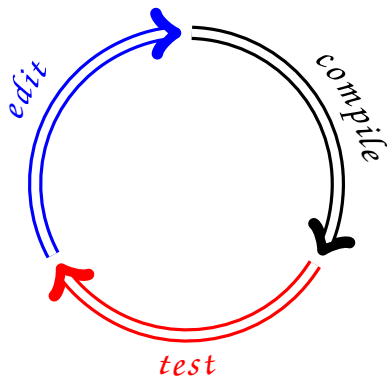


# Program Development



# Developing Java Programs – BlueJ

The image shows the BlueJ IDE interface. On the left, a code editor displays the following Java code for the `Student` class:

```
/**  
 * A class representing students for a simple  
 *  
 * @author Michael Kolling  
 * @version 1.0, January 1999  
 */  
class Student extends Person  
{  
    private String SID; // student ID number  
  
    /**  
     * Create a student with default settings  
     */  
    public Student()  
    {  
        super("(unknown name)", 0000);  
        SID = "(unknown ID)";  
    }  
  
    /**  
     * Create a student with given name, year  
     */  
}
```

On the right, the class diagram shows the following structure:

- `Person` is an abstract class (indicated by `<<abstract>>`).
- `Staff` and `Student` inherit from `Person` (indicated by solid lines with hollow triangle heads).
- `Address` is associated with `Person` (indicated by a dashed line with an open arrowhead).
- `Database` is associated with `Person` (indicated by a dashed line with an open arrowhead).

A context menu is open over the `Person` class, showing the following options:

- inherited from Object
- inherited from Person
- String getRoom()
- void setRoom(String)
- String toString()
- Inspect
- Remove

The IDE window title is "BlueJ: people2". The menu bar includes "Project", "Edit", "Tools", "View", and "Help". The toolbar includes "New Class...", "Compile", and "View" buttons. The "View" panel shows "Uses" and "Inheritance" checked. The "student\_1:" and "staff\_1:" instances are visible at the bottom, with "Student" and "Staff" labels respectively. A "saved" button is located at the bottom right of the IDE window.

# Developing Java Programs – Eclipse

The screenshot displays the Eclipse IDE interface. The title bar reads "Java - HelloWorld.java - Eclipse Platform" and the system clock shows "Tue Jan 06, 08:57:25". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The Package Explorer on the left shows a project structure with a "Test" folder containing "HelloWorld.java". The main editor window shows the source code for "HelloWorld.java", which includes a package declaration, a class declaration, and a main method. The Ant view on the right is empty. The Properties view at the bottom is also empty.

```
/*
 * Created on Dec 22, 2003
 *
 * To change the template for this generated file go to
 * Window - Preferences - Java - Code Generation - Code and Comments
 */
package src;

/**
 * @author depco
 *
 * To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Generation - Code and Comments
 */
public class HelloWorld {

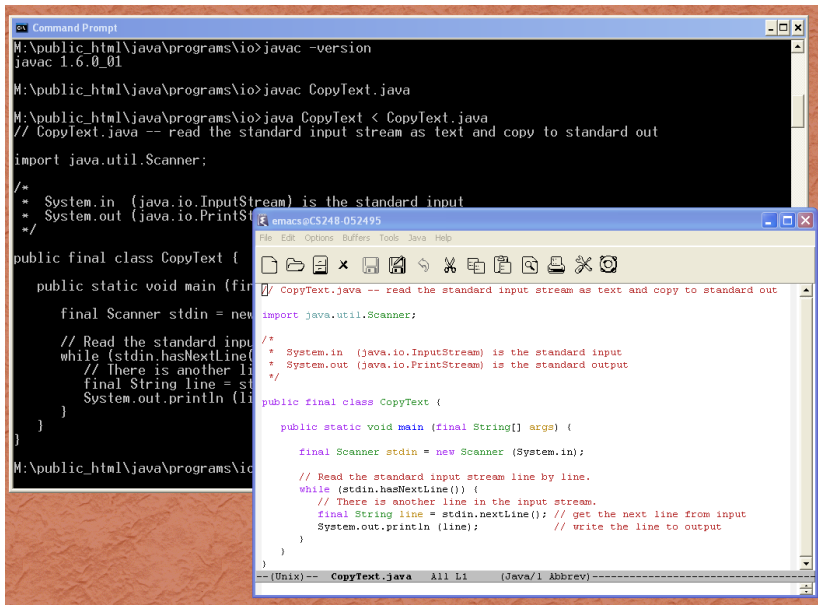
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Package Explorer Hierarchy

Problems | Console | CVS Console | Properties

Writable Smart Insert 1 : 1

# Developing Java Programs – Emacs



```
M:\public_html\java\programs\io>javac -version
javac 1.6.0_01

M:\public_html\java\programs\io>javac CopyText.java

M:\public_html\java\programs\io>java CopyText < CopyText.java
// CopyText.java -- read the standard input stream as text and copy to standard out

import java.util.Scanner;

/*
 * System.in (java.io.InputStream) is the standard input
 * System.out (java.io.PrintStream) is the standard output
 */

public final class CopyText {

    public static void main (final String[] args) {

        final Scanner stdin = new Scanner (System.in);

        // Read the standard input stream line by line.
        while (stdin.hasNextLine()) {
            // There is another line in the input stream.
            final String line = stdin.nextLine(); // get the next line from input
            System.out.println (line); // write the line to output
        }
    }
}

M:\public_html\java\programs\io>
```

```
emacs@CS248-052495
File Edit Options Buffers Tools Java Help

// CopyText.java -- read the standard input stream as text and copy to standard out
import java.util.Scanner;

/*
 * System.in (java.io.InputStream) is the standard input
 * System.out (java.io.PrintStream) is the standard output
 */

public final class CopyText {

    public static void main (final String[] args) {

        final Scanner stdin = new Scanner (System.in);

        // Read the standard input stream line by line.
        while (stdin.hasNextLine()) {
            // There is another line in the input stream.
            final String line = stdin.nextLine(); // get the next line from input
            System.out.println (line); // write the line to output
        }
    }
}

--(Unix)-- CopyText.java All L1 (Java/1 Abbrev)--
```

- ▶ compile error
  - ▶ syntax error — example program
  - ▶ semantic error — example program
    - ▶ type error — example program

- ▶ style error — example program

Style errors are mistakes in the program source code that contravene policy or hamper the ability of programmers to read and understand the program even though the program can be translated by the compiler into a executable program.

- ▶ execution error or (fatal) runtime error — example program

Runtime errors are mistakes that manifest themselves during the execution of the program. These errors prevent the computer from completing the execution of the program.

- ▶ logic error — example program

Logic errors are mistakes in the behavior of the program even though the program can be translated into a running, executable program.

If you make a mistake and write a program that goes into an endless loop, and the computer runs out time or space resources and terminates your program prematurely, is this a runtime or a logic error? Either, both, what difference does it make?

What is a compiler warning (as opposed to an error)?

What you ever encountered a compiler warning issued by `javac`?