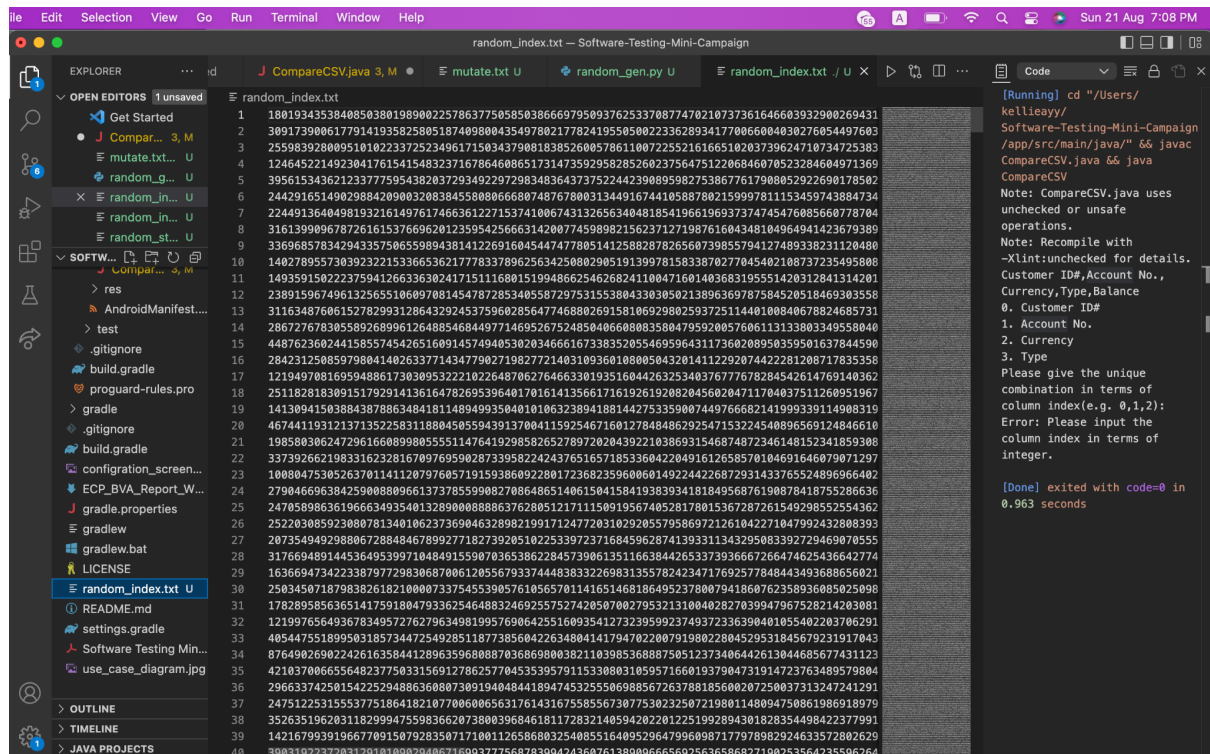


Name: Aw Yong Jia Min Kellie
Student ID: 1005466

Group Member: Zhuoran

Regarding her implementation, her user input for the combination of headers required indexes which were integers. Hence, I decided to modify my fuzzer code in which I generate random long integers instead.

One instance of those random long integers are as shown:



The screenshot shows a code editor with a file named `random_index.txt` open. The file contains a single line of a very long integer value. The terminal on the right shows the output of running a Java program, which includes a note about recompiling with `-Xlint:unchecked` and an error message: `Error: Please input the column index in terms of integer.`

```
1 180193435384085038019890022578637750585038666979509376253370877470210737361646603932900269431
2 309173900617791419358258051874098004354978021778241955050022336289341770066004030276054497603
3 255983920009510102233725234961715034379081838526005786110072255216166510203739624710734725383
4 12465422149230417615415483237167864608651731473592958285260237564751220684607052328464971369
5 39561534362123377595438203729349229729834836437375224423908959827538677617908052922690178502
6 244291651490298842090032083845588086680748780031344016744105887802159997811153459743884734
7 22449136404981932161497617466361227153741006743132656340481854196619693737474547608566078704
8 3161399096787261615376696201235954256631420077459898215623712719876160438104964941423679389
9 369685783429433575065598943814126916045447477805141258828726560739855794127489338231120480
10 2467276783055892689612648054684497714695267524850406600803508479592005760611313380334958040
11 14935915022237594126633630241715116801519948662546268241100471601403683195551420312841314201
12 238915967496132565510609708145478728534055872975315538043303050213896369787584526518469303558
13 311634876061527829931286432924537596028564774688026915810692980259372511440100846678824685731
14 2867276783055892689612648054684497714695267524850406600803508479592005760611313380334958040
15 448726360244158557454263714347790271982772140310936018800584320411292074422281208717835358
16 2842312508597980414026337714347790271982772140310936018800584320411292074422281208717835358
17 1210497081609548861730309532221022648706274663301935160442632540376777678284542614769140362
18 251182818309151792914136164782695396401701614135661716192610208204560284711704037511260951067
19 1413094150388438788634841811480949358401010632389418814427533900744976668214099330114080310
20 46744119312137135258311880450559439137004115925467160127048486292547153224540896560124846610
21 198580862472961660899055511476419209582652789720204392210306931646074872346148152341859308
22 337392662198331623281670976959028733858224243765165718533604220491612658570104691646079071297
23 1008047530526370114112079760571915938609228017236823596006014922426872931433765524896966402
24 27904669638425900880966137378559908270864865140615041584193839418184956876190878418755286636
25 2470589863619666349354813289008888637628051217111509195674052817801536728726154929855254362
26 252203085923080781340186237369046328982199171247720310292055795869721261042271047992432808393
27 20735494298806728848467899277182739205102253397237168459628741393311343295083392729469070555
28 317669489144536495399710484915590783653862284573906135116638442653373936667266474625436642774
29 163669473422338797729830867006120556793374483832269831355782423471769257784843493688656021
30 25034507187983826182100073375455390571605657011416514681864952378007941860762358936085025098
31 178289585636514175918047933625464883246925575274205070027953213100402827039947967528214203081
32 121997656493678349572762796349819957690642551609354127323959923749372336390401055402283706291
33 405447551899503185734472549285118773856042263480414179470220079198022804529531845679231917043
34 876490269022426163584412896359680887036869800382110398536888750512373406442613044685677431123
35 421037006655648237750331569127862577045353957911533080318595716601937904731914723763489579804
36 2966337425015425294886360807309194048441894745325506988567263788185600207250037313247241291
37 2475122531553775186418210770060065631761431675855673077530232507219801148094750861986918979
38 3526351962031182999475598366207380039684618117283214059420220122519928943081828354498422377991
39 142167993919667709150792835071710011265868032540855164095296479960987177978982924633572802629
40 390319223720312910100029406716993777568783994243607613890966656925636586827190253564235596264
```

```
[Running] cd "/Users/kellieay/"
Software-Testing-Mini-Campaign
/app/src/main/java/" && javac
CompareCSV.java && java
CompareCSV
Note: CompareCSV.java uses
unchecked or unsafe
operations.
Note: Recompile with
-Xlint:unchecked for details.
Customer ID#,Account No.,
Currency,Type,Balance
0. Customer ID#
1. Account No.
2. Currency
3. Type
Please give the unique
combination in terms of
column index(e.g. 0,1,2):
Error: Please input the
column index in terms of
integer.
[Done] exited with code=0 in
0.963 seconds
```

I then modified the Scanner used in her code to accept this text file instead. The output from running the code afterwards showed an error message as shown in the image above.

Since her code was working, I also tried to feed the Scanner with random strings instead to monitor its behaviour.

An example of the random strings and the output from running the code are as shown:

Group Member: Namitha

With the intent to check if my current fuzzing code would disrupt the behaviour of code through her user input, I checked through her code to check for the segment on user input so as to insert my text files there. However, I was unable to find any section that asked for user input hence I was not able to proceed with fuzzing here.