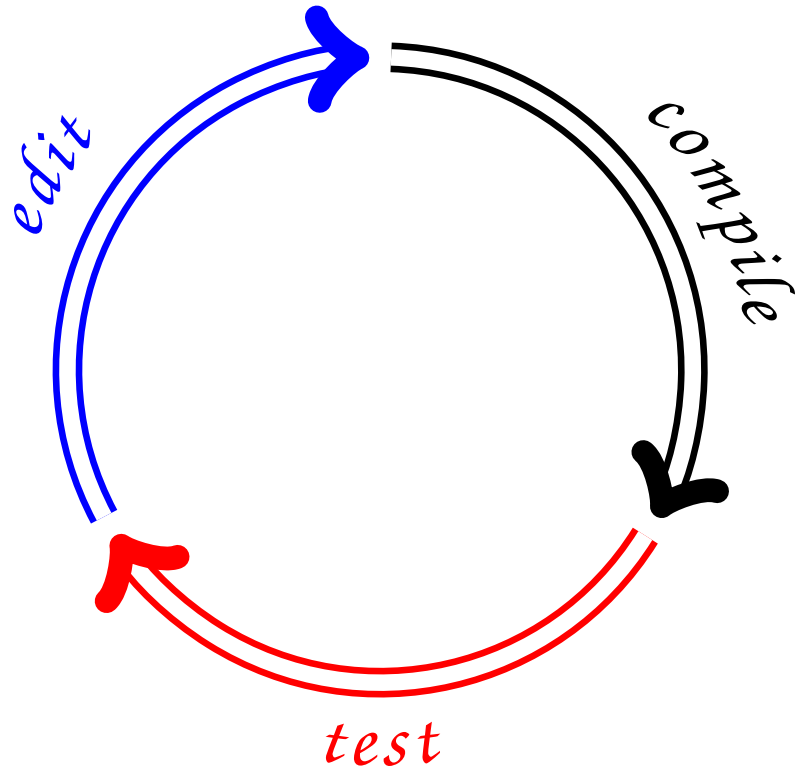
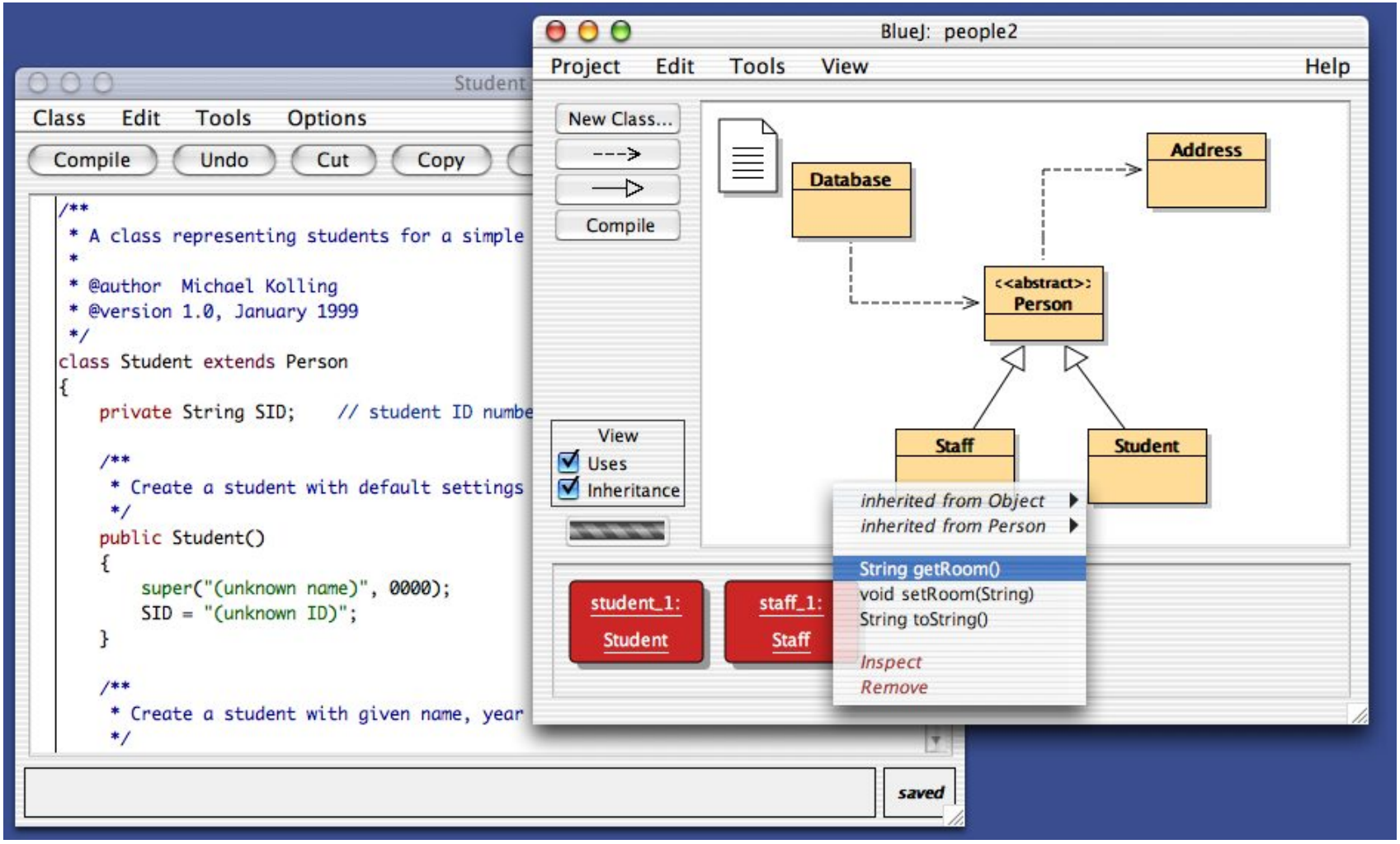
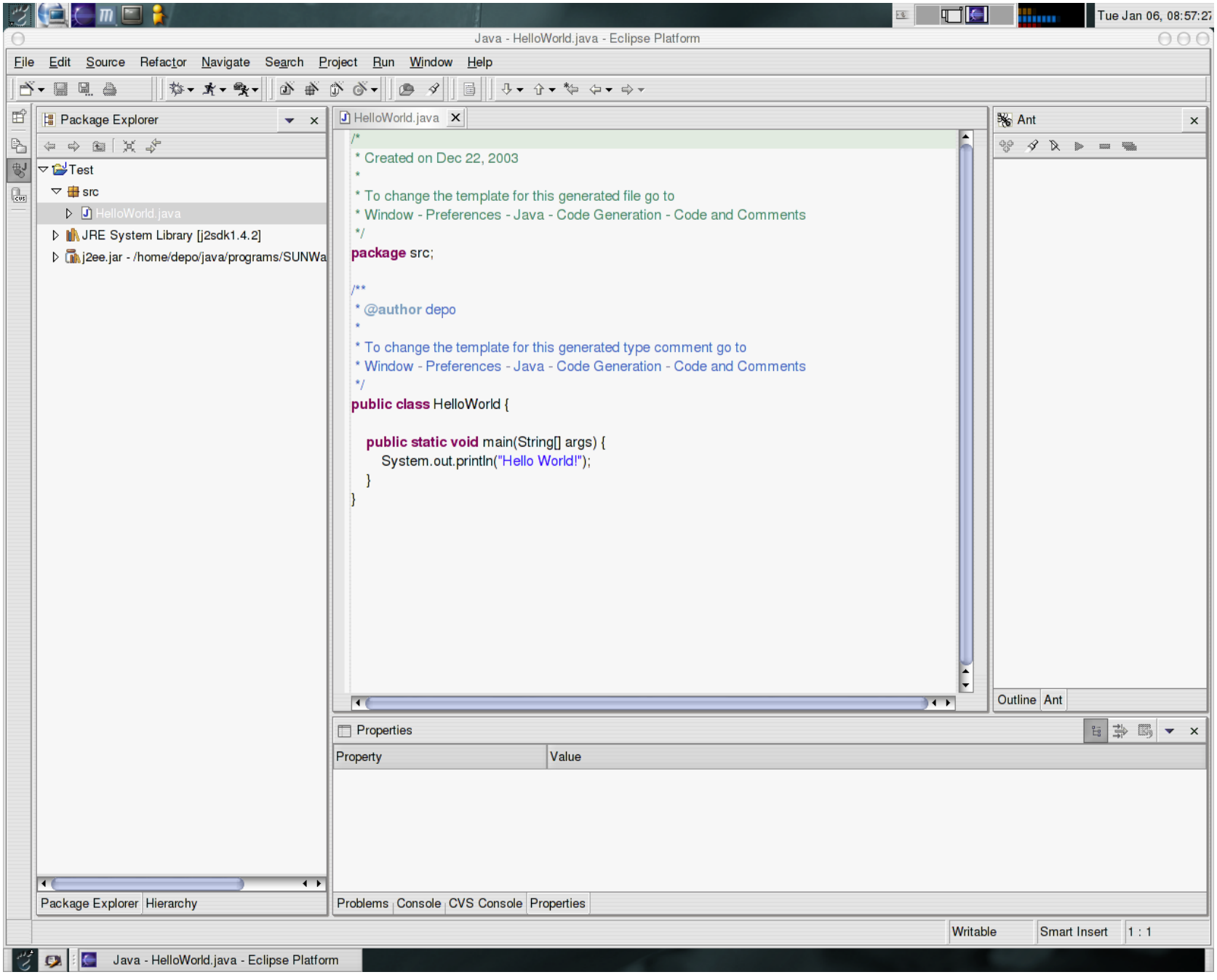
Program Development



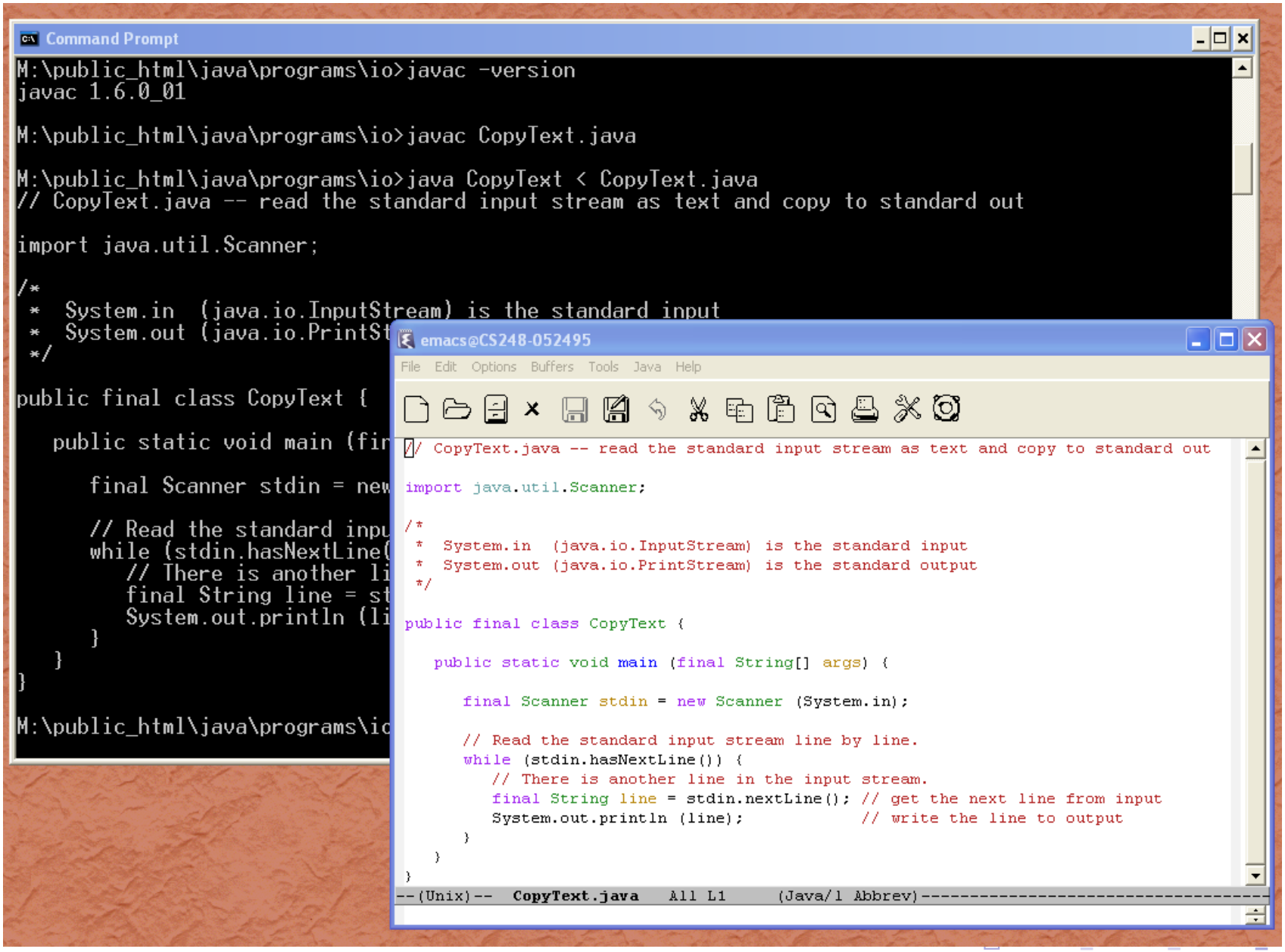
Developing Java Programs – BlueJ



Developing Java Programs – Eclipse



Developing Java Programs – Emacs



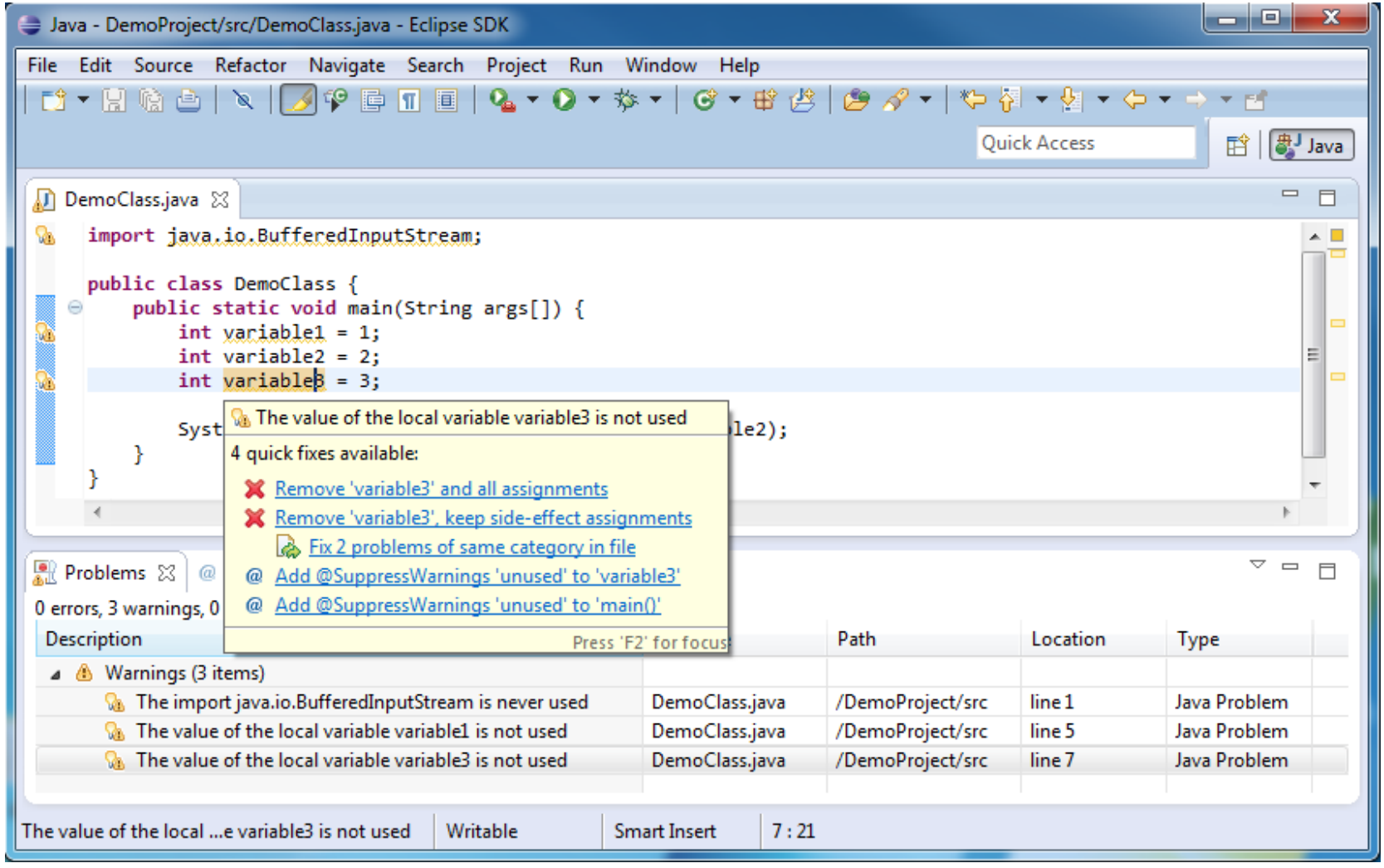
- compile error
  - syntax error — example program
  - semantic error — example program
    - type error — example program
- Eclipse warnings
- style error — example program
  Style errors are mistakes in the program source code that contravene policy or hamper the ability of programers to read and understand 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.

Java requires many suspicious behaviors to be flaged as errors (not just warnings).
According to the Java Language Specification:
“It is a compile-time error if a statement cannot be executed because it is unreachable.”
Java has optional warnings enabled by `javac -Xlint`
In Java 1.6 the complete list (obtained by `javac -X`):

```
cast,deprecation,divzero,empty,unchecked,
fallthrough,path,serial,finally,overrides
```

The warnings `deprecation` and `unchecked` are checked in all cases (regardless of the command line options).

```
java -Xlint:all -Xlint:-serial
```



Eclipse warns about semantic problems not required by the Java language specification

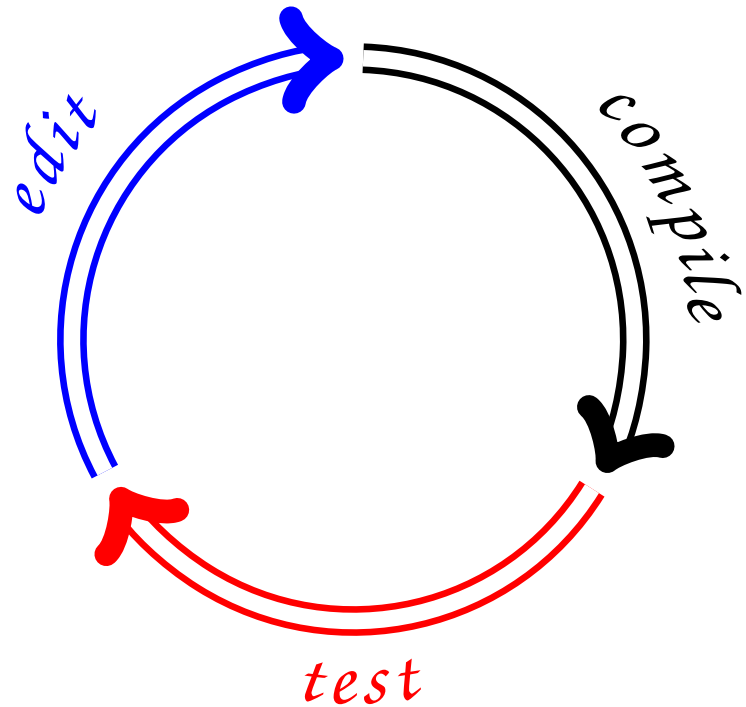
If you make a mistake and write a program that goes into an endless loop, and the computer runs out time or space resources and terminates your program prematurely, is this a runtime or a logic error? Either, both, what difference does it make?

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

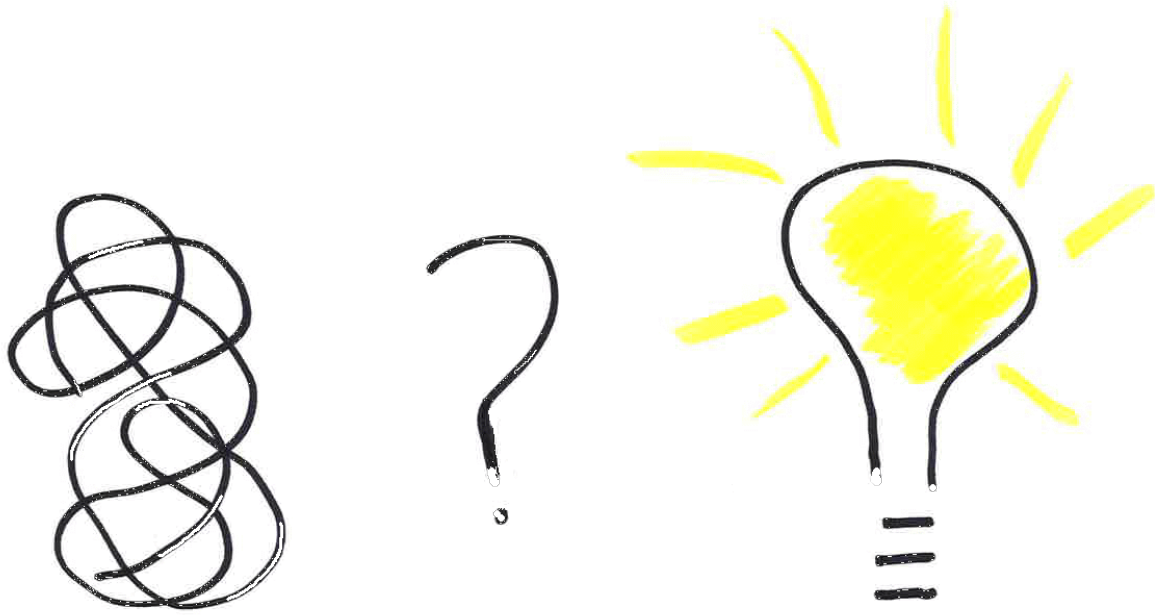
What you ever encountered a compiler warning issued by `javac`?

Editing versus refactoring

Program Development



At what point does planning and thinking come in?



1. design
2. experience
3. problem solving
4. pseudo code, flow charts
5. bring pencil and paper to lab