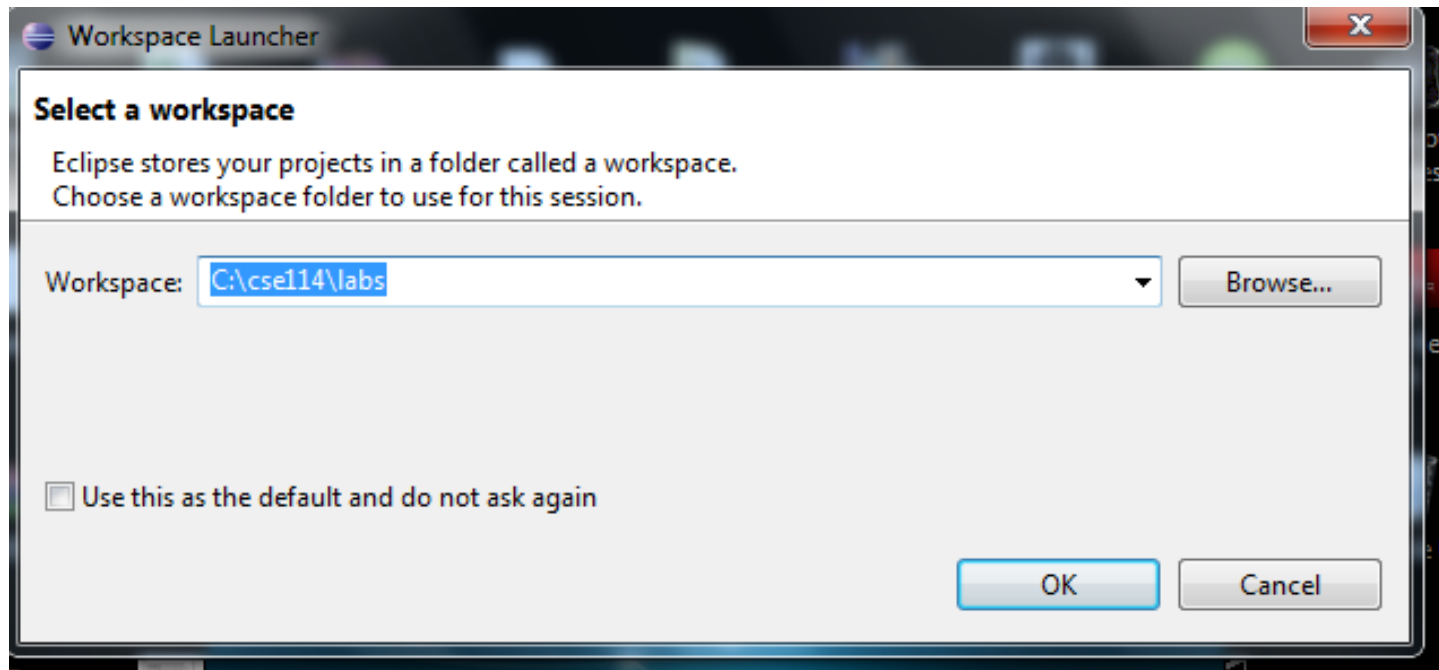


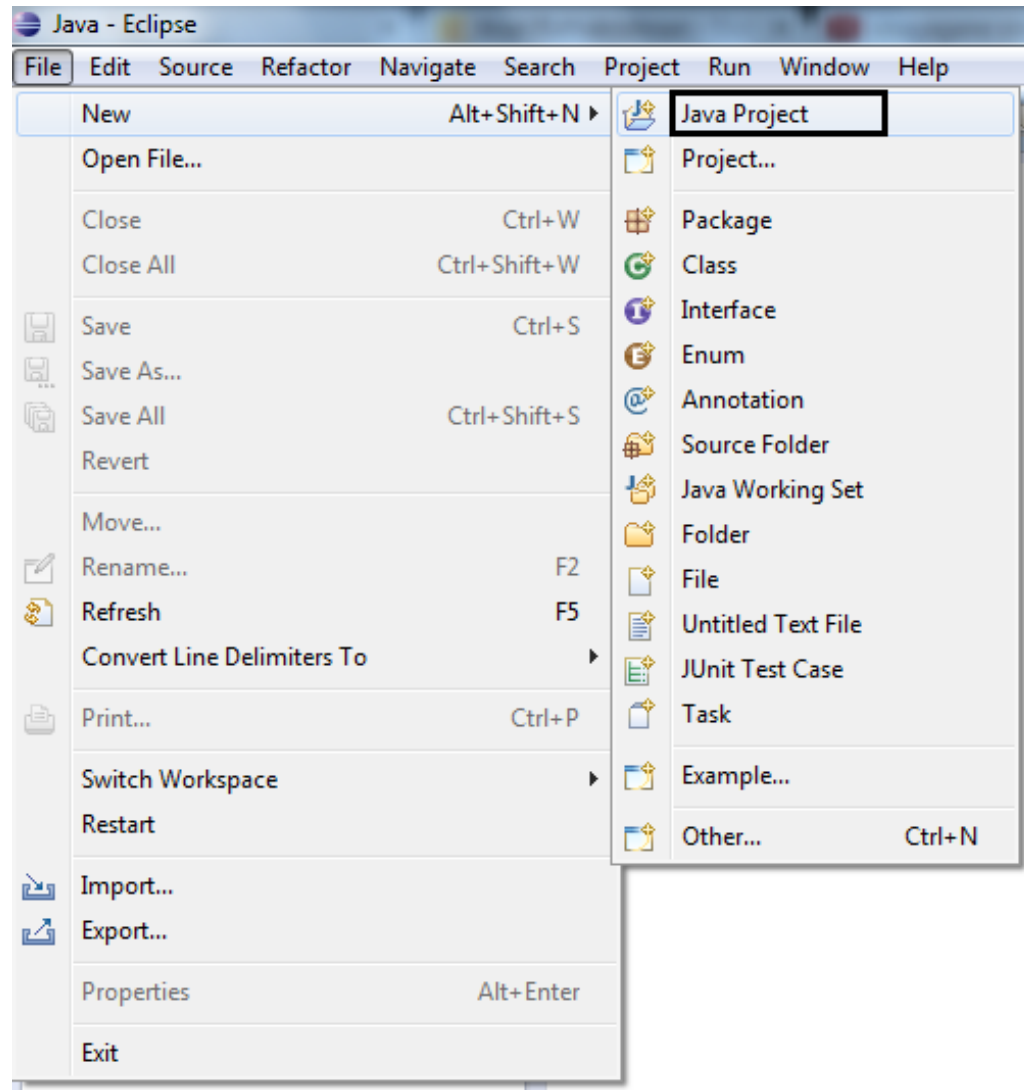
Getting Started with Eclipse

CSE 114 Lab1

Setting up Workspace



Creating a New Project



Creating a New Project: Step 2

Enter the
new project
name here

New Java Project

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

Project name: **HelloWorld**

☒ Use default location

Location: C:\cse114\labs\HelloWorld [Browse...](#)

JRE

☒ Use an execution environment JRE: JavaSE-1.7

☐ Use a project specific JRE: jre7

☐ Use default JRE (currently 'jre7') [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

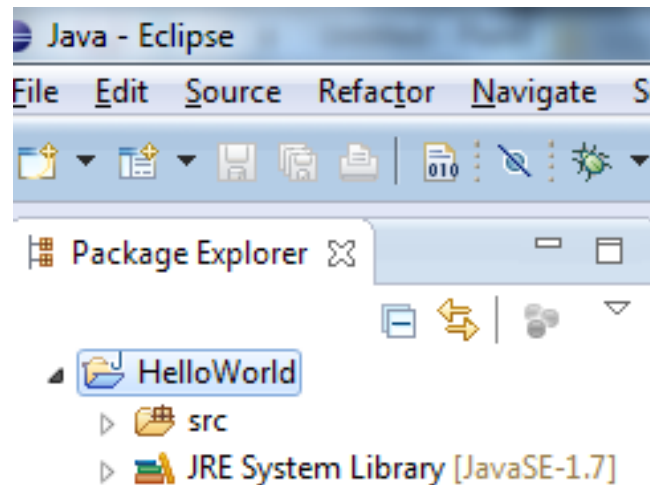
☐ Add project to working sets

Working sets: [Select...](#)

? The wizard will automatically configure the JRE and the project layout based on the existing source.

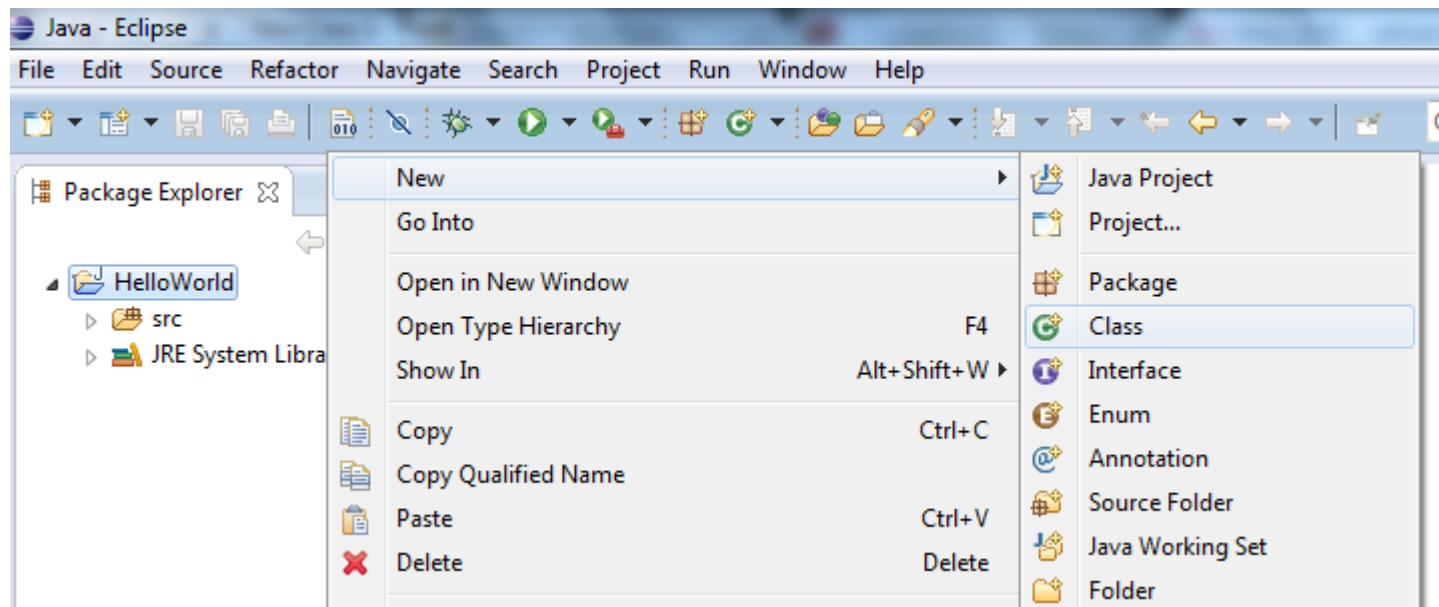
[? < Back](#) [Next >](#) [Finish](#) [Cancel](#)

Creating a New Project



First Java Program

- Java programs are all classes (as many other things are)
- Create a new Java class.



Creating a new Class

Enter class name.

New Java Class

Type already exists.

Source folder: HelloWorld/src Browse...

Package: (default) Browse...

☐ Enclosing type: Browse...

Name: HelloWorld

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add...
Remove

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

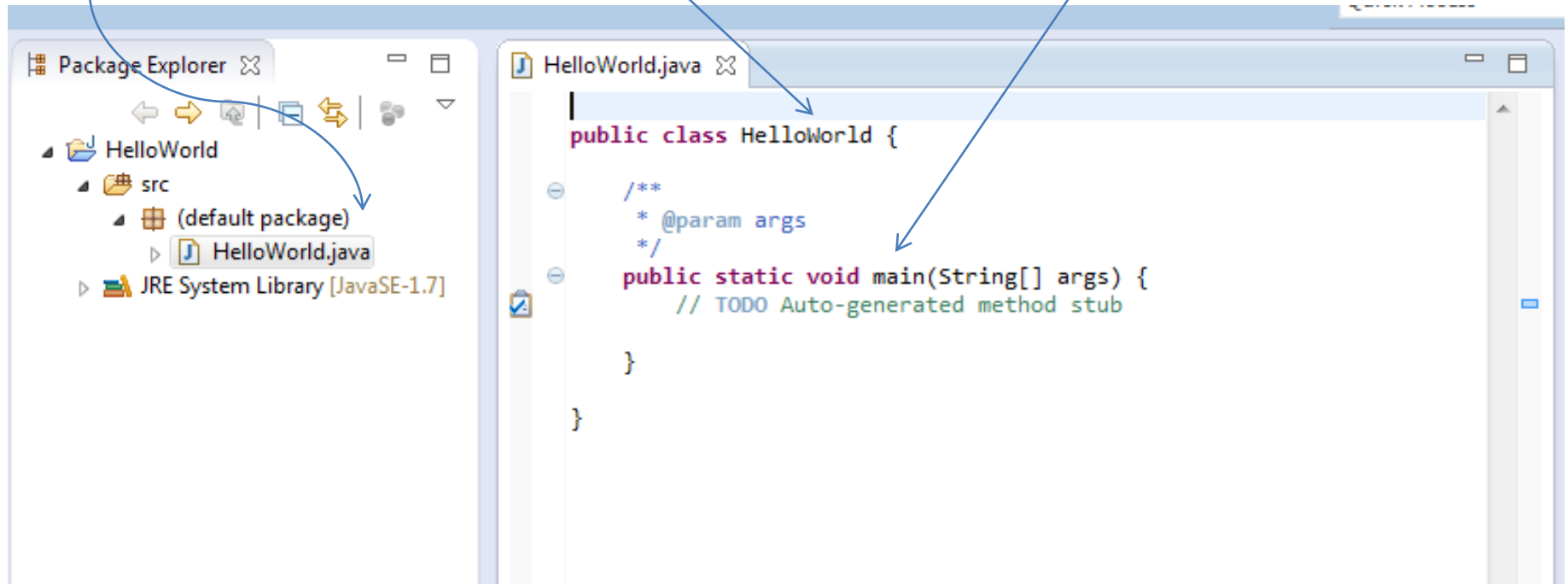
? Finish Cancel

Check this box for main method

New Class

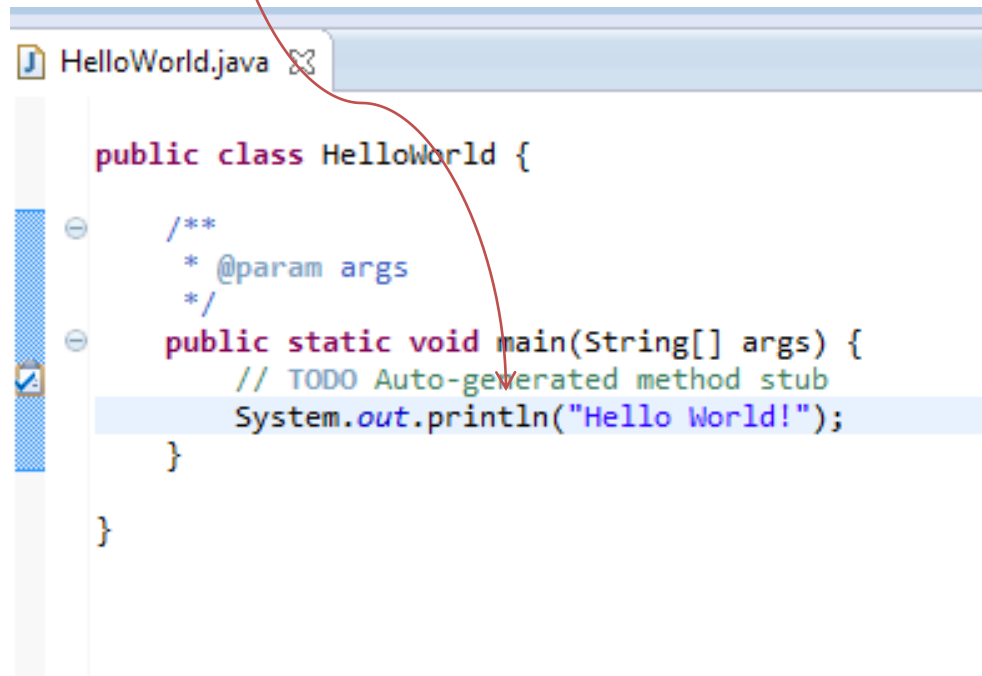
New Class name is the file name.

Your program starts executing from main(). The argument type is String[] always.



Hello World!

To print we use
System.out.println
Or
System.out.print



```
HelloWorld.java

public class HelloWorld {

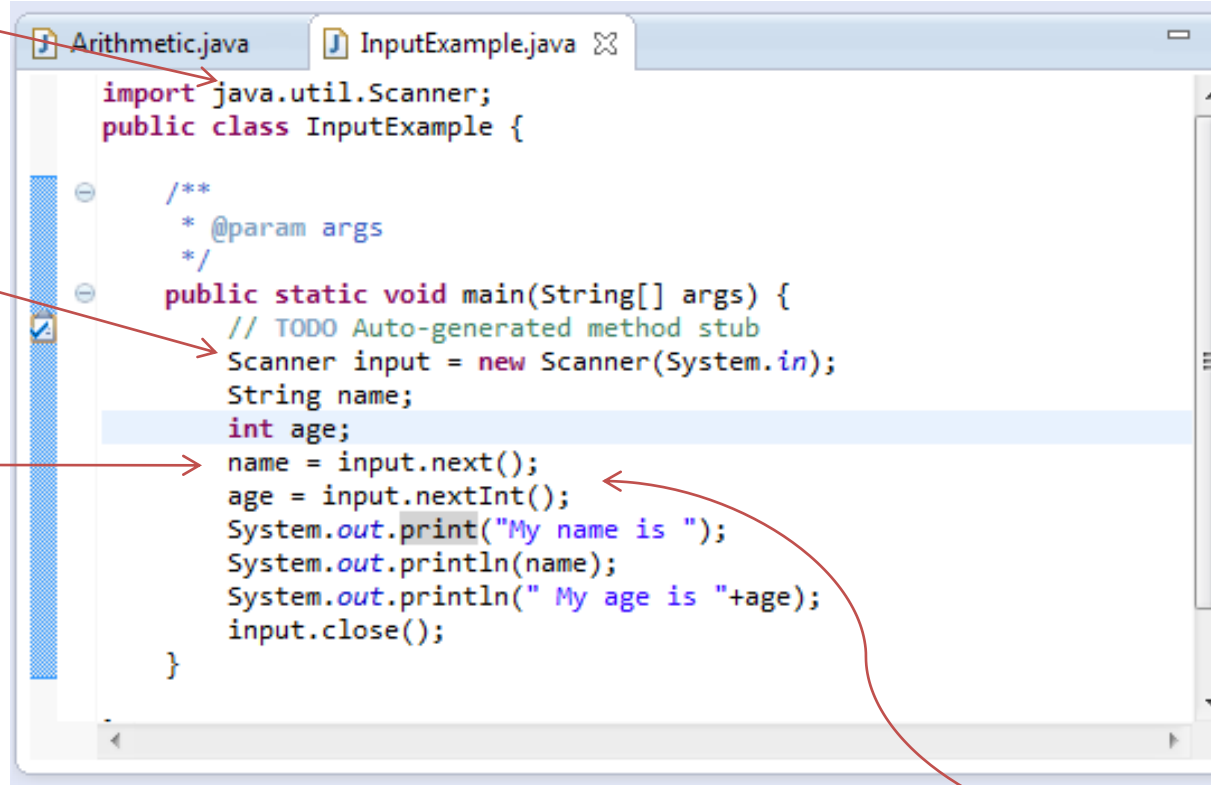
    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Hello World!");
    }

}
```

Input

Scanner class is used for getting input.

Use input.next()
For getting a
String as input

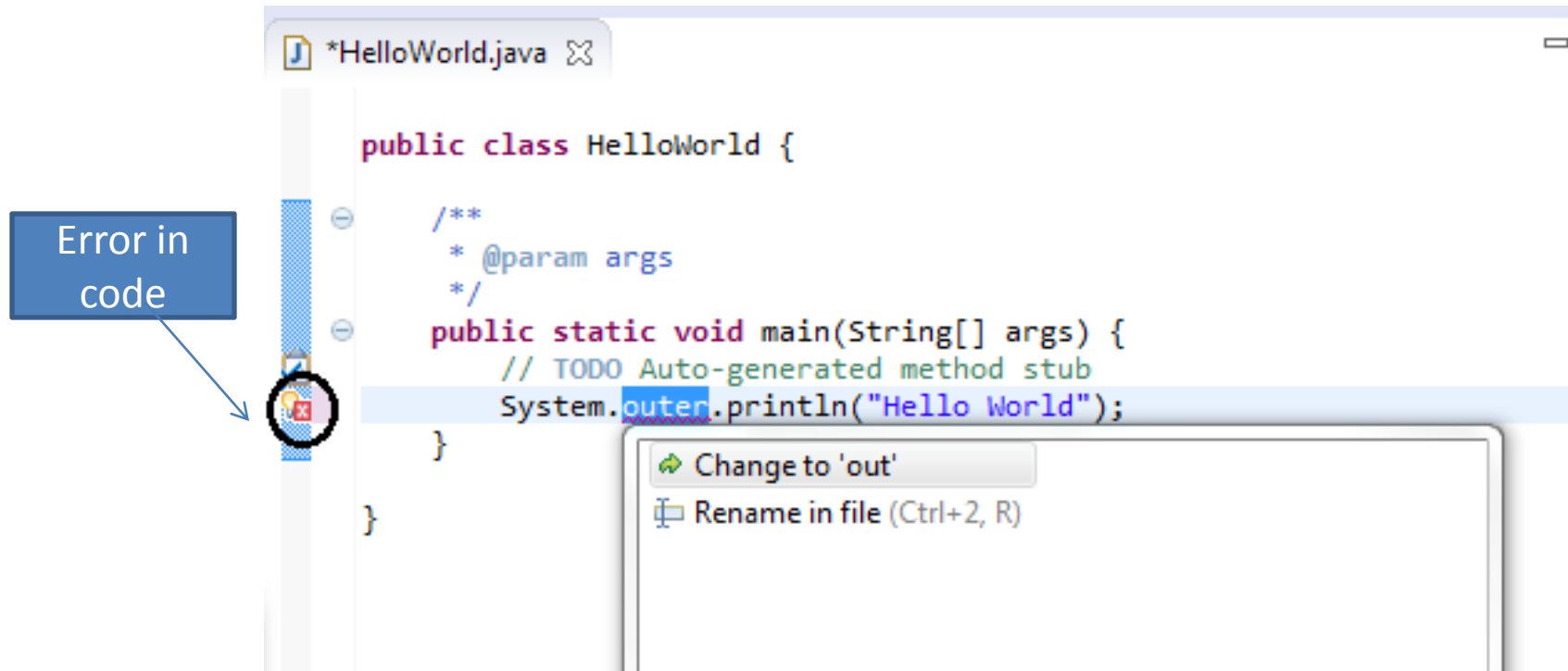


```
Arithmetic.java | InputExample.java ✕
import java.util.Scanner;
public class InputExample {

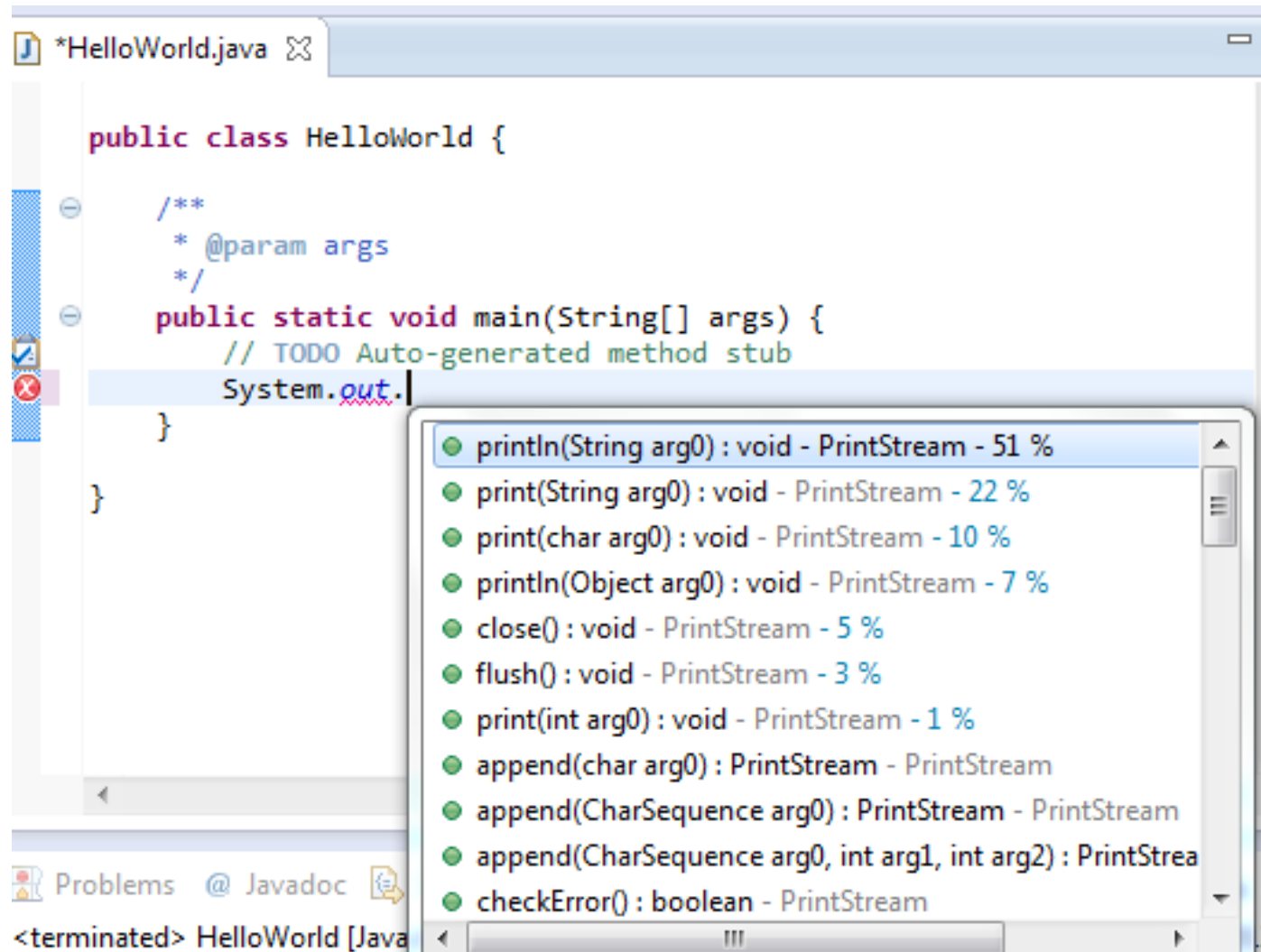
    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner input = new Scanner(System.in);
        String name;
        int age;
        name = input.next();
        age = input.nextInt();
        System.out.print("My name is ");
        System.out.println(name);
        System.out.println(" My age is "+age);
        input.close();
    }
}
```

Use input.nextInt() for integers

Eclipse Features: Errors in code

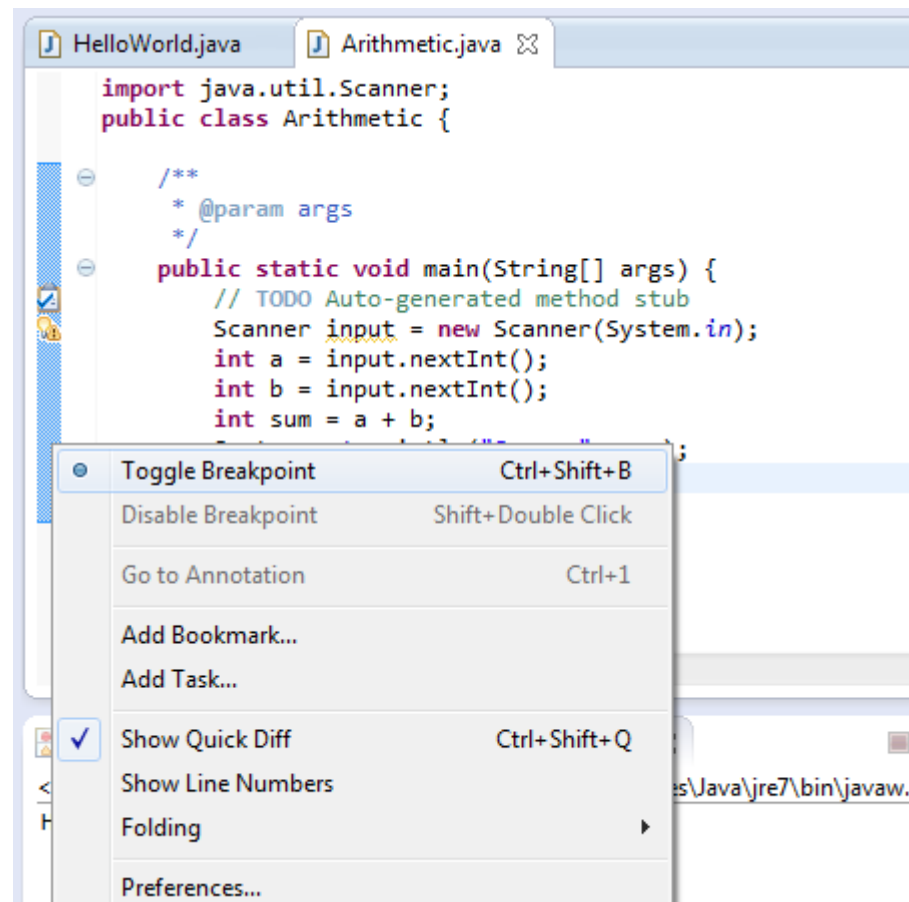


Eclipse Features: Auto complete



Eclipse Features: Debugging

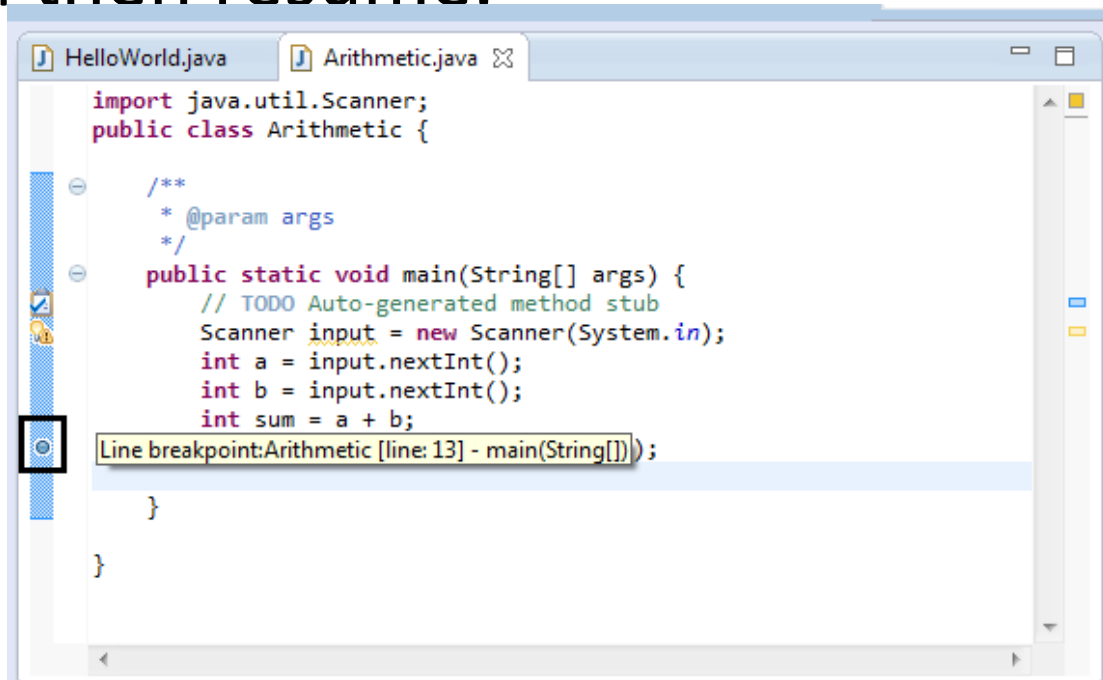
Step 1: Set a break point



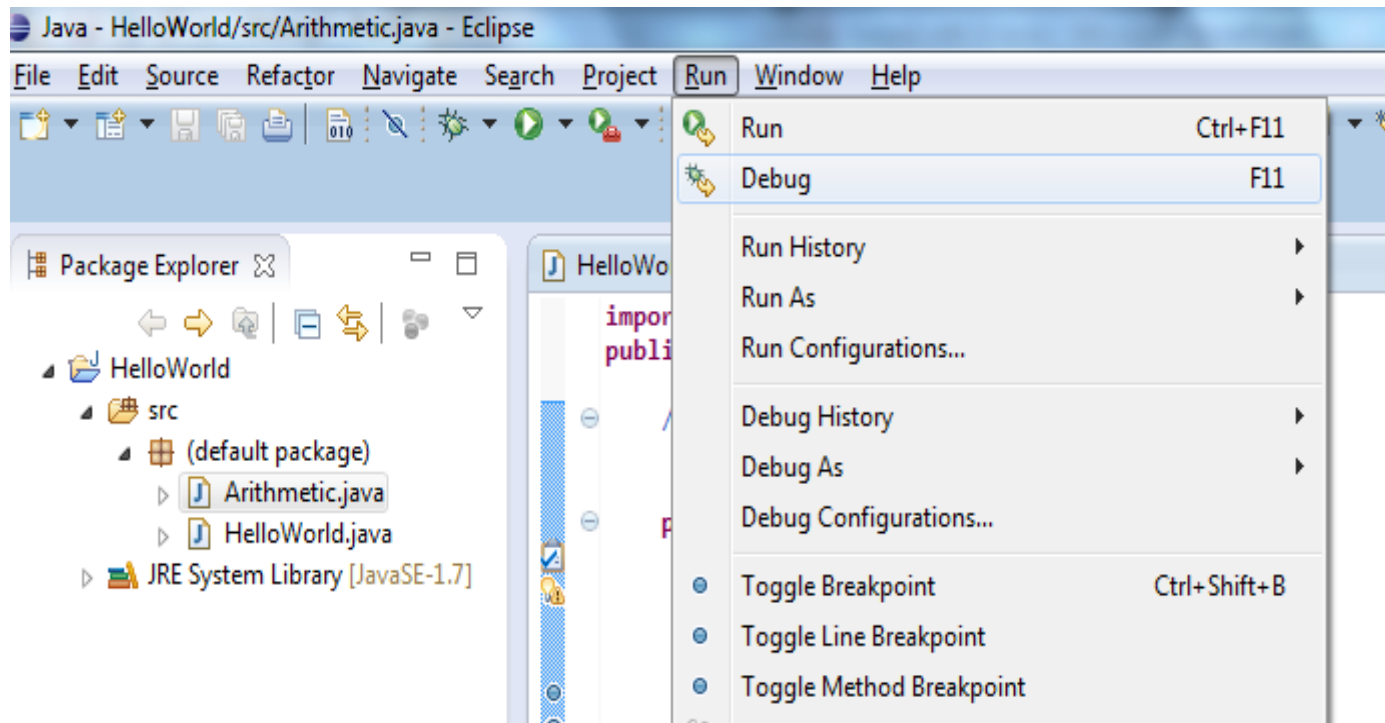
Debugging: Break Point

- Program is run till a break point and then paused.
- We can look at what happened to the variables and then resume.

Break point



Start the Debugger



Currently Paused
here.

Debug Step 1

Variables and Values

Resume (F8)

Debug

- Arithmetic [Java Application]
 - Arithmetic at localhost:50098
 - Thread [main] (Running)
 - Arithmetic [Java Application]
 - Arithmetic at localhost:50104
 - Thread [main] (Suspended (breakpoint at line 12 in Arithmetic))
 - Arithmetic.main(String[]) line: 12

Variables

Name	Value
args	String[0] (id=16)
input	Scanner (id=18)
a	10
b	14

Arithmetic.java

```
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
    Scanner input = new Scanner(System.in);  
    int a = input.nextInt();  
    int b = input.nextInt();  
    int sum = a + b;  
    System.out.println("Sum = " + sum);  
    input.close();  
}
```

Outline

- Arithmetic
 - main()

Click Resume to
move to next
break point

Currently Paused
here.

Debug Step 2

Notice the variable
sum

The screenshot displays the Eclipse IDE interface during a debug session. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. Below the menu is a toolbar with various icons for file operations, navigation, and debugging. The main workspace is divided into several panels:

- Debug Console:** Shows the execution stack. The top entry is "Arithmetic [Java Application]" with a sub-entry "Arithmetic at localhost:50098". Below it is "Thread [main] (Running)". The second entry is "Arithmetic [Java Application]" with a sub-entry "Arithmetic at localhost:50110". Under this, "Thread [main] (Suspended (breakpoint at line 14 in Arithmetic))" is shown, with "Arithmetic.main(String[]) line: 14" highlighted.
- Variables View:** A table showing the current state of variables:

Name	Value
args	String[0] (id=16)
input	Scanner (id=19)
a	10
b	14
sum	24
- Arithmetic.java Editor:** The source code is displayed with the breakpoint at line 14 highlighted in green:

```
int a = input.nextInt();
int b = input.nextInt();
int sum = a + b;
System.out.println("Sum = " + sum);
input.close();
}
```
- Outline View:** Shows the project structure with "Arithmetic" and "main(String)" listed.