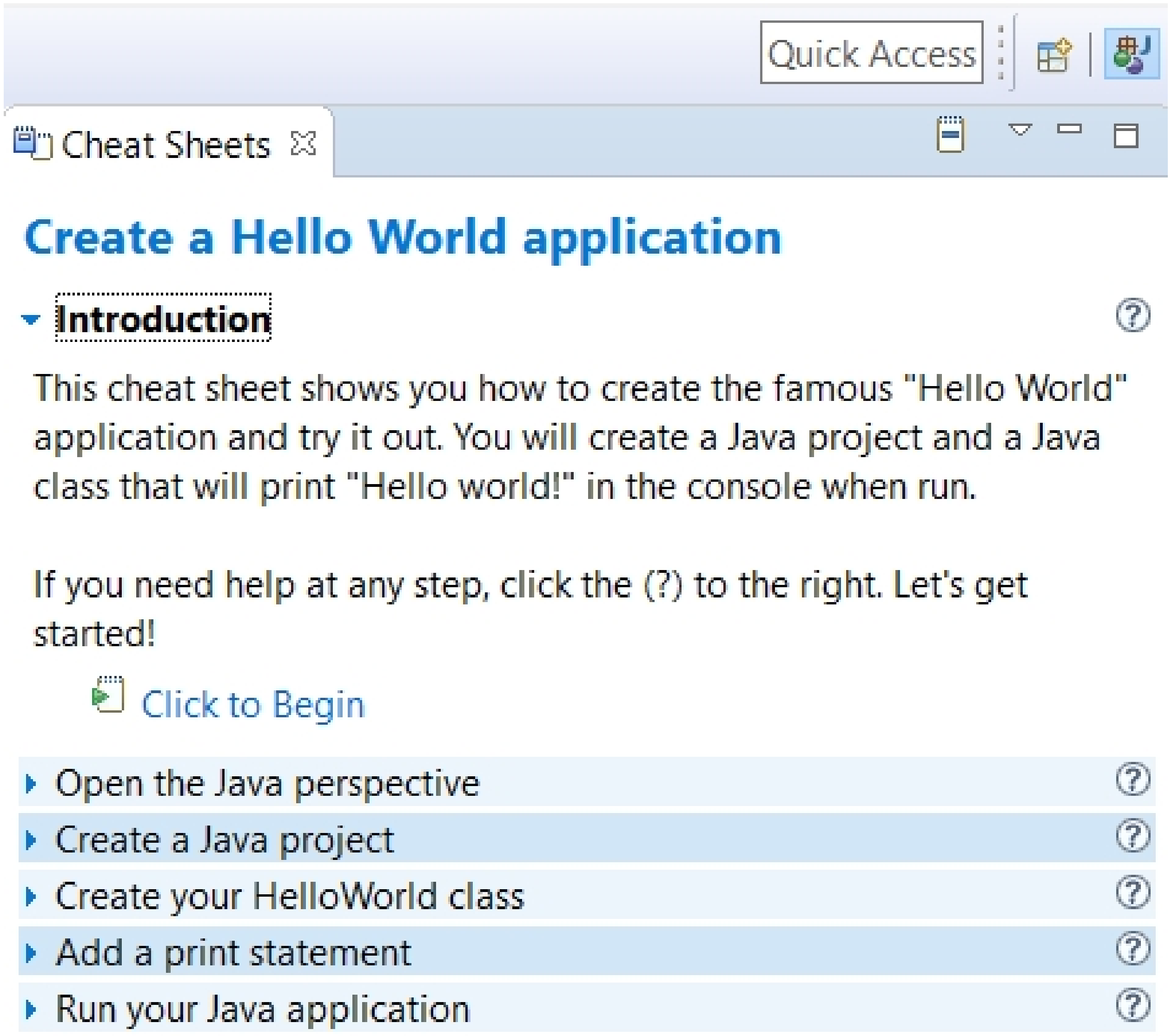
Connect to your task and ALM tools or create a local task.

An outline is not available.

0 items

Description	Resource	Path	Location	Type

Hello World App



Create a Hello World application

Introduction

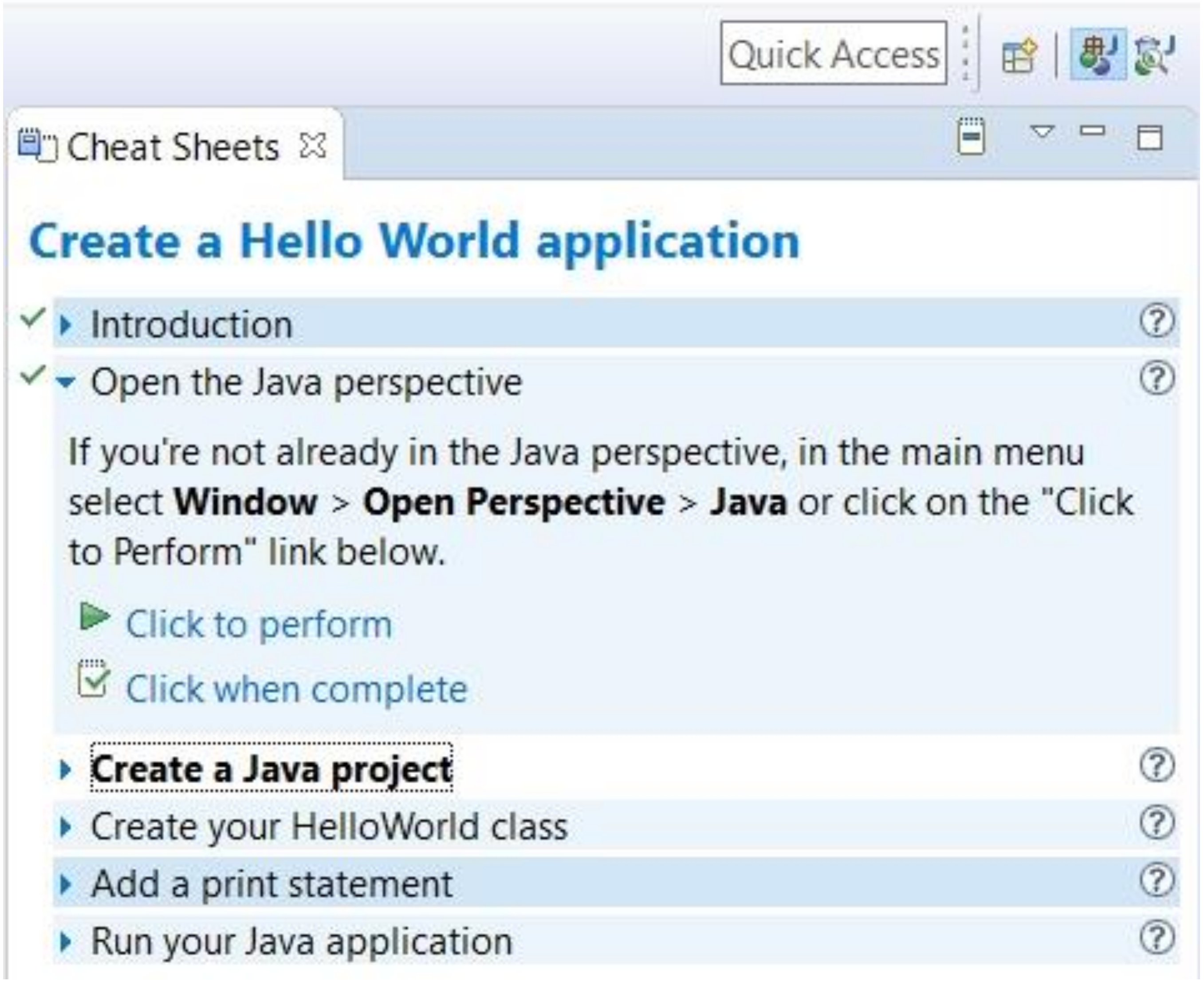
This cheat sheet shows you how to create the famous "Hello World" application and try it out. You will create a Java project and a Java class that will print "Hello world!" in the console when run.

If you need help at any step, click the (?) to the right. Let's get started!

Click to Begin

- Open the Java perspective
- Create a Java project
- Create your HelloWorld class
- Add a print statement
- Run your Java application

Hello World App



Create a Hello World application

✓ Introduction

✓ Open the Java perspective

If you're not already in the Java perspective, in the main menu select **Window** > **Open Perspective** > **Java** or click on the "Click to Perform" link below.

Click to perform

Click when complete

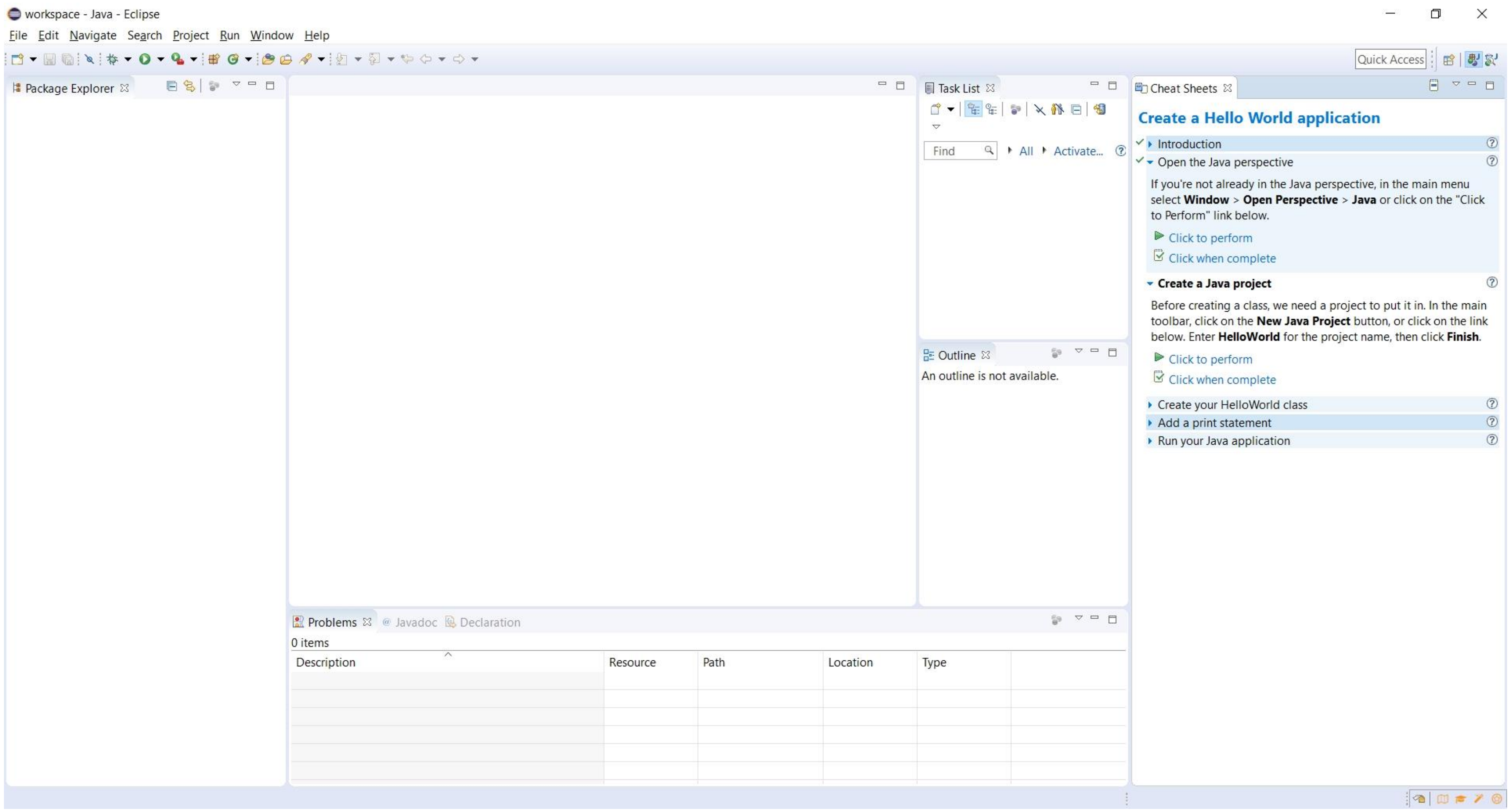
Create a Java project

Create your HelloWorld class

Add a print statement

Run your Java application

Hello World App



An outline is not available.

0 items

Description	Resource	Path	Location	Type

Create a Hello World application

- Introduction
- Open the Java perspective

If you're not already in the Java perspective, in the main menu select **Window** > **Open Perspective** > **Java** or click on the "Click to Perform" link below.

Click to perform

Click when complete

- **Create a Java project**

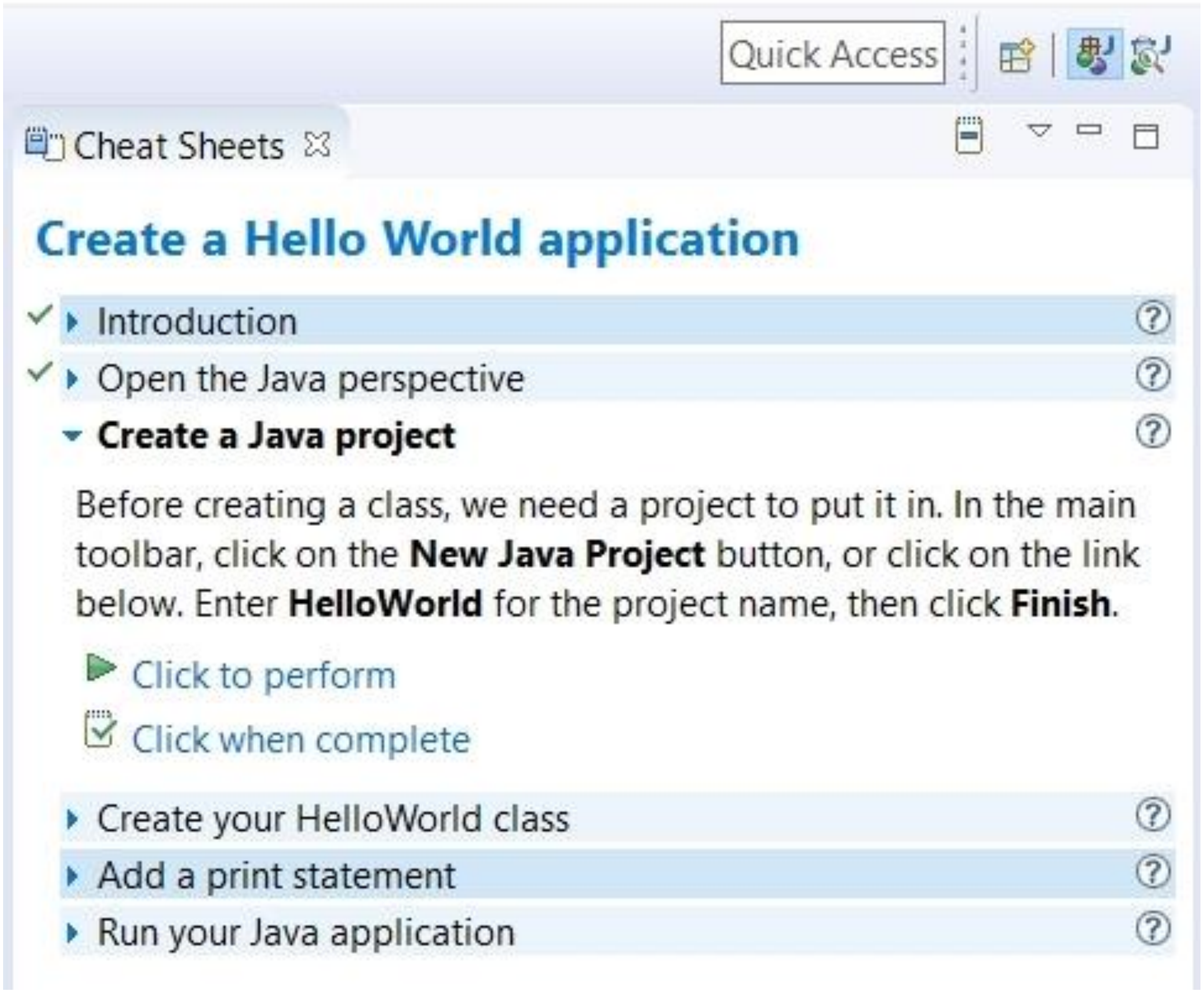
Before creating a class, we need a project to put it in. In the main toolbar, click on the **New Java Project** button, or click on the link below. Enter **HelloWorld** for the project name, then click **Finish**.

Click to perform

Click when complete

- Create your HelloWorld class
- Add a print statement
- Run your Java application

Hello World App



Create a Hello World application

- ✓ Introduction
- ✓ Open the Java perspective
- **Create a Java project**

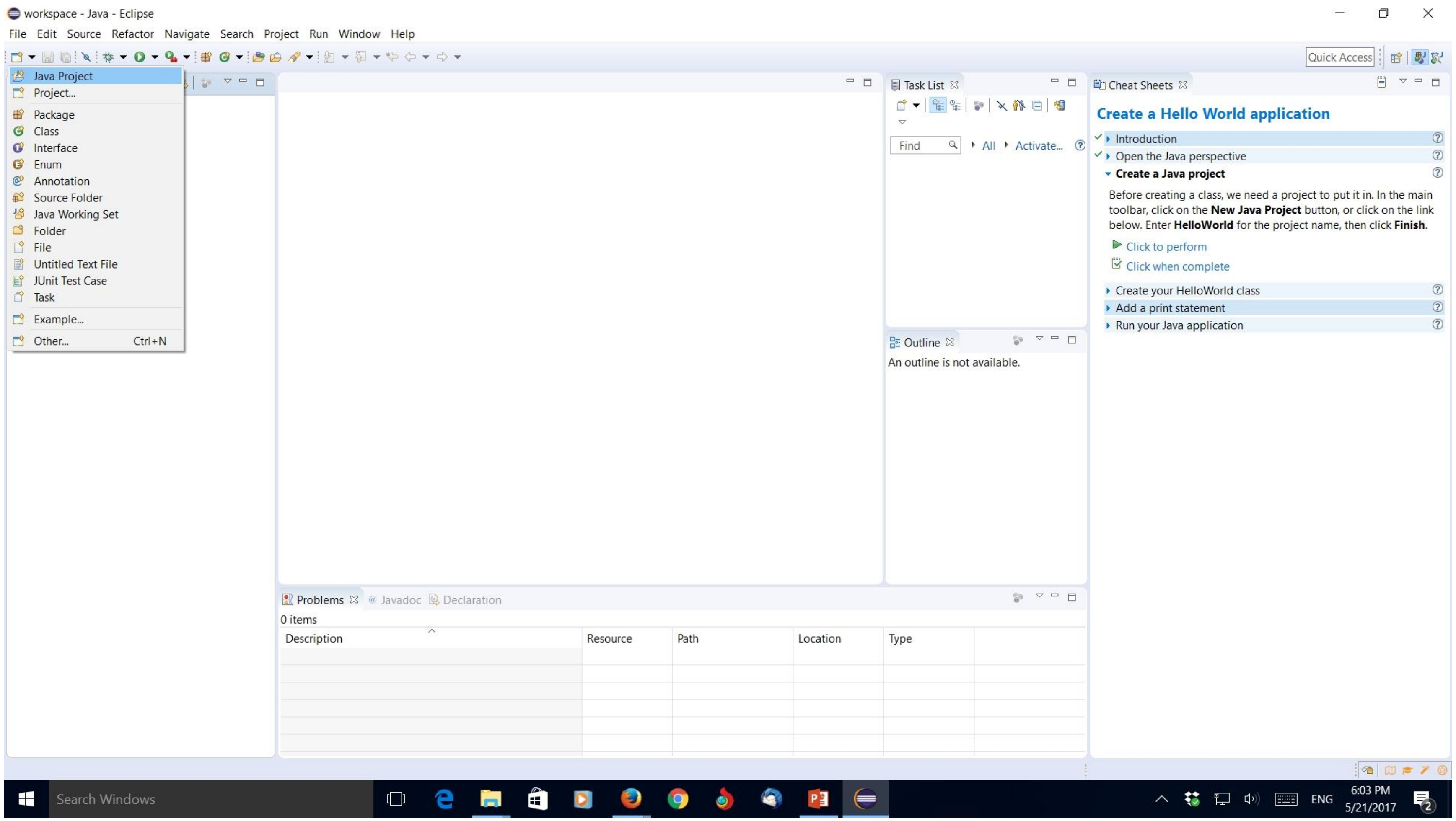
Before creating a class, we need a project to put it in. In the main toolbar, click on the **New Java Project** button, or click on the link below. Enter **HelloWorld** for the project name, then click **Finish**.

Click to perform

Click when complete

- Create your HelloWorld class
- Add a print statement
- Run your Java application

Hello World App



Create a Hello World application

- Introduction
- Open the Java perspective
- **Create a Java project**

Before creating a class, we need a project to put it in. In the main toolbar, click on the **New Java Project** button, or click on the link below. Enter **HelloWorld** for the project name, then click **Finish**.

Click to perform

Click when complete

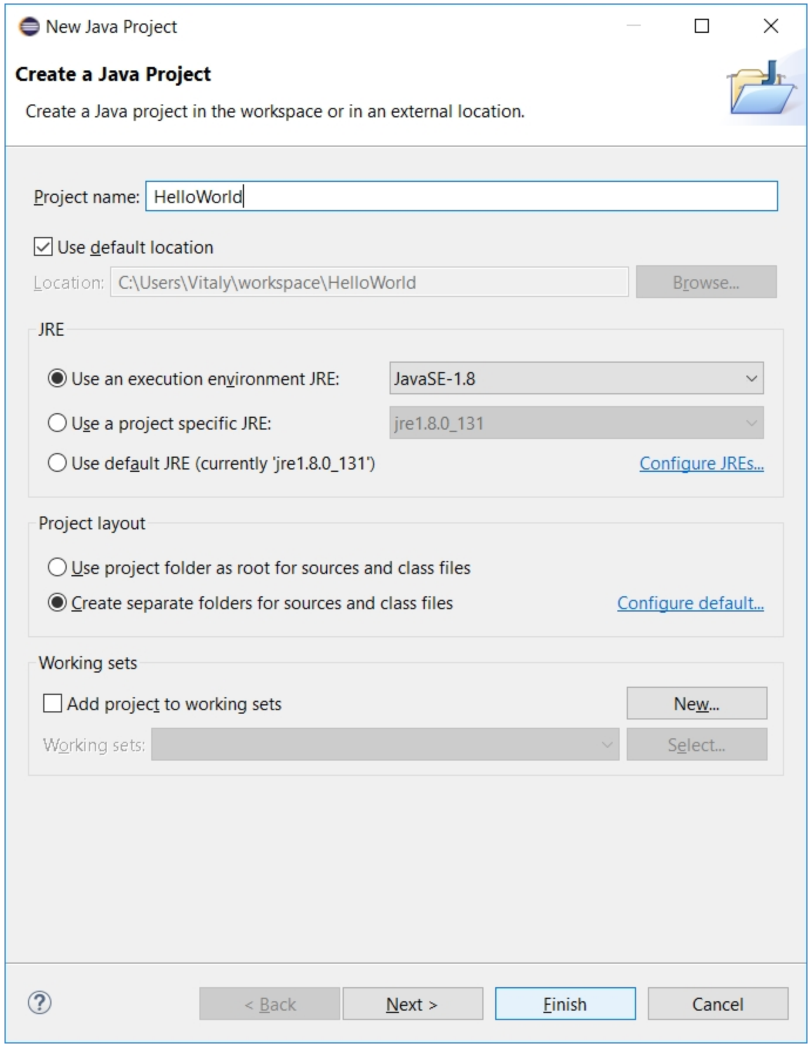
- Create your HelloWorld class
- Add a print statement
- Run your Java application

An outline is not available.

0 items

Description	Resource	Path	Location	Type

Hello World App



New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: HelloWorld

[✓] Use default location

Location: C:\Users\Vitaly\workspace\HelloWorld Browse...

JRE

● Use an execution environment JRE: JavaSE-1.8

○ Use a project specific JRE: jre1.8.0_131

○ Use default JRE (currently 'jre1.8.0_131') Configure JREs...

Project layout

○ Use project folder as root for sources and class files

● Create separate folders for sources and class files Configure default...

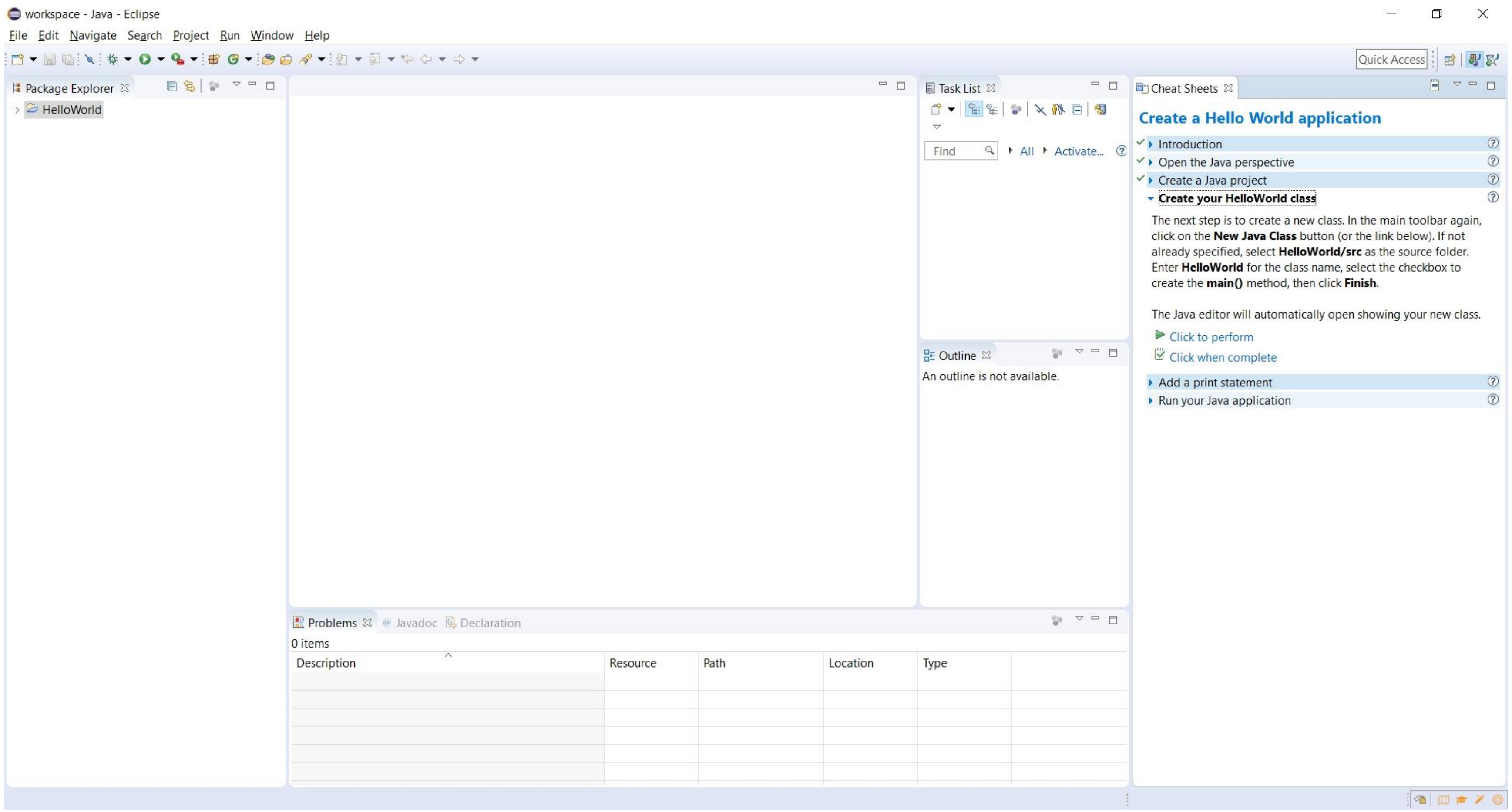
Working sets

☐ Add project to working sets New...

Working sets: Select...

< Back Next > Finish Cancel

Hello World App



An outline is not available.

0 items

Create a Hello World application

- Introduction
- Open the Java perspective
- Create a Java project
- Create your HelloWorld class

The next step is to create a new class. In the main toolbar again, click on the **New Java Class** button (or the link below). If not already specified, select **HelloWorld/src** as the source folder. Enter **HelloWorld** for the class name, select the checkbox to create the **main()** method, then click **Finish**.

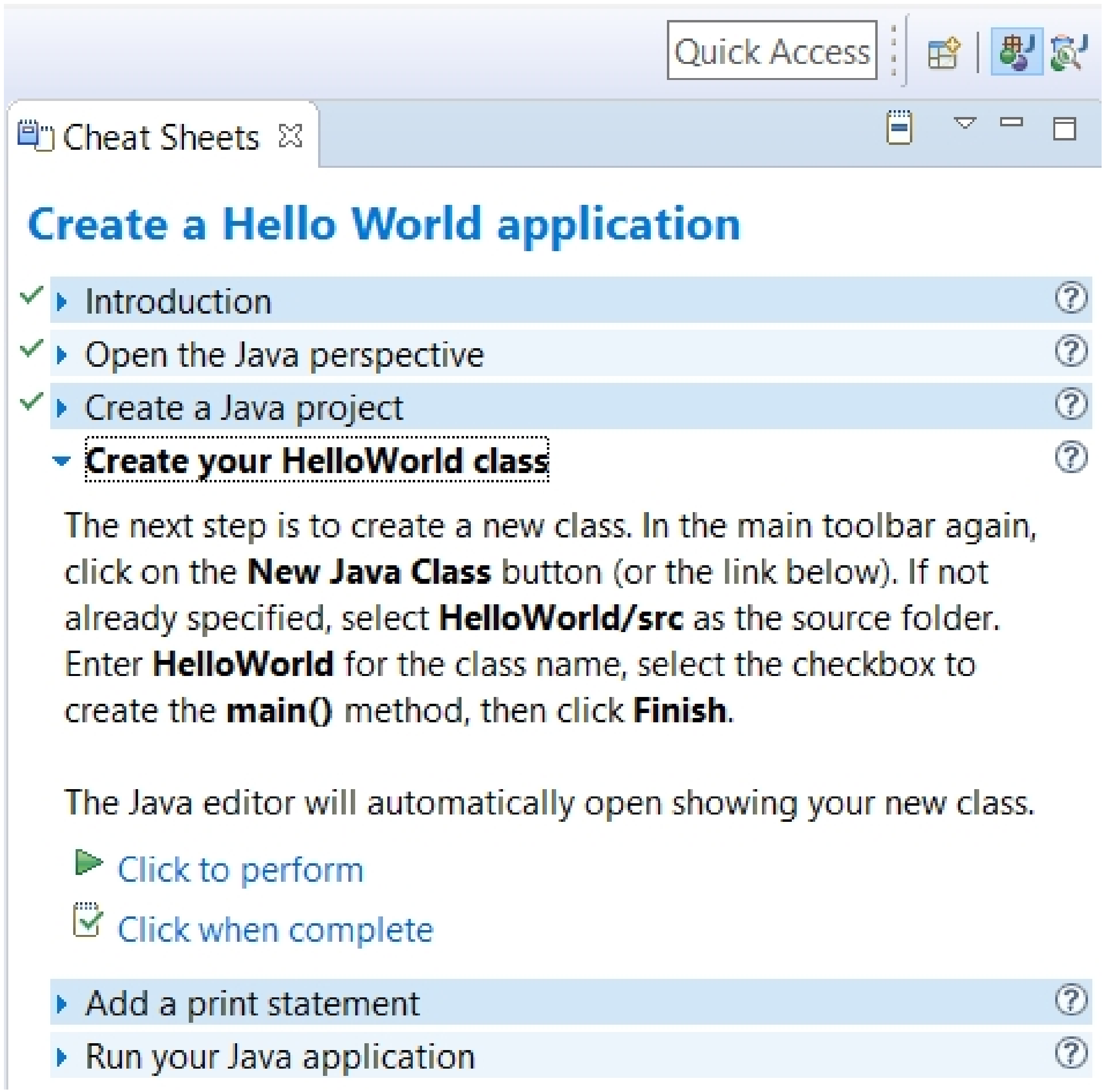
The Java editor will automatically open showing your new class.

Click to perform

Click when complete

- Add a print statement
- Run your Java application

Hello World App



Create a Hello World application

- ✓ Introduction
- ✓ Open the Java perspective
- ✓ Create a Java project
- **Create your HelloWorld class**

The next step is to create a new class. In the main toolbar again, click on the **New Java Class** button (or the link below). If not already specified, select **HelloWorld/src** as the source folder. Enter **HelloWorld** for the class name, select the checkbox to create the **main()** method, then click **Finish**.

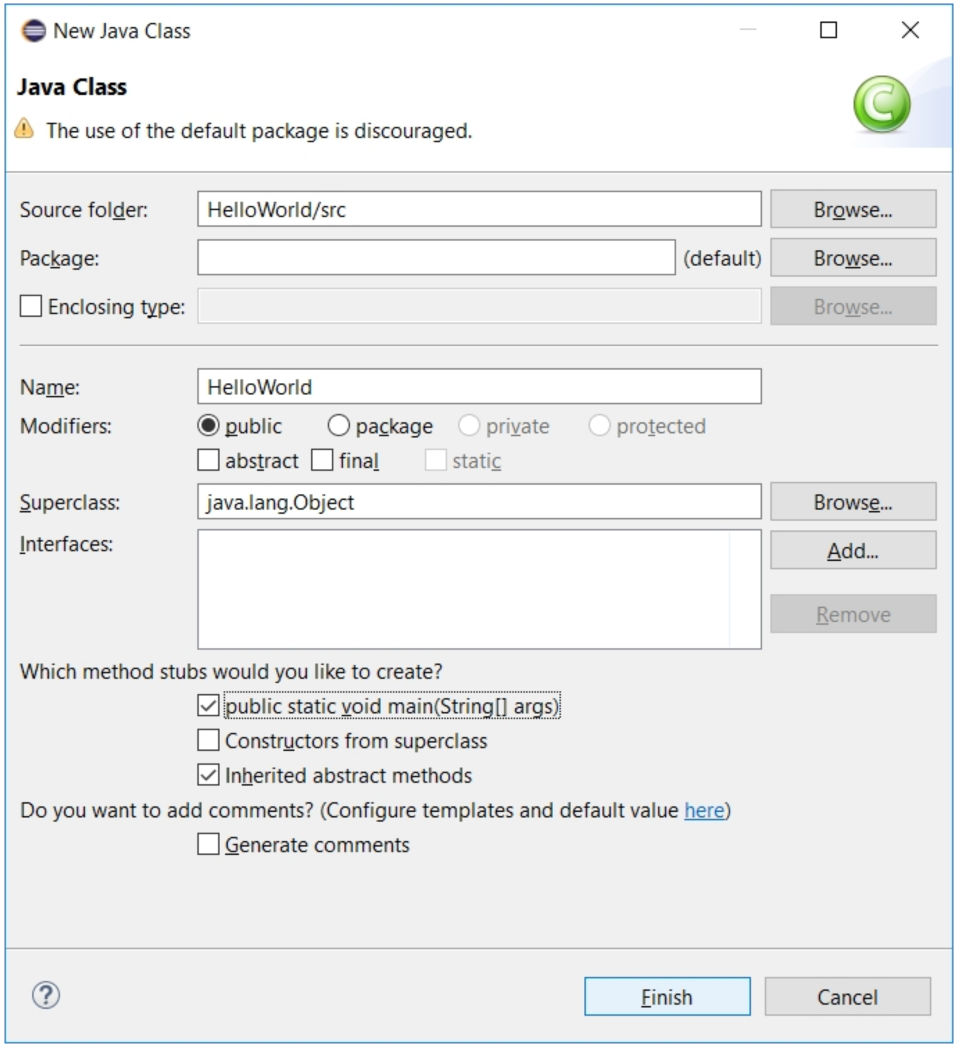
The Java editor will automatically open showing your new class.

Click to perform

Click when complete

- Add a print statement
- Run your Java application

Hello World App


New Java Class

Java Class

⚠ The use of the default package is discouraged.

Source folder: HelloWorld/src

Package: (default)

☐ Enclosing type:

Name: HelloWorld

Modifiers: [✓ public] package private protected

☐ abstract ☐ final ☐ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

☑ public static void main(String[] args)

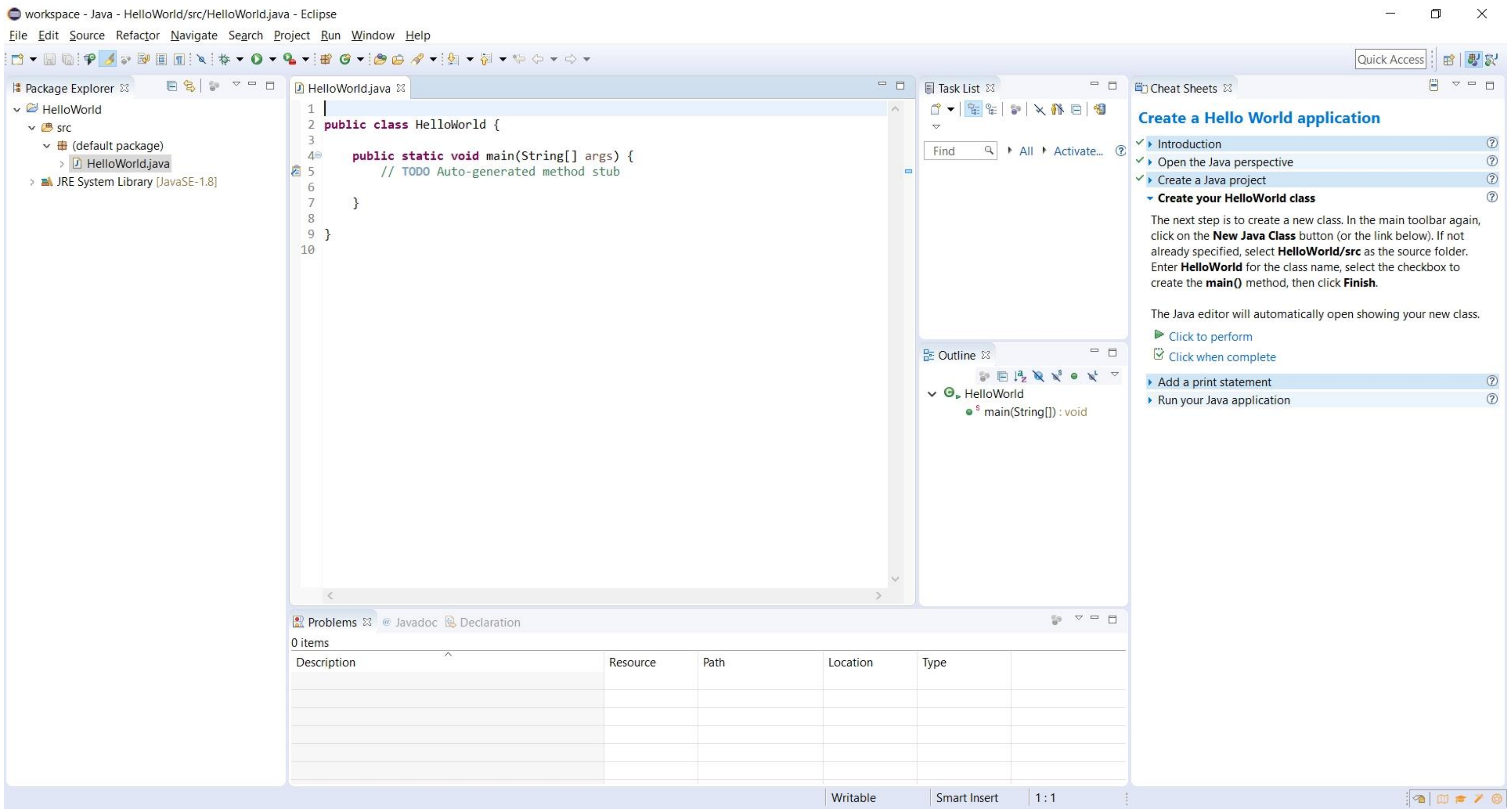
☐ Constructors from superclass

☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

☐ Generate comments

Hello World App



```

public class HelloWorld {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}

```

Create a Hello World application

- ✓ Introduction
- ✓ Open the Java perspective
- ✓ Create a Java project
- **Create your HelloWorld class**

The next step is to create a new class. In the main toolbar again, click on the **New Java Class** button (or the link below). If not already specified, select **HelloWorld/src** as the source folder. Enter **HelloWorld** for the class name, select the checkbox to create the **main()** method, then click **Finish**.

The Java editor will automatically open showing your new class.

Click to perform

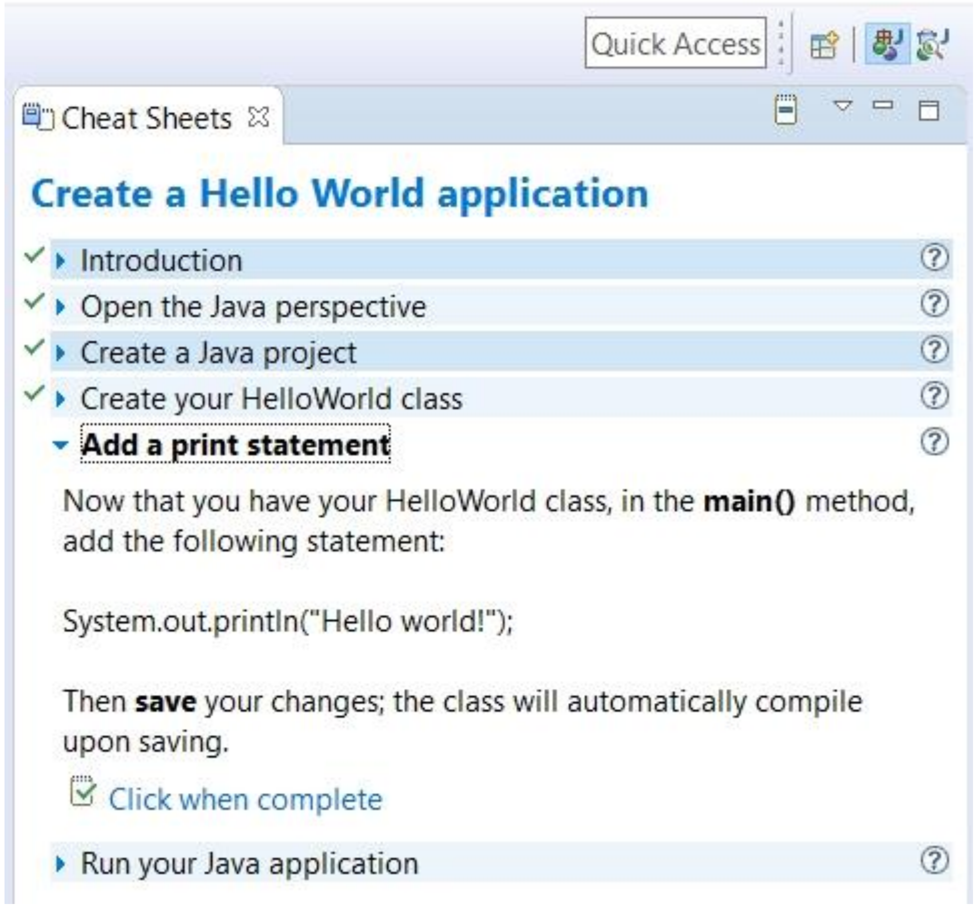
Click when complete

- Add a print statement
- Run your Java application

0 items

Description	Resource	Path	Location	Type

Hello World App



Create a Hello World application

- ✓ Introduction
- ✓ Open the Java perspective
- ✓ Create a Java project
- ✓ Create your HelloWorld class
- **Add a print statement**

Now that you have your HelloWorld class, in the **main()** method, add the following statement:

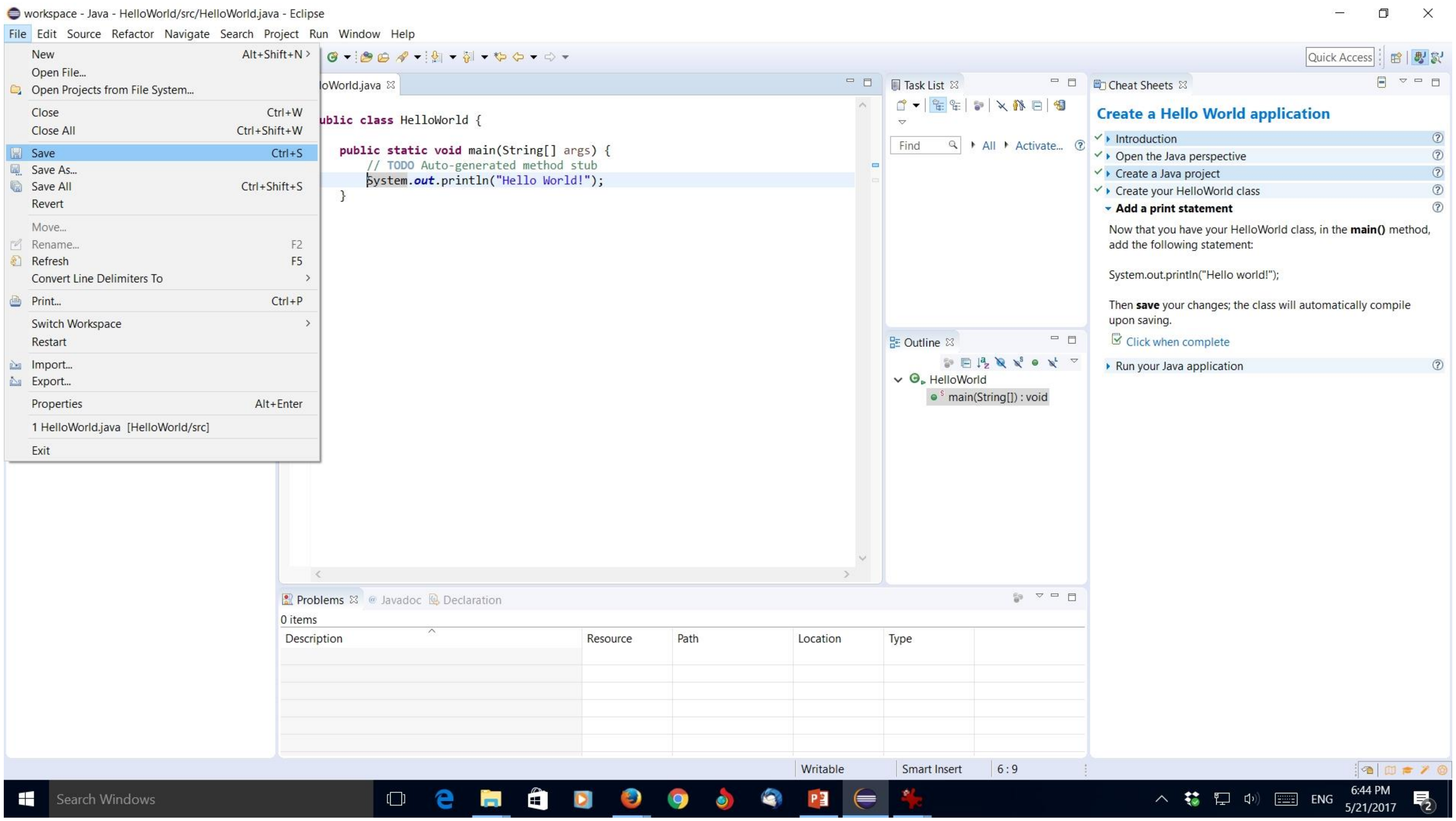
```
System.out.println("Hello world!");
```

Then **save** your changes; the class will automatically compile upon saving.

Click when complete

- Run your Java application

Hello World App



```
ublic class HelloWorld {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Hello World!");
    }
```

Create a Hello World application

- Introduction
- Open the Java perspective
- Create a Java project
- Create your HelloWorld class
- **Add a print statement**

Now that you have your HelloWorld class, in the **main()** method, add the following statement:

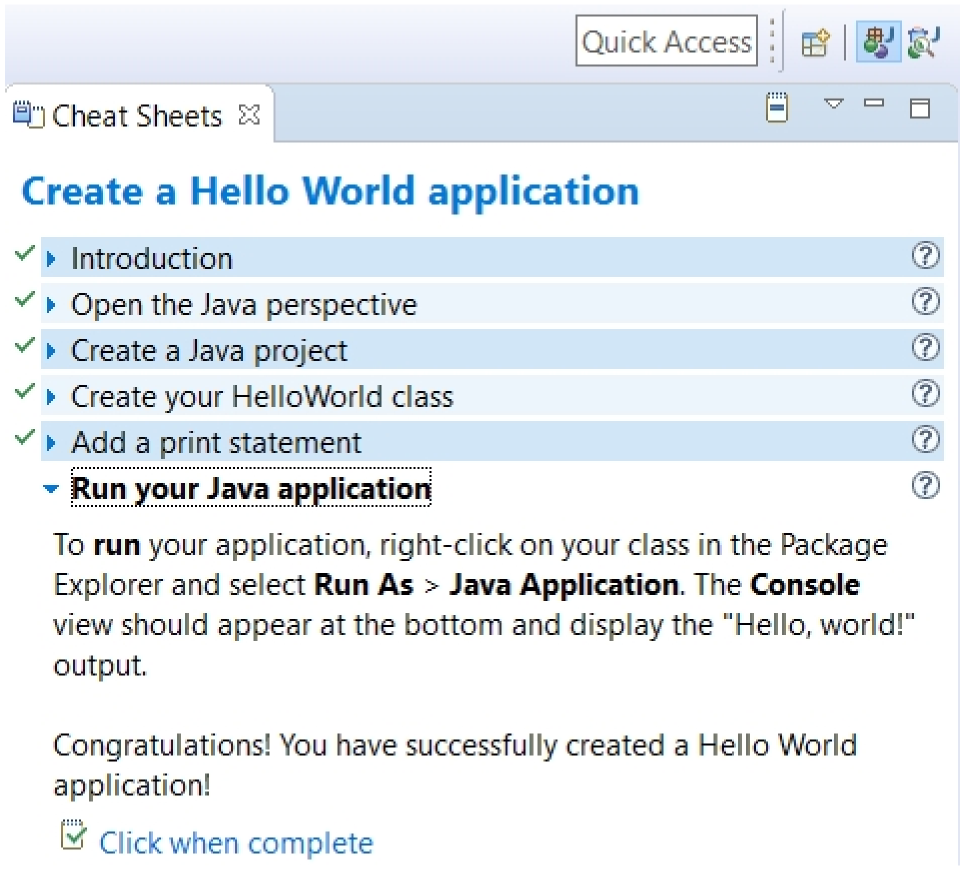
System.out.println("Hello world!");

Then **save** your changes; the class will automatically compile upon saving.

Click when complete

- Run your Java application

Hello World App



Create a Hello World application

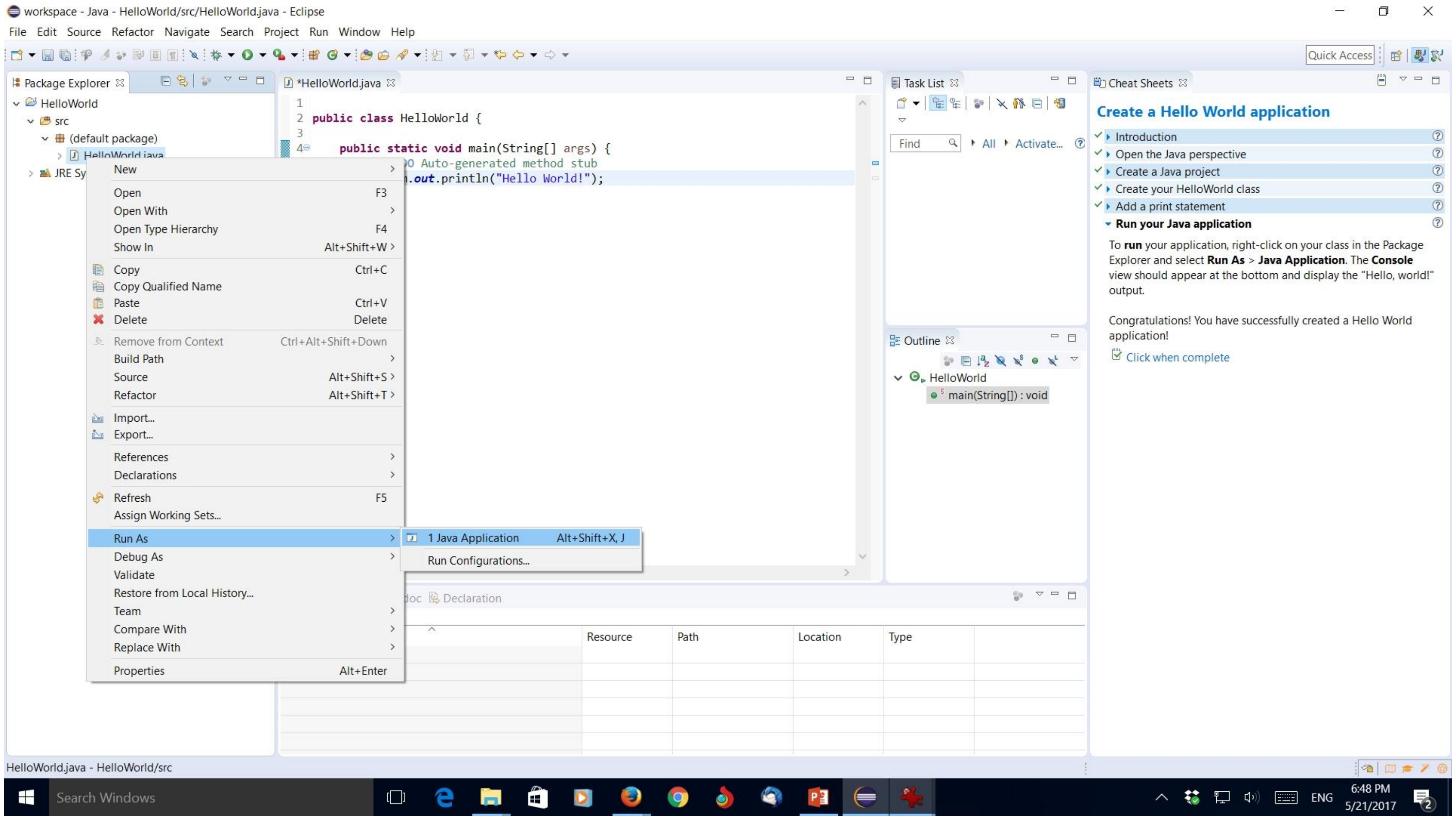
- ✓ Introduction
- ✓ Open the Java perspective
- ✓ Create a Java project
- ✓ Create your HelloWorld class
- ✓ Add a print statement
- **Run your Java application**

To **run** your application, right-click on your class in the Package Explorer and select **Run As** > **Java Application**. The **Console** view should appear at the bottom and display the "Hello, world!" output.

Congratulations! You have successfully created a Hello World application!

Click when complete

Hello World App



```
public class HelloWorld {

    public static void main(String[] args) {
        O Auto-generated method stub
        n.out.println("Hello World!");
```

Create a Hello World application

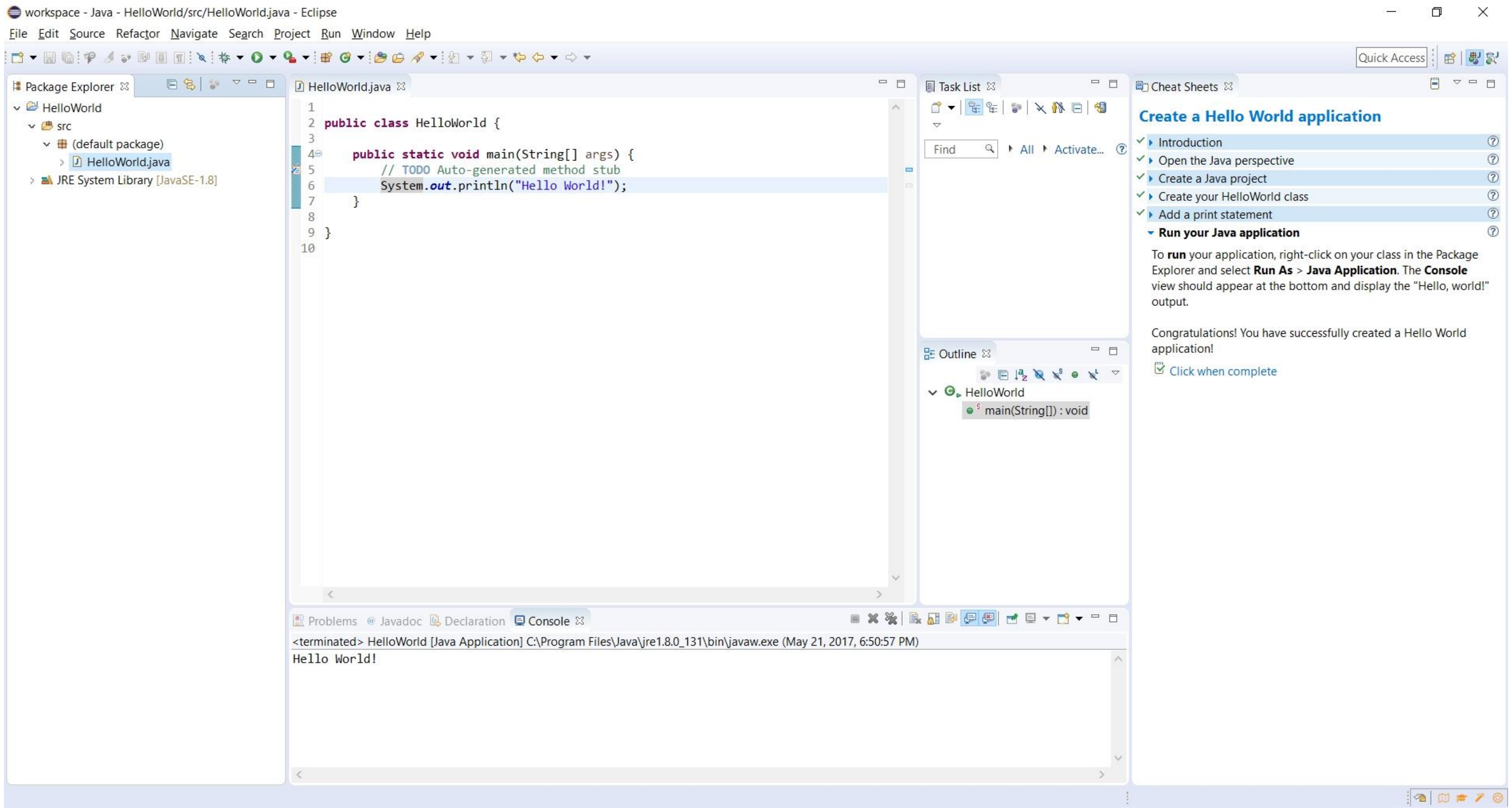
- Introduction
- Open the Java perspective
- Create a Java project
- Create your HelloWorld class
- Add a print statement
- **Run your Java application**

To **run** your application, right-click on your class in the Package Explorer and select **Run As** > **Java Application**. The **Console** view should appear at the bottom and display the "Hello, world!" output.

Congratulations! You have successfully created a Hello World application!

Click when complete

Hello World App



```
public class HelloWorld {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Hello World!");
    }

}
```

```
<terminated> HelloWorld [Java Application] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe (May 21, 2017, 6:50:57 PM)
Hello World!
```

Create a Hello World application

- Introduction
- Open the Java perspective
- Create a Java project
- Create your HelloWorld class
- Add a print statement
- **Run your Java application**

To **run** your application, right-click on your class in the Package Explorer and select **Run As** > **Java Application**. The **Console** view should appear at the bottom and display the "Hello, world!" output.

Congratulations! You have successfully created a Hello World application!

Click when complete

Summary

- Start Eclipse.
- Create a new Java Project:
  - File->New->Java Project.
  - Enter a project name into the Project name field, for example, "Hello World Project".
  - Click "Finish"

Summary

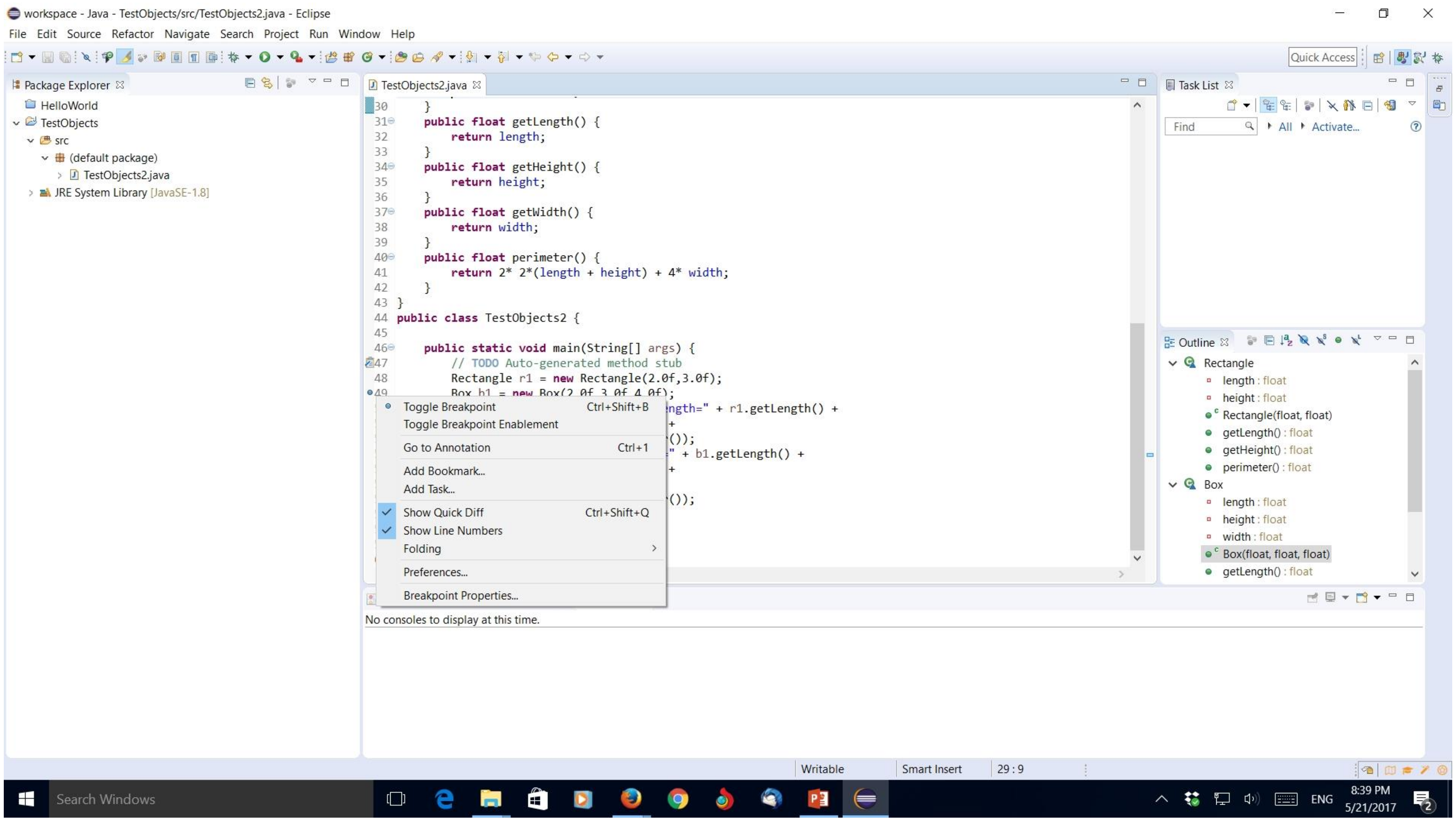
- Create a new Java class:
  - File->New->Class
  - Enter "HelloWorld" into the Name field.
  - Click the checkbox indicating that you would like Eclipse to create a "public static void main(String[] args)" method.
  - Click "Finish".
  - A Java editor for HelloWorld.java will open. In the main method enter the following line:

    System.out.println("Hello World!");
  - Save using ctrl-s. This automatically compiles HelloWorld.java.

Summary

- Right Click on the HelloWorld name class.
- Select "Run as“ -> “Java Application”
- You will be prompted to create a Launch configuration. Select "Java Application“.
  - The console will open and display "Hello World!".

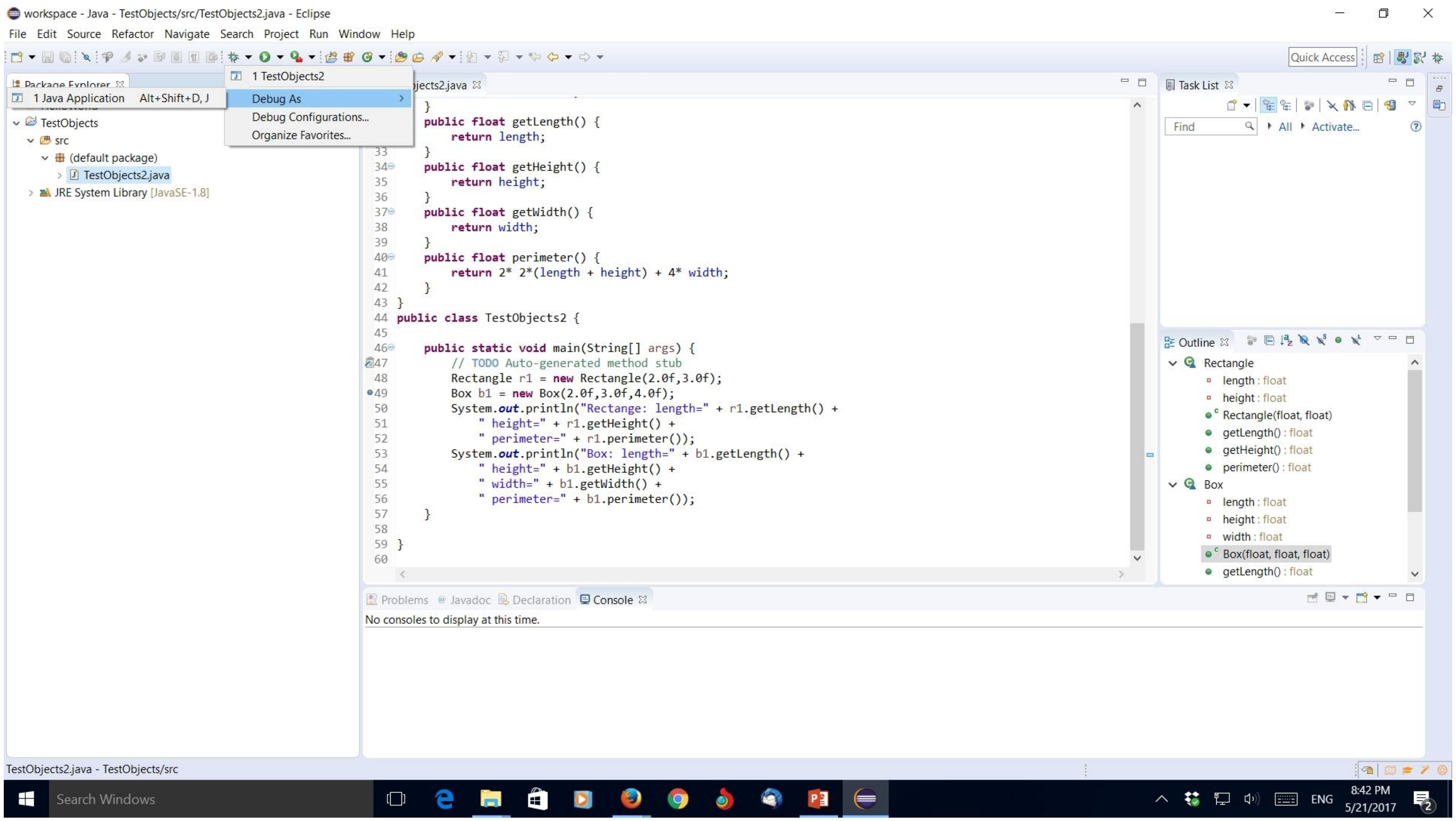
Debugging



```java
    }
    public float getLength() {
        return length;
    }
    public float getHeight() {
        return height;
    }
    public float getWidth() {
        return width;
    }
    public float perimeter() {
        return 2* 2*(length + height) + 4* width;
    }
}
public class TestObjects2 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Rectangle r1 = new Rectangle(2.0f,3.0f);
        Box b1 = new Box(2.0f,3.0f,4.0f);
                                   ngth=" + r1.getLength() +
                                   +
                                   ());
                                   " + b1.getLength() +
                                   +
                                   ());
```

No consoles to display at this time.

Debugging

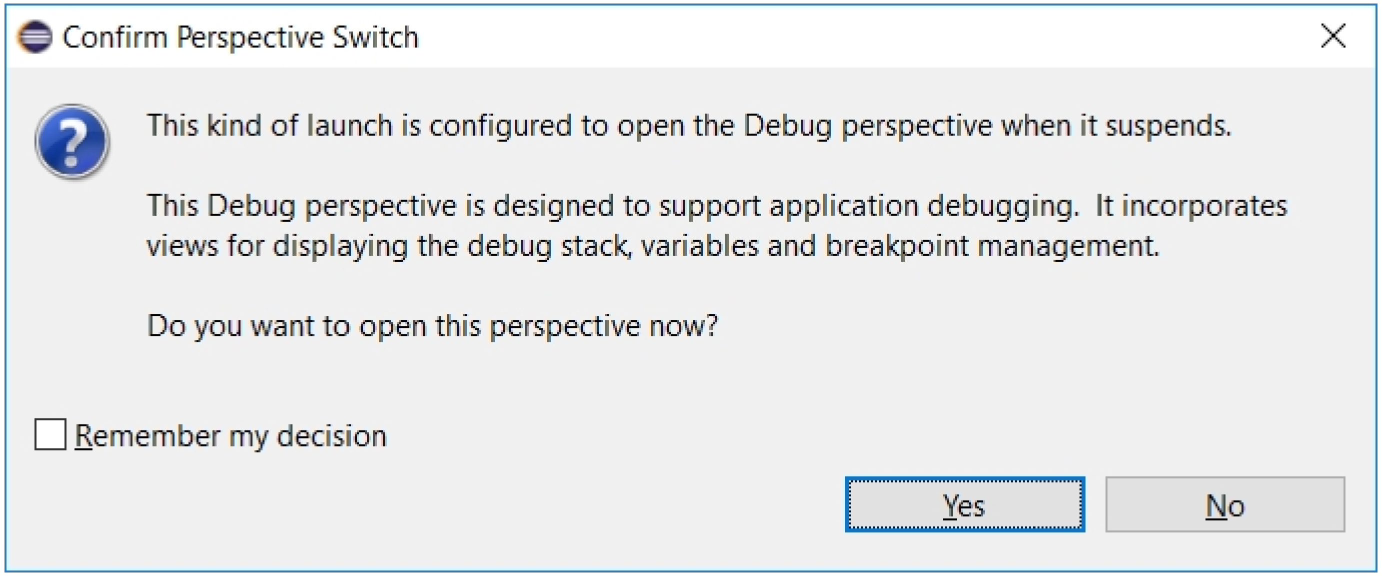


```
    }
    public float getLength() {
        return length;
    }
    public float getHeight() {
        return height;
    }
    public float getWidth() {
        return width;
    }
    public float perimeter() {
        return 2* 2*(length + height) + 4* width;
    }
}
public class TestObjects2 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Rectangle r1 = new Rectangle(2.0f,3.0f);
        Box b1 = new Box(2.0f,3.0f,4.0f);
        System.out.println("Rectange: length=" + r1.getLength() +
            " height=" + r1.getHeight() +
            " perimeter=" + r1.perimeter());
        System.out.println("Box: length=" + b1.getLength() +
            " height=" + b1.getHeight() +
            " width=" + b1.getWidth() +
            " perimeter=" + b1.perimeter());
    }

}
```

```
No consoles to display at this time.
```

TestObjects2.java - TestObjects/src

Debugging

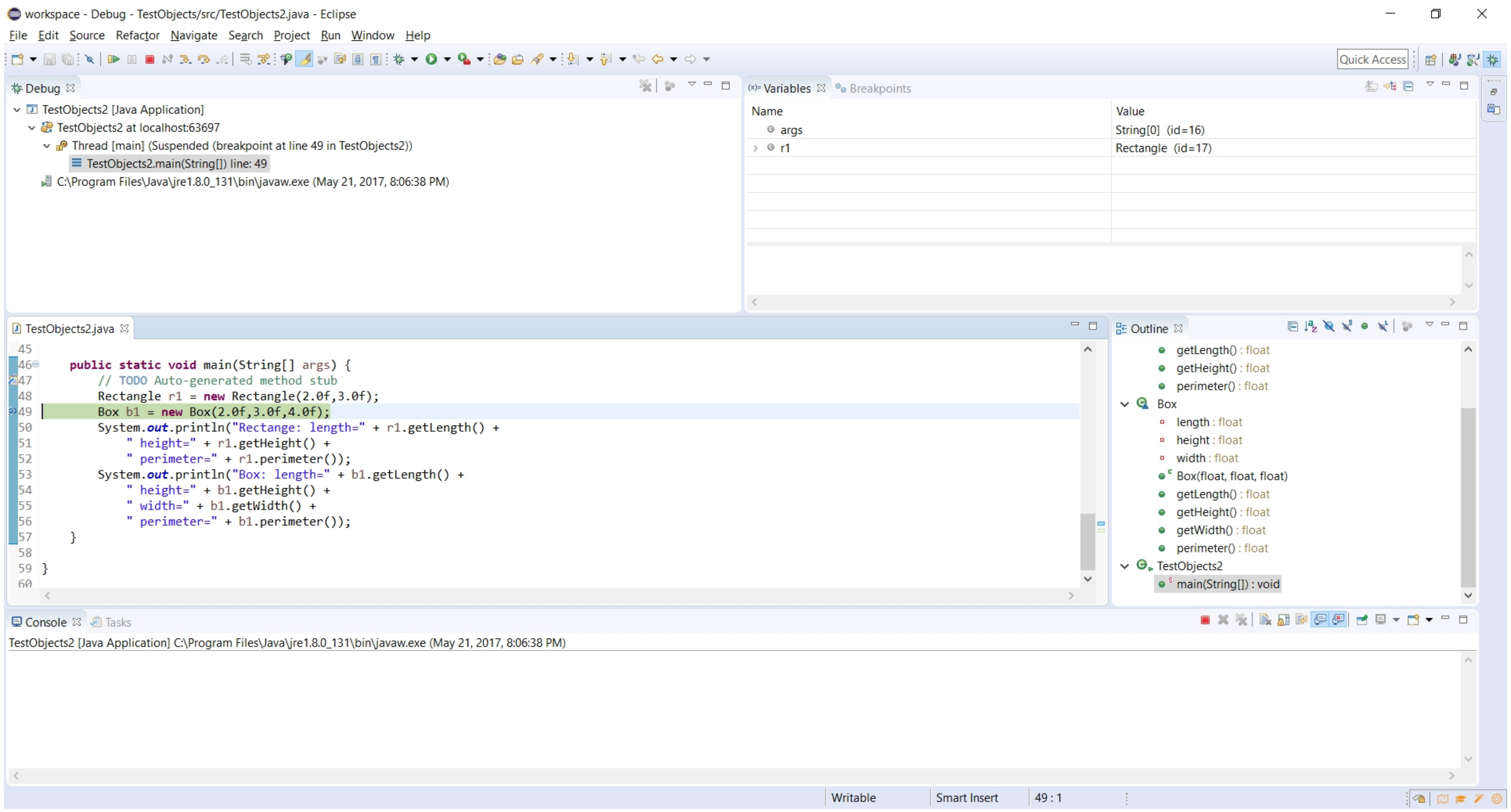

Confirm Perspective Switch

This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

Do you want to open this perspective now?

Debugging


workspace - Debug - TestObjects/src/TestObjects2.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Debug
Variables
Breakpoints
TestObjects2.java
Outline
Console
Tasks
Writable
Smart Insert
49 : 1

```
TestObjects2 [Java Application]
  TestObjects2 at localhost:63697
    Thread [main] (Suspended (breakpoint at line 49 in TestObjects2))
      TestObjects2.main(String[]) line: 49
  C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe (May 21, 2017, 8:06:38 PM)
```

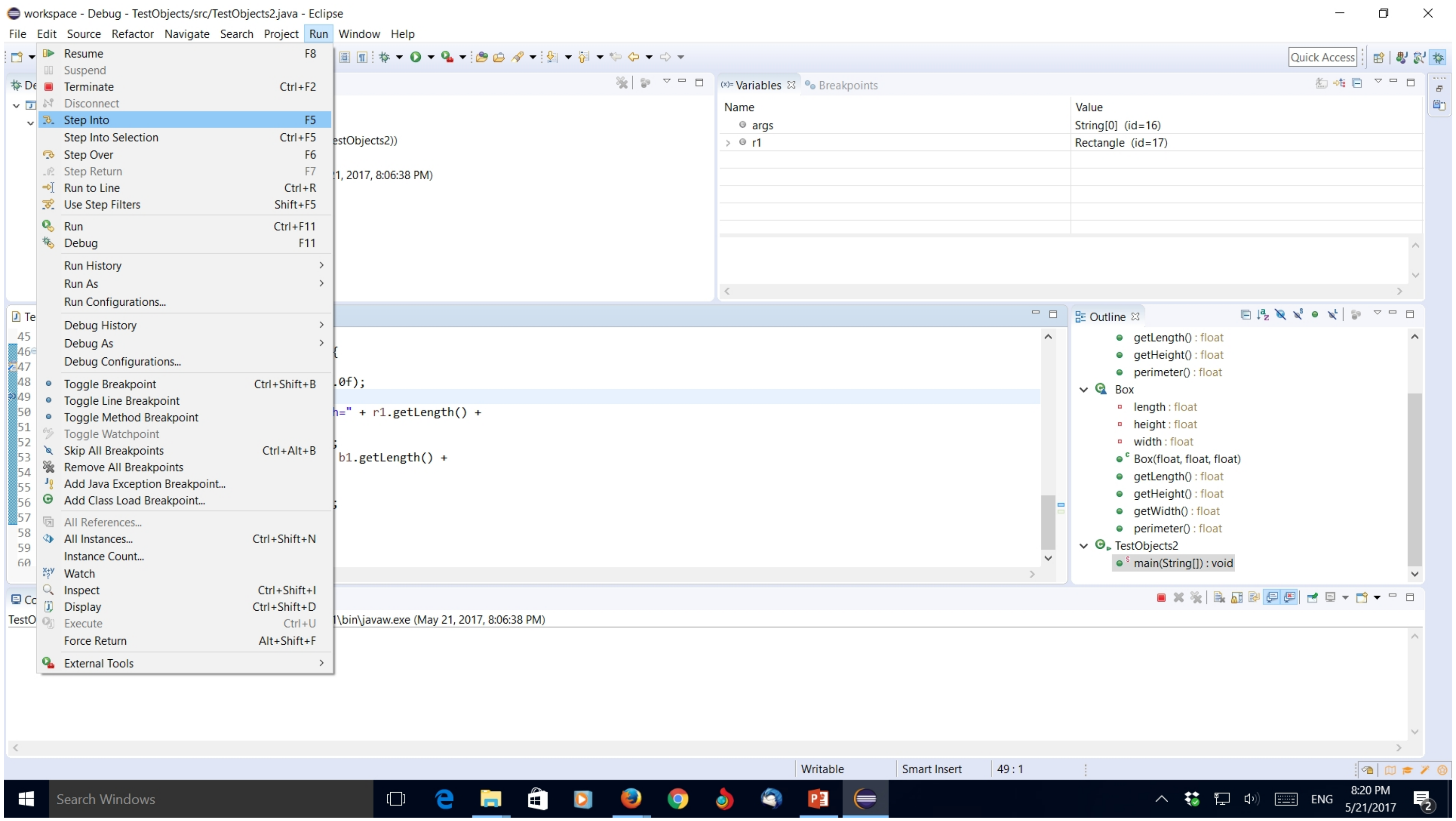
Name	Value
args	String[0] (id=16)
r1	Rectangle (id=17)

```java

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Rectangle r1 = new Rectangle(2.0f,3.0f);
        Box b1 = new Box(2.0f,3.0f,4.0f);
        System.out.println("Rectange: length=" + r1.getLength() +
            " height=" + r1.getHeight() +
            " perimeter=" + r1.perimeter());
        System.out.println("Box: length=" + b1.getLength() +
            " height=" + b1.getHeight() +
            " width=" + b1.getWidth() +
            " perimeter=" + b1.perimeter());
    }

}
```

```
    getLength() : float
    getHeight() : float
    perimeter() : float
  Box
    length : float
    height : float
    width : float
    Box(float, float, float)
    getLength() : float
    getHeight() : float
    getWidth() : float
    perimeter() : float
  TestObjects2
    main(String[]) : void
```

```
TestObjects2 [Java Application] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe (May 21, 2017, 8:06:38 PM)
```

Debugging



Name	Value
args	String[0] (id=16)
r1	Rectangle (id=17)

```
estObjects2))
1, 2017, 8:06:38 PM)
{
.0f);
h=" + r1.getLength() +
;
 b1.getLength() +
;
```

Outline:

```
getLength() : float
getHeight() : float
perimeter() : float
Box
    length : float
    height : float
    width : float
    Box(float, float, float)
    getLength() : float
    getHeight() : float
    getWidth() : float
    perimeter() : float
TestObjects2
    main(String[]) : void
```

```
TestO    \bin\javaw.exe (May 21, 2017, 8:06:38 PM)
```

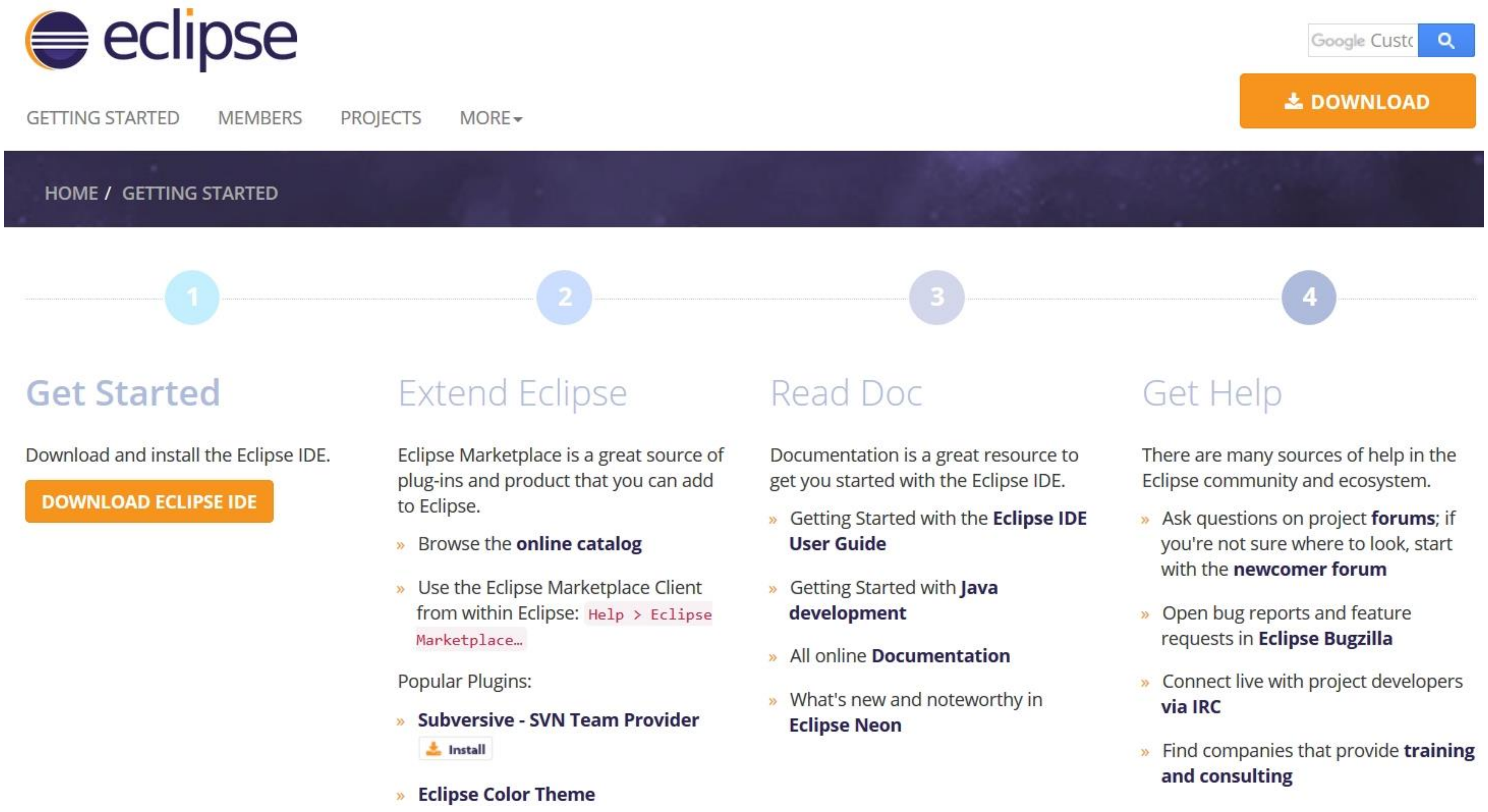
Comments on the previous slide

- Debug: shows the live stack trace of methods. We are currently only in the main method of out program. If other programs were currently running, we would see them here. It also has some stepping buttons that we will cover shortly.
- Variables: shows the values of all the variables that have been declared.
- Breakpoints: this view shows the location of all our breakpoints in the code (we can have many different breakpoints).
- TestObject2.java: we can view where are in our code at the time of the breakpoint

Comments on the previous slide

- To create a *breakpoint*, right-click along the left side of the Java editor on the line of code you wish to break on, (Also, double-clicking on the left side of the editor will create a breakpoint.)
- Step into command: to step into the next method call at the currently executing line of code.
- Step over command: to step over the next method call (without entering it) at the currently executing line of code. Even though the method is never stepped into, the method will be executed normally.
  - To step into / step over a method you must have execution suspended and be stepping through code.

https://www.eclipse.org/users/



Get Started

Download and install the Eclipse IDE.

DOWNLOAD ECLIPSE IDE

Extend Eclipse

Eclipse Marketplace is a great source of plug-ins and product that you can add to Eclipse.

» Browse the **online catalog**

» Use the Eclipse Marketplace Client from within Eclipse: `Help > Eclipse Marketplace…`

Popular Plugins:

» **Subversive - SVN Team Provider**
Install

» **Eclipse Color Theme**

Read Doc

Documentation is a great resource to get you started with the Eclipse IDE.

» Getting Started with the **Eclipse IDE User Guide**

» Getting Started with **Java development**

» All online **Documentation**

» What's new and noteworthy in **Eclipse Neon**

Get Help

There are many sources of help in the Eclipse community and ecosystem.

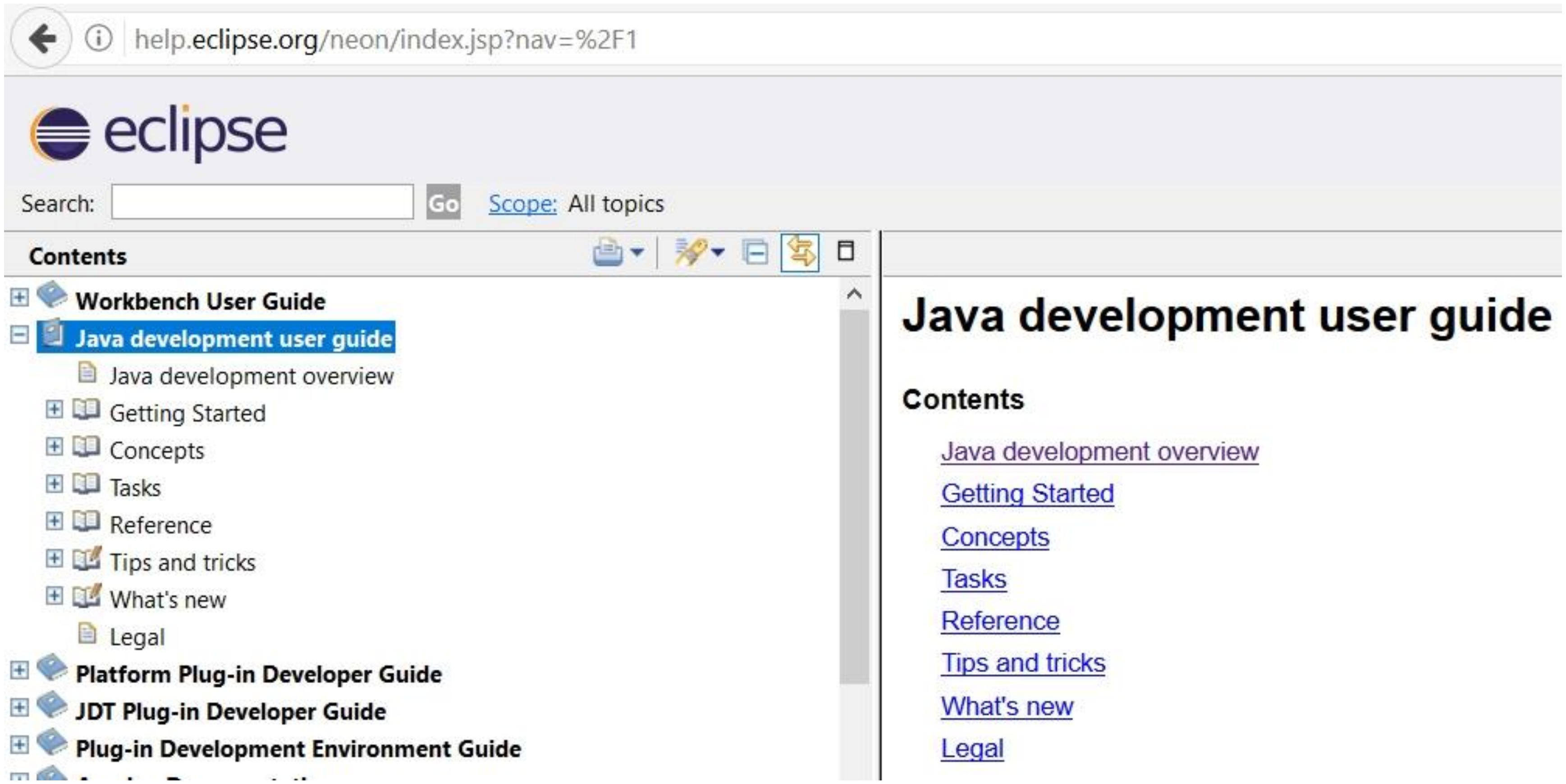
» Ask questions on project **forums**; if you're not sure where to look, start with the **newcomer forum**

» Open bug reports and feature requests in **Eclipse Bugzilla**

» Connect live with project developers **via IRC**

» Find companies that provide **training and consulting**

https://www.eclipse.org/neon/



help.eclipse.org/neon/index.jsp?nav=%2F1

Search: Go Scope: All topics

Contents

- Workbench User Guide
- **Java development user guide**
  - Java development overview
  - Getting Started
  - Concepts
  - Tasks
  - Reference
  - Tips and tricks
  - What's new
  - Legal
- Platform Plug-in Developer Guide
- JDT Plug-in Developer Guide
- Plug-in Development Environment Guide

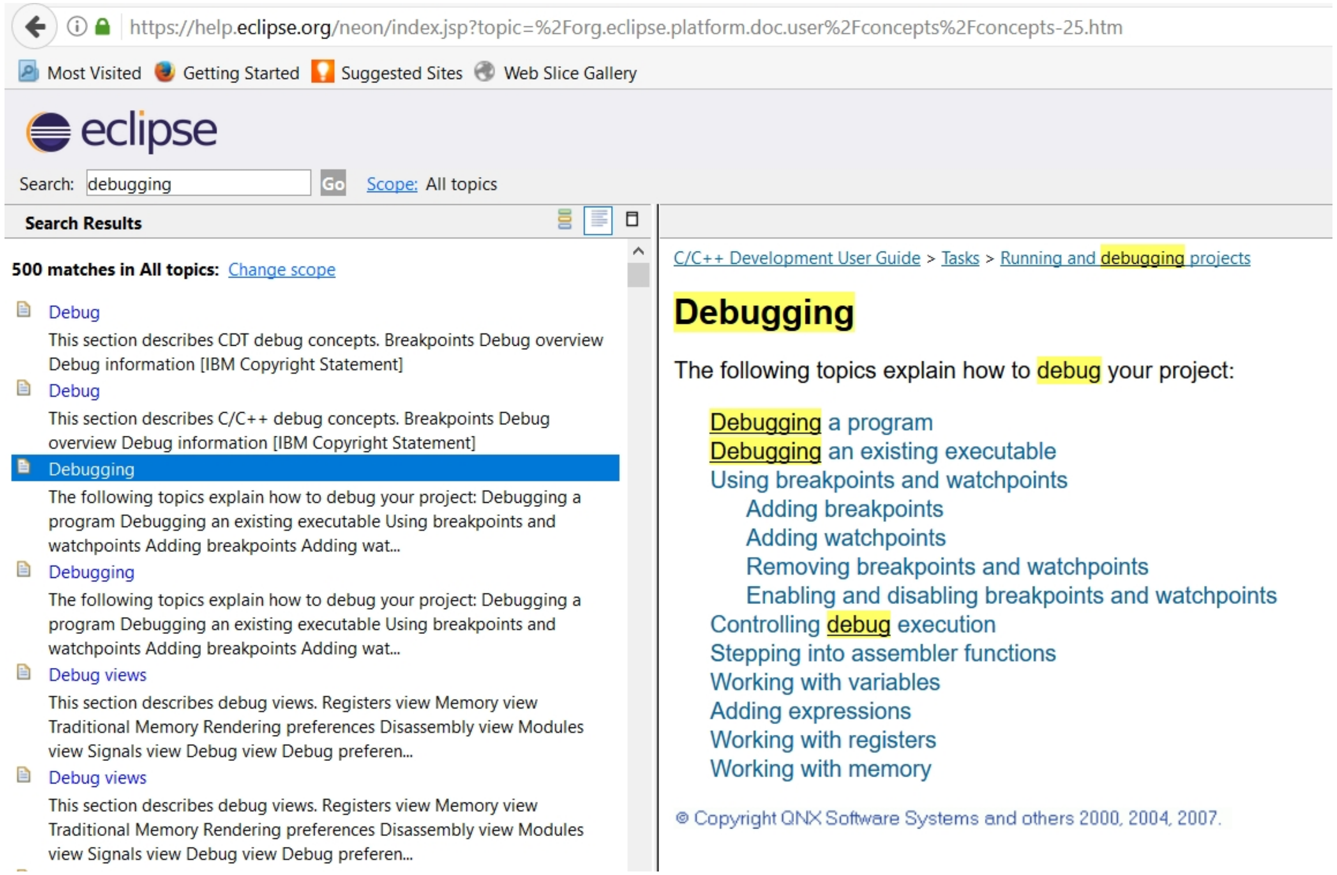
Java development user guide

Contents

- Java development overview
- Getting Started
- Concepts
- Tasks
- Reference
- Tips and tricks
- What's new
- Legal

Getting help information

- Query in Google:
  - how to debug java program in eclipse



https://help.eclipse.org/neon/index.jsp?topic=%2Forg.eclipse.platform.doc.user%2Fconcepts%2Fconcepts-25.htm

Most Visited | Getting Started | Suggested Sites | Web Slice Gallery

eclipse

Search: debugging Go Scope: All topics

Search Results

500 matches in All topics: Change scope

Debug
This section describes CDT debug concepts. Breakpoints Debug overview Debug information [IBM Copyright Statement]

Debug
This section describes C/C++ debug concepts. Breakpoints Debug overview Debug information [IBM Copyright Statement]

Debugging
The following topics explain how to debug your project: Debugging a program Debugging an existing executable Using breakpoints and watchpoints Adding breakpoints Adding wat...

Debugging
The following topics explain how to debug your project: Debugging a program Debugging an existing executable Using breakpoints and watchpoints Adding breakpoints Adding wat...

Debug views
This section describes debug views. Registers view Memory view Traditional Memory Rendering preferences Disassembly view Modules view Signals view Debug view Debug preferen...

Debug views
This section describes debug views. Registers view Memory view Traditional Memory Rendering preferences Disassembly view Modules view Signals view Debug view Debug preferen...

C/C++ Development User Guide > Tasks > Running and debugging projects

Debugging

The following topics explain how to debug your project:

- Debugging a program
- Debugging an existing executable
- Using breakpoints and watchpoints
  - Adding breakpoints
  - Adding watchpoints
  - Removing breakpoints and watchpoints
  - Enabling and disabling breakpoints and watchpoints
- Controlling debug execution
- Stepping into assembler functions
- Working with variables
- Adding expressions
- Working with registers
- Working with memory

© Copyright QNX Software Systems and others 2000, 2004, 2007.

https://blog.codecamp.jp/eclipse

Tech 2 GO
となりの田中さんのためのテクノロジーメディア
Powered by CodeCamp
プログラミング
Webデザイン
テクノロジー
アプリ開発
ビジネス
キャリア
Tech2GO › プログラミング › 【入門編】Eclipseを使ったJavaプログラミングを基礎から解説

【入門編】Eclipseを使ったJavaプログラミングを基礎から解説

更新日: 2017年5月22日　公開日: 2016年7月25日

Summary

- Eclipse is a powerful IDE.
- It is a main instrument to create and debug Java application.
- It protects users from many types of errors at the development process.
- It provides users with simple and powerful tools to debug applications.
- It has intuitively understood GUI.