Step into -used to get into the method (F5)

Step Over – execture line by line selection (F6)

Step Return – used to return to back to the calling function starting point or called part … eg- be.write();

Step into Selection – If there is chian of methods ’ int firstPersonAge = DataUtil.*getPersonData*().get(0).getAge();’ and you want to get into specific method like get(0) and getAge() you can select the method and click Step into selection

Run to Line – Used to directly get into the respective line \

-------------------------------------------------------------------------------------------------------------------------

**Breakpoints view in details and Deactivating BreakPoints**

Disable Is used to disable a particular line line the program where if the execution is continued the BeakPoint line which is disabled wont be stopped there means the line will be executed, since the breakpoint is disabled the program wont be stopped at the beakpoint place

Also you can Disable All to disable every line

SkipAll – used to skip all the break points

----------------------------------------------------------------------------------------------------------------------------

The **Inspect** feature in Eclipse IDE's debugger is a powerful tool that allows you to evaluate and examine the value of expressions or variables during runtime.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Evaluating Expressing**

Right click -> Watch or create a new Expression and test with possible expressions.

-----------------------------------------------------------------------------------------------------------------------------------

Execute- > Is used to execute a specific or the selected line no return is returned just the method is

Logical Structure – is used to show the logical view or the more clarity view of the variales.. we can click on the show logical structure

**Changing of variable values during debugging**

Just go and double click on the vairbale lon the side and change the values and try …. With different values

**Display View**

By using Debug shell we can use this , we can use this area to write , and inspect java expressions in the context of debug session

Example e- “ DataUtil.getPersonData().get(0).getAge() ” -> Select the query and press the Display result evaluate button to execute the expression inside

\* We can change the code and see the results insitaly for particular code or expressions \*

**Use of Detail Formatter**

Whenever debugging collections or objects or some detail of objects instead of having look in default toString method ouput , we can create a detail formatter and while debugging the op according to the custom formatter or detail formatter can be used … handy feature

A computer screen shot of a program

AI-generated content may be incorrect.1)

A computer screen shot of a program

AI-generated content may be incorrect.

2)

A screenshot of a computer

AI-generated content may be incorrect.

3)

A screenshot of a computer program

AI-generated content may be incorrect.4)

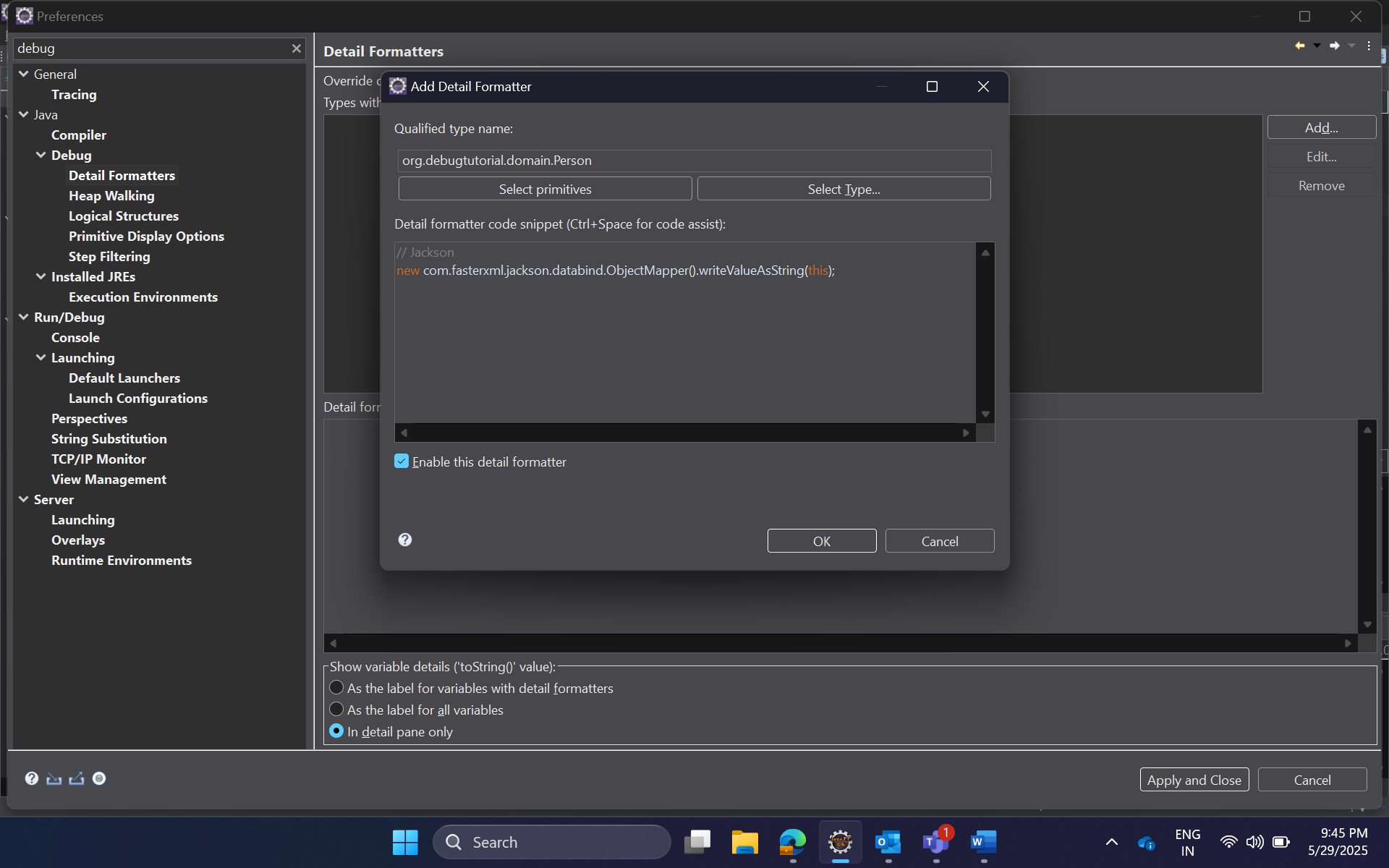
**Displaying objects as JSON in Eclipse Debugging**

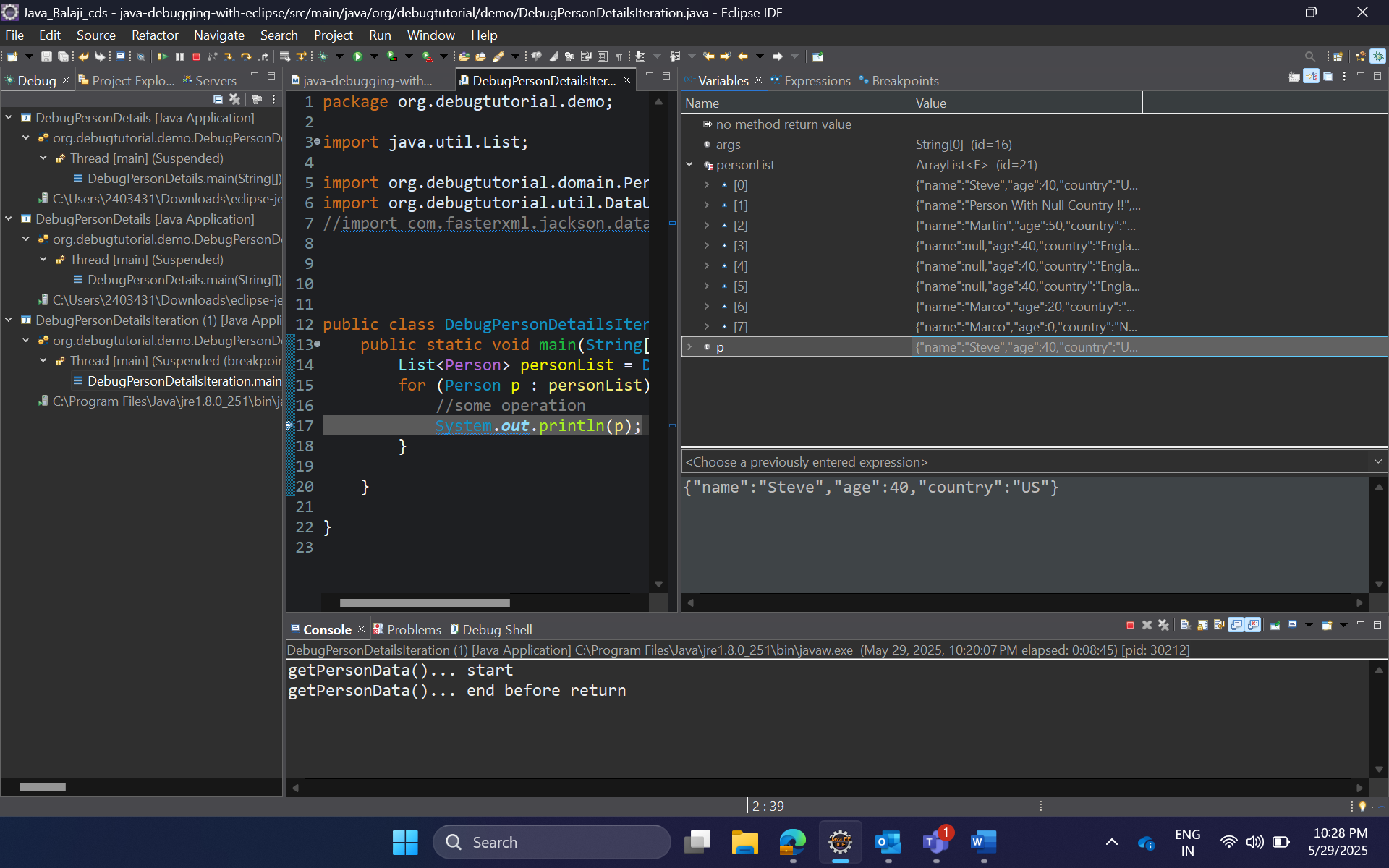
After downloading project from git go to properiteis and check the Build path and compiler in It as per the current java versions… Check the file has M Sybmol and convert it into Mvaen project and update the maven dependices and **click on pom.xml file and right click on It run as Mvaen build -> for the purpose of -> as ‘clean install’**

And thn run …. Check the dependencies Jackson databind and gson are dependencies .

A screenshot of a computer

AI-generated content may be incorrect.





Object’s data are viewed in json

**For pretty json we can use gson as below**

Use this ->

A computer screen shot of a program

AI-generated content may be incorrect.“new com.google.gson.GsonBuilder().setPrettyPrinting().create().toJson(this);”

A screenshot of a computer program

AI-generated content may be incorrect.

NOTE : in when using GSON Null values are not displayed means the field which has null values are not displayed and viewed … but in Jackson all the fields and values are displayed including null values

**Conditional break point and Hit point**

A conditional breakpoint in Eclipse allows you to pause the execution of your program only when a specified condition is met. This is particularly useful for debugging complex scenarios.

A screenshot of a computer

AI-generated content may be incorrect.

**Hit count**

Hit count Is used to execute .. if there is an iteration and this iterations occurs the number of times and to jump to exact point ….(Eg- if we set hit count to 9, the iterations skips first 8and start executing from the 9th..(do step over to get more clarity on it .))

**Trigger point**

It is conditional activation of other break points

We use It when we want to stop at a particular method when called from specific code path. i.e lets say there is a method called validator() that is common in class a,b and c. where discount , checkout, and billing method are the respective methods(**Which has validator() in It**) inside each classes !!! … Now if we place a break point inside validator , the break point(inside the validator) is called every time when the discount, checkout and billing methods are called …. So but the case is I want only the validator() to be called only from specific method - validator() should be only triggered only from checkout or billing !! so we use trigger point at the place

A screen shot of a computer

AI-generated content may be incorrect.

Place a trigger point before it Is called the common method from the actual method from where it is called

A screenshot of a computer program

AI-generated content may be incorrect.

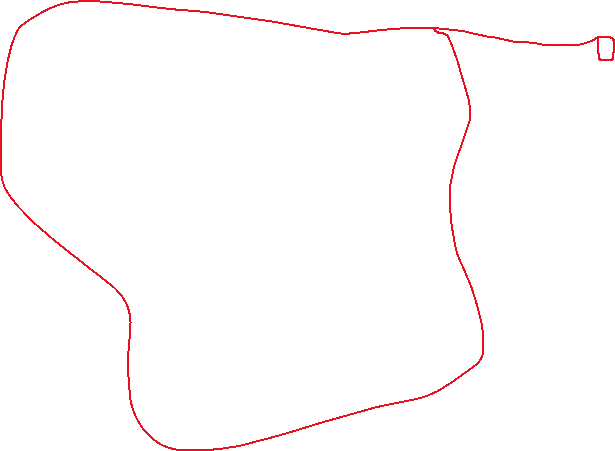
A screenshot of a computer screen

AI-generated content may be incorrect.

**Exception Breakpoint**

A screenshot of a computer screen

AI-generated content may be incorrect.



The breakpoint is hit automatically when there is an exception occurred we don’t need to set a breakpoint where does an exception occurs… we just need to set a exception breakpoint and add what are the expcetions may occurs arithmetic or nullpointer etc….. and debug the program if there is an expcetion the program will stop at the some specific place

**Method Breakpoint**

It is used to control the method flow.. where we can set the **entry** level breakpoint but we can also set **exit**(Means if we tick entry and exit , it will automatically stop at the method begging and stop before the method is going to exit or return … It is useful when there Is a huge lines of code inside the method and we don’t need to search and keep any breakpoint it will do autoamctailly by itself )

A screenshot of a computer

AI-generated content may be incorrect.

Step Filtering

It Is used to avoid going inside the java default classes like system.out.println … .equals()… chatAt()..etc while debugging and also while moving step over and **Step Into**

