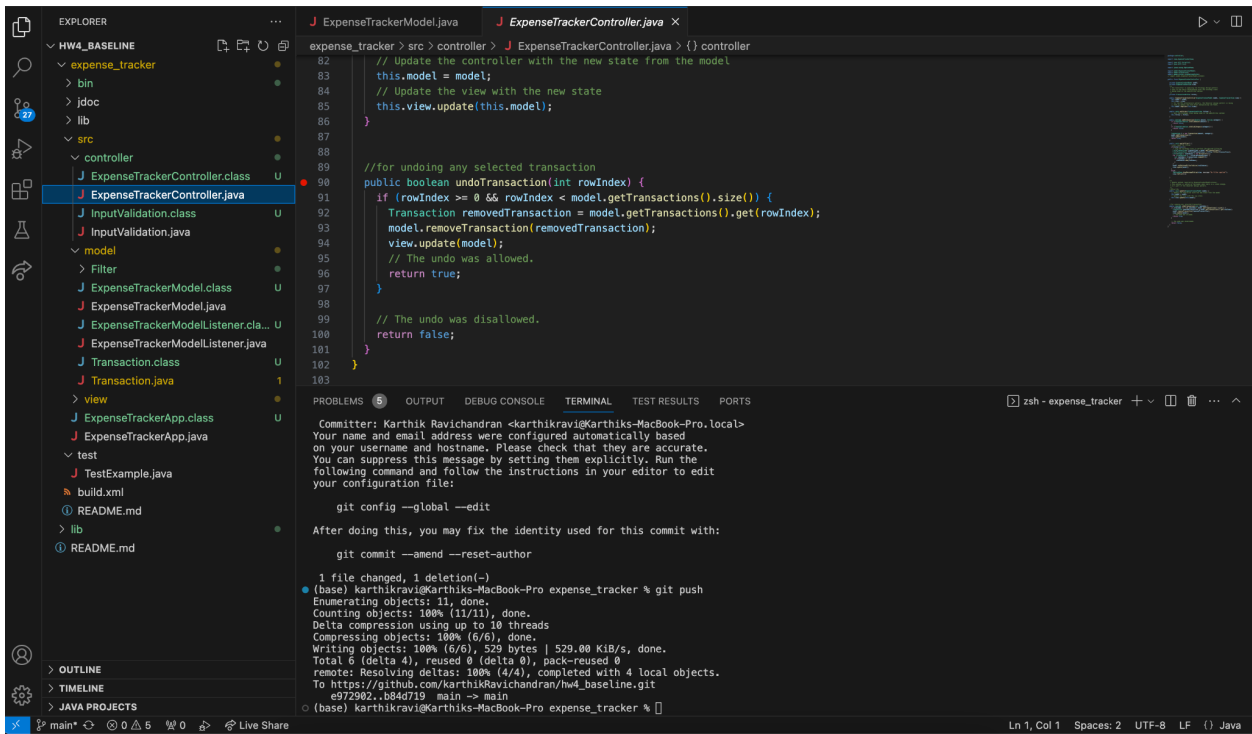


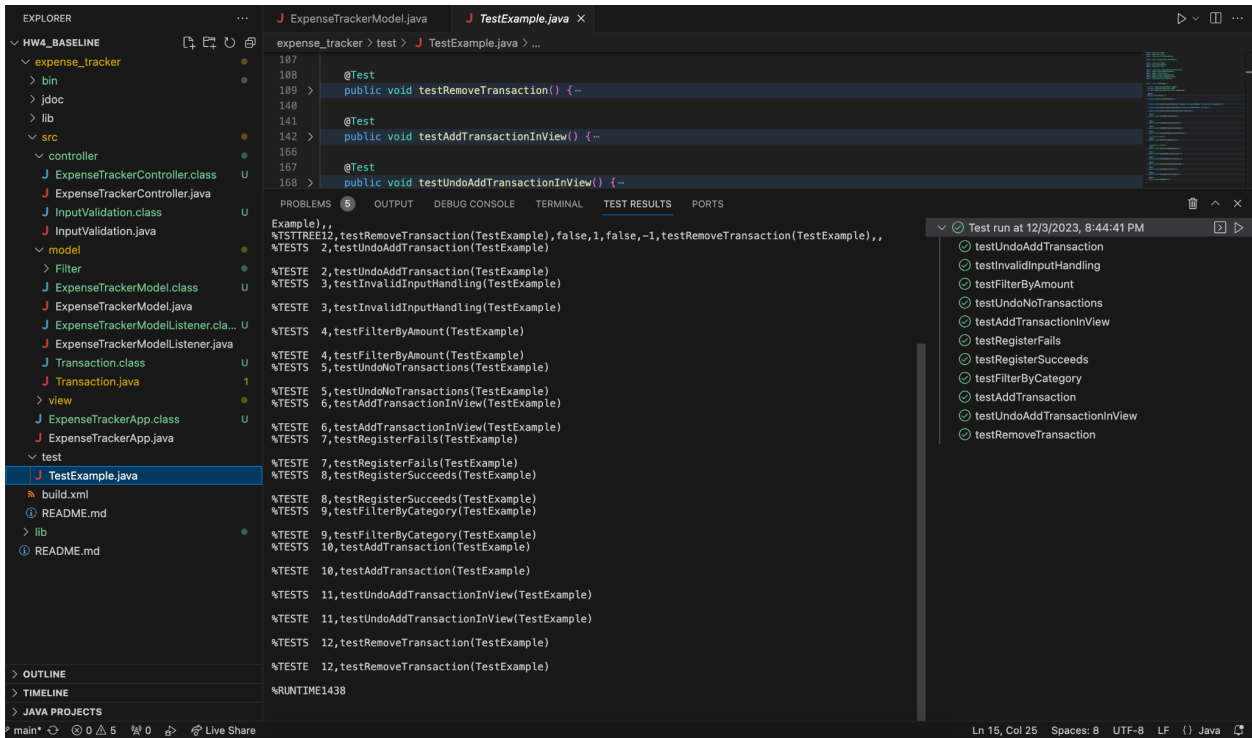
Debuggability:

Here are the five (5) screenshots that show the debugging status be included:

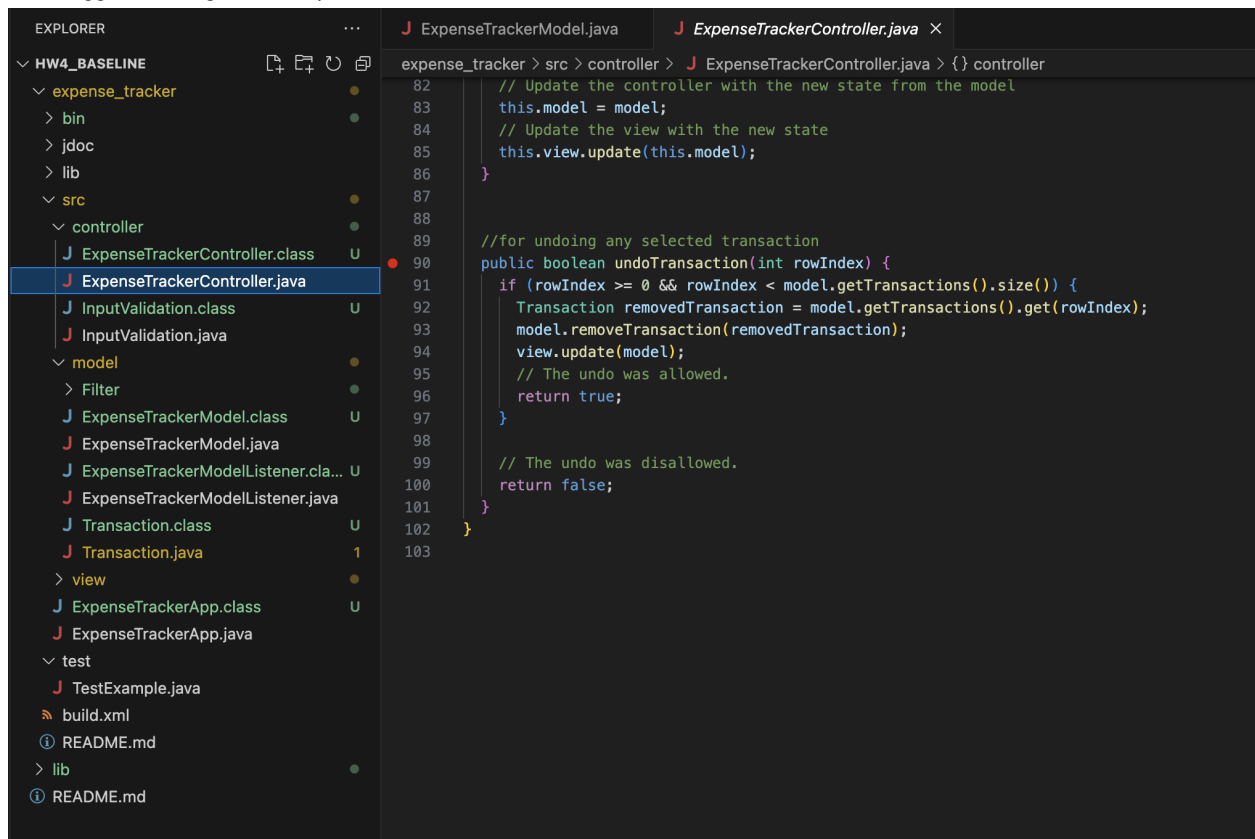
1. *Java API view (e.g., javadoc, class outline)*



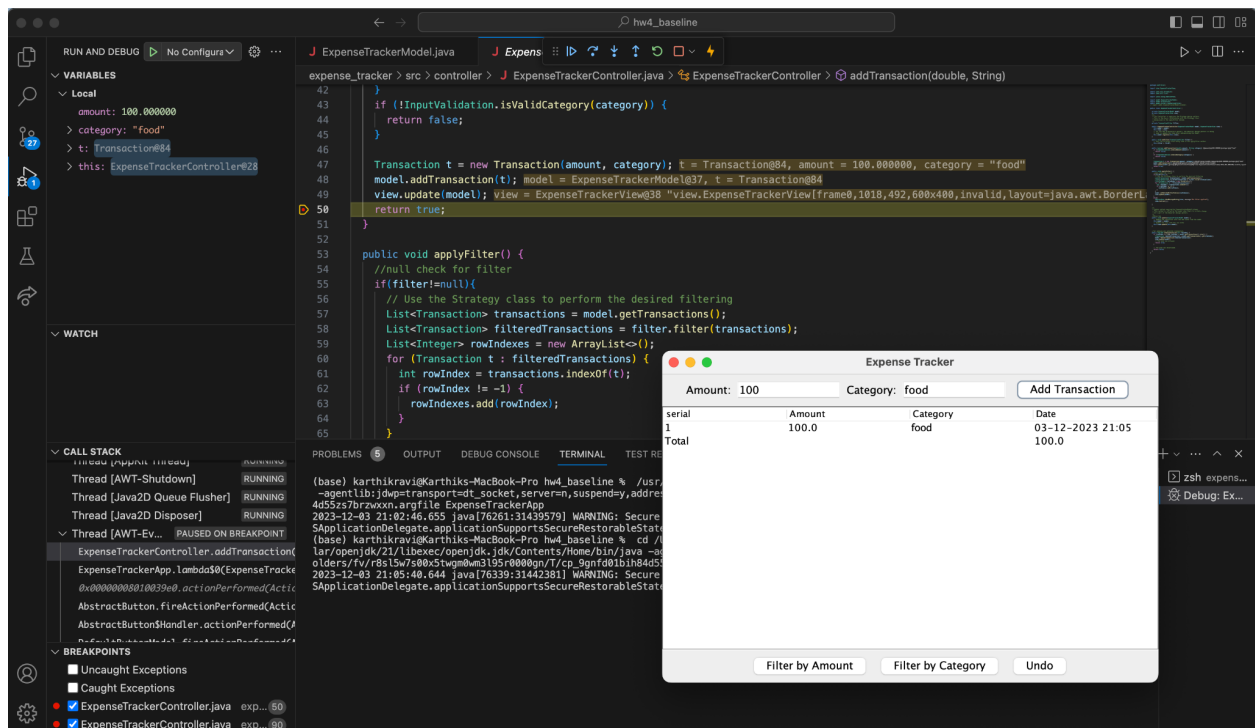
2. JUnit test runner showing all of your test cases passing



3. Debugger showing the breakpoint for the undo



4. Debugger showing the program execution state (usually the variables view) after calling add transaction but before calling undo (show the model and/or UI widgets set appropriately)



5. Debugger showing the program execution state (usually the variables view) after calling the undo (show the model and/or UI widgets are empty again)

The screenshot shows an IDE with a Java application running in a debugger. The main window, titled "Expense Tracker", displays a form with "Amount: 100" and "Category: food", and an "Add Transaction" button. Below the form is a table with columns "serial", "Amount", "Category", and "Date". The table is currently empty, with a "Total" row showing "0.0". At the bottom of the window are buttons for "Filter by Amount", "Filter by Category", and "Undo".

The IDE's left sidebar shows the "VARIABLES" and "WATCH" panels, which are currently empty. The "CALL STACK" panel shows the following threads running:

- Thread [Signal Dispatcher] RUNNING
- Thread [Notification Thread] RUNNING
- Thread [Common-Cleaner] RUNNING
- Thread [AppKit Thread] RUNNING
- Thread [AWT-Shutdown] RUNNING
- Thread [Java2D Queue Flusher] RUNNING
- Thread [Java2D Disposer] RUNNING
- Thread [AWT-EventQueue-0] RUNNING
- Thread [TimerQueue] RUNNING
- Thread [DestroyJavaVM] RUNNING

The "BREAKPOINTS" panel shows three breakpoints set in the code:

- ExpenseTrackerApp.java expense... 73
- ExpenseTrackerController.java exp... 50
- ExpenseTrackerController.java exp... 90

The main editor shows the source code of the application, with the following code visible:

```
expense_tracker > src > J ExpenseTrackerApp.java > ExpenseTrackerApp > main(String[])
    controller.applyFilter();
}
}
} catch (IllegalArgumentException exception) {
    JOptionPane.showMessageDialog(view, exception.getMessage());
    view.toFront();
}
}

// Add action listener to the "Undo" button
view.addUndoButtonListener(e -> {
    int selectedRowIndex = view.getSelectedRowIndex();
    boolean undoStatus = controller.undoTransaction(selectedRowIndex);
    if (undoStatus == false) {
        JOptionPane.showMessageDialog(view, message: "Please select a transaction to undo");
        view.toFront();
    }
});
}
```