- Debugging is the process of detecting and removing of existing errors (bugs)in a software.
- To debug a program
    - We should isolate the source code and then fix it.
    - Stop the execution wherever required to get the value of variables or reset the program variables.
    - We trace the program execution step-by-step by evaluating the value of variables
- To stop the execution, we use “Breakpoint”.
- Once the program is stopped we can investigate variables, change their content, etc.

Setting Breakpoints

- To define a breakpoint
    - right-click in the left margin and select *Toggle Breakpoint*.
    - double-click on this position.

Starting the Debugger

- To debug, select a Java file in package by right-clicking on it and select Debug as Java Application.
- Or we can select debug mode from toolbar
- If you have not defined any breakpoints, program executes normally. So to debug we need to define breakpoints.
- Eclipse asks to switch to the *Debug perspective* once a stop point is reached. Answer *Yes* in the dialog.

Controlling the program execution

- Step Into(f5): goes inside method else it proceed to next line.
- Step Over(f6):process current line and proceed to next line.
- Step Return(f7): finishes currently executing method and goes back to calling method.
- Resume: Continues execution until next breakpoint or end of the program.

Breakpoints view and deactivating breakpoints

- We can delete and deactivate *breakpoints* and modify their properties.
- To deactivate a breakpoint, remove the corresponding checkbox in the *Breakpoints* view. To delete it you can use the corresponding buttons in the view toolbar.
- To disable all breakpoints, we can press the “skip all breakpoint” button.
- If we press it again, all breakpoints are reactivated.

Variables view:

- The *Variables* view displays fields and local variables.
- To see the variables in view, we have to execution.
- We can change the values assigned to variable at runtime.

Breakpoints Properties:

- right-click → Breakpoint Properties.
- In breakpoint properties we can define a condition that restricts the activation of this breakpoint.
- Two types of conditions:
    - For example specify a breakpoint that it should become active after it has reached some more times by using "***Hit Count*** **property**".
    - We can also create a **conditional expression**, by specifying condition
        - It stops execution only when the condition is true.

Watchpoint:

- A watchpoint is a special breakpoint that stops the execution of an application whenever the value of a given expression changes
    - To stop execution when the watch expression is **read**, select the **Read check box**.
    - To stop execution when the watch expression is **written** to, select the **Write check box**.

Exception breakpoints

- we can set Exception breakpoints for thrown exceptions.
- To define an exception breakpoint click on the Java Exception Break Point button icon in the *Breakpoints* view toolbar.
- we can configure, if the debugger should stop at caught or uncaught exceptions.

Method breakpoint

- A method breakpoint is defined by double-clicking in the left margin of the editor next to the method header.
- We can configure if you want to stop the program before entering or after leaving the method.