

UNIVERSITY OF THE ARMED FORCES – ESPE
WS32 –SOFTWARE PRINCIPLES GROUPS

Name: Joel Alexander Zeas Clavijo

Date: 03th February 2022

NRC: 7490

Career: Telecommunications

TEAM 1:

Salma Villegas

Alexander Ruano

Daniel Lincango

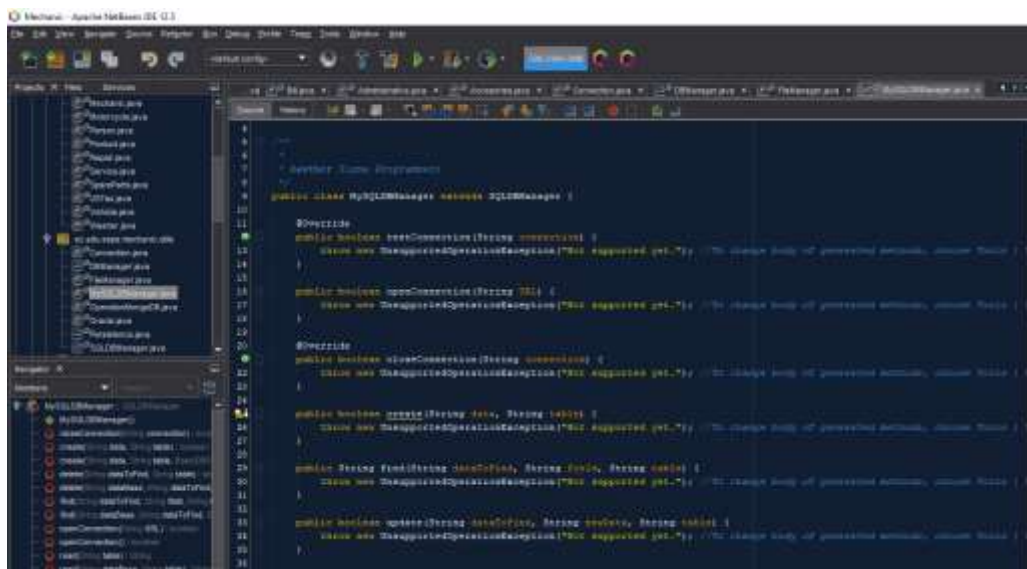
Mateo Maldonado

Leandro Quinga

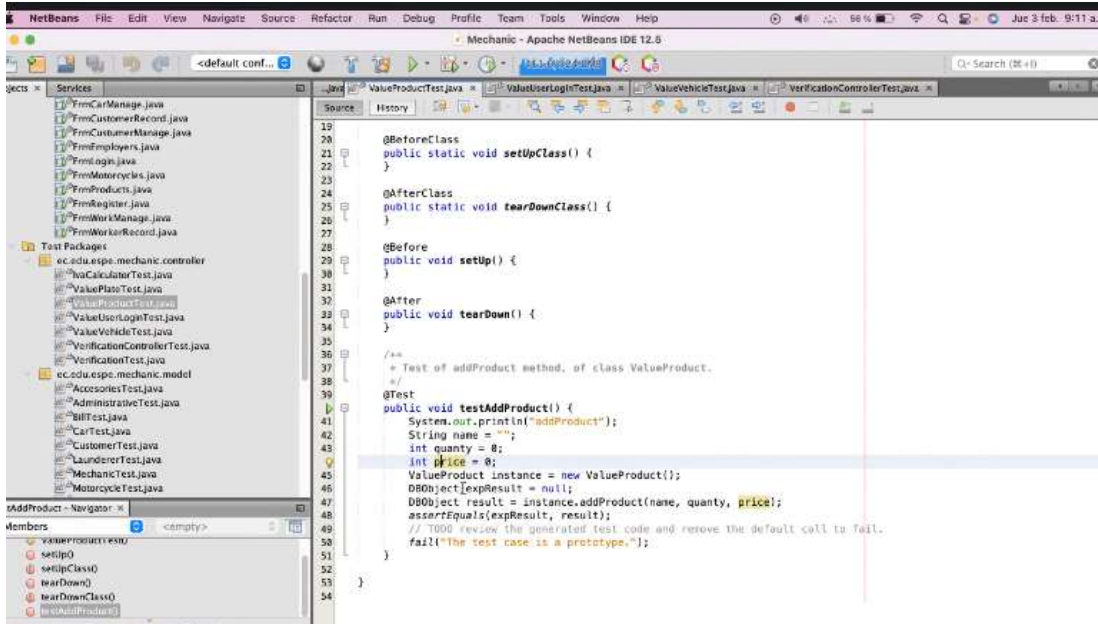
Joel Zeas



Innecesary comments, unclean code



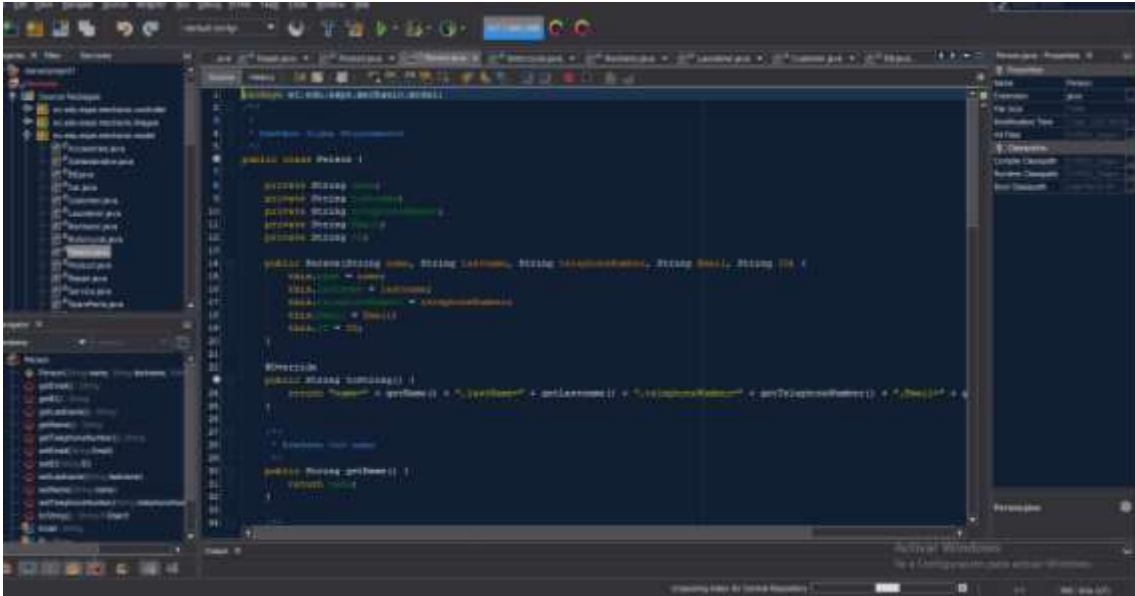
Unclean Code



```

19
20 @BeforeClass
21 public static void setUpClass() {
22 }
23
24 @AfterClass
25 public static void tearDownClass() {
26 }
27
28 @Before
29 public void setUp() {
30 }
31
32 @After
33 public void tearDown() {
34 }
35
36 /**
37  * Test of addProduct method, of class ValueProduct.
38  */
39 @Test
40 public void testAddProduct() {
41     System.out.println("addProduct");
42     String name = "";
43     int quantity = 8;
44     int price = 0;
45     ValueProduct instance = new ValueProduct();
46     DBObject expectedResult = null;
47     DBObject result = instance.addProduct(name, quantity, price);
48     assertEquals(expectedResult, result);
49     // TODO review the generated test code and remove the default call to fail.
50     fail("The test case is a prototype.");
51 }
52
53
54

```



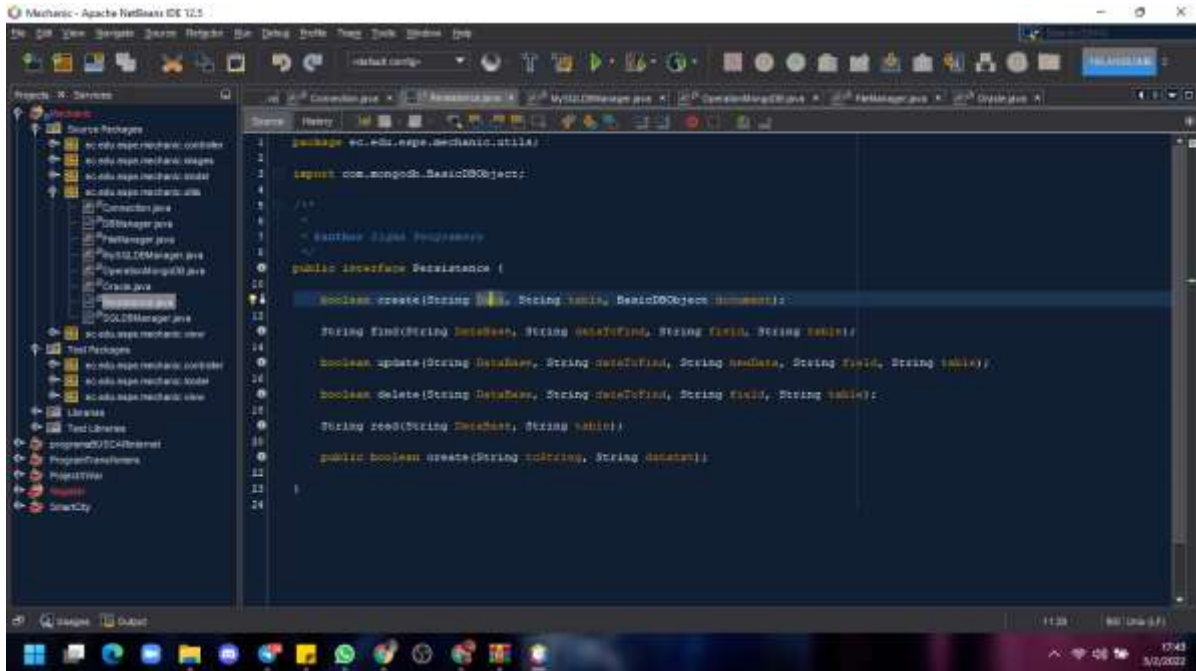
```

1 package ec.edu.espe.mechanic.modelo;
2
3 import java.util.*;
4
5 public class Pelota {
6
7     private String color;
8     private String tamaño;
9     private String material;
10    private String marca;
11    private String precio;
12
13    // Constructor
14    public Pelota(String color, String tamaño, String material, String marca, String precio) {
15        this.color = color;
16        this.tamaño = tamaño;
17        this.material = material;
18        this.marca = marca;
19        this.precio = precio;
20    }
21
22    // Método para mostrar los datos
23    public void mostrarDatos() {
24        System.out.println("Datos de la pelota:");
25        System.out.println("Color: " + this.color + " Tamaño: " + this.tamaño + " Material: " + this.material + " Marca: " + this.marca + " Precio: " + this.precio);
26    }
27
28    // Método para obtener el precio
29    public String getPrecio() {
30        return this.precio;
31    }
32
33 }

```

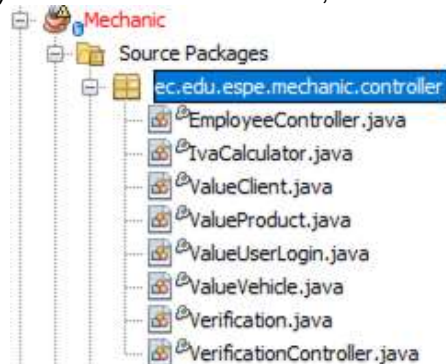
do not use the format Camelcase

Refactor jbutton



Don't use camelcase

- Very redundant class names, difficult to differentiate



- **Single Responsibility Principles:** there is no relationship between the “document” variable and the “addVehicle” class. It is also a very general term.

```
BasicDBObject document = new BasicDBObject();

public BasicDBObject addVehicle(int Name, String LastName, String Phone, String Code) {

    document.put("Name", Name);
    document.put("LastName", LastName);
    document.put("Phone", Phone);
    document.put("Code", Code);

    return null;
}
```

- Use an else variable that replaces the return and returns an actual data

```
public class ValueClient {

    public boolean create(Customer customer) {
        boolean created = false;
        String personData;

        Persistence persistence;

        persistence = new FileManager();

        if (persistence.create(customer.toString(), "Data.txt")) {
            JOptionPane.showMessageDialog(null, customer + "was saved");
        }

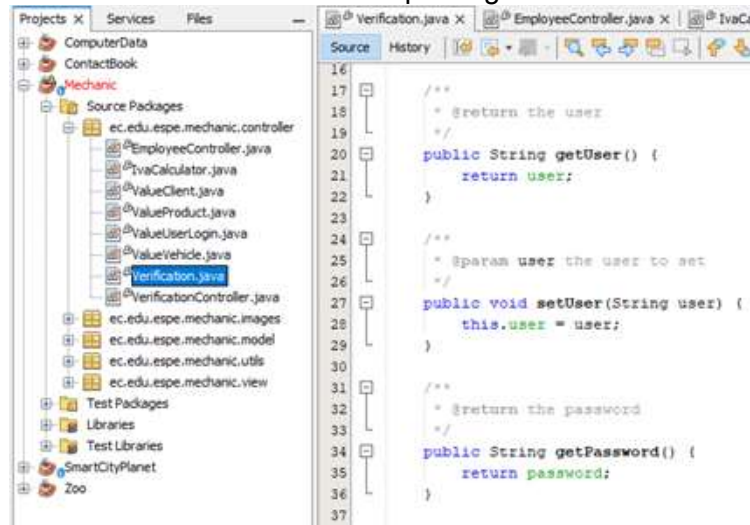
        return created;
    }
}
```

- Use lower case for variables

```
public DBObject addVehicle(int year, String Registration, String Plate, String trademark, String model, float mileage) {

    document.put("Year", year);
    document.put("Registration", Registration);
    document.put("Plate", Plate);
}
```

- Getters and setters must be programmed in the ".model" package, no in the ".controller" package



- **Open/Closed Principle:** Very general variable and method names, also the user validation programming is usually done in the .view package and it must be open to extend it

```
public boolean login(Verification verification, String user, String password) {

    boolean logged = false;
    String readLine;
    Persistence persistence;
    persistence = new FileManager();
    readLine = persistence.read("", "Users.json");

    Gson gson = new Gson();

    verification = gson.fromJson(readLine, Verification.class);

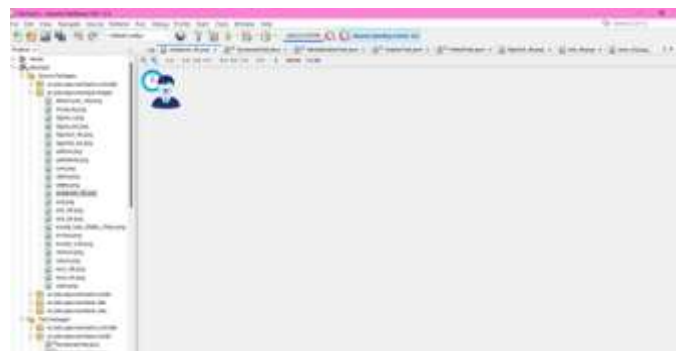
    if (user.isEmpty() || password.isEmpty()) {
        JOptionPane.showMessageDialog(null, "FILL ALL THE FIELDS");
    } else if (user.equals(verification.getUser()) == false && password.equals(verification.getPassword()) == false) {
```


Source Packages

Package: ec.edu.espe.mechanic.images

Nothing to mention, each class has only one image.

Used Principles: Single Responsibility Principle (each class have only one image)

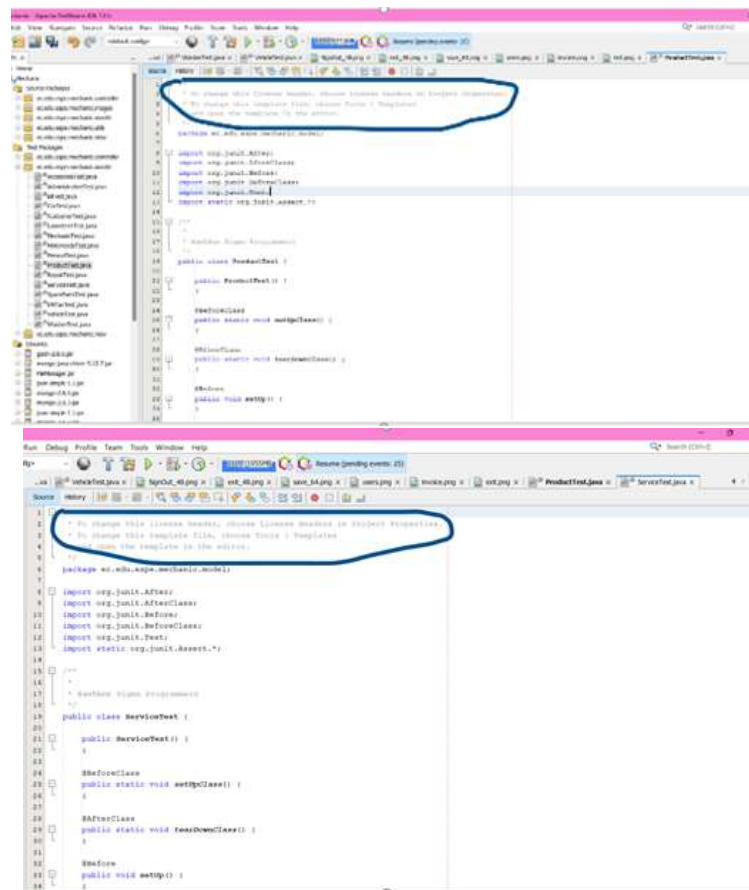


Test Packages

Package: ec.edu.espe.mechanic.model

Unnecessary comments, unclean code,

Used Principles: Single Responsibility Principle (This package is being used for testing only, so each class has a unique reason)





```

1  * To change this license header, please review the project homepage.
2  * To change this template file, choose the type of template
3  * and open the template in the editor.
4
5  package ec.edu.ups.ws.client.demo1;
6
7
8  import org.json.JSONArray;
9  import org.json.JSONObject;
10 import org.json.JSONException;
11 import org.json.JSONObject;
12 import org.json.JSONArray;
13 import org.json.JSONObject;
14
15 import static org.junit.Assert.*;
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
100
```

```

1  // part of testRunner method, of class Test
2
3  // test runner
4
5  public void testResults() {
6      System.out.println("testResults");
7      Roll outcome = roll();
8      inting outcome = 0;
9      inting result = 0;
10     //outcome is initialized by result
11     //roll returns the number of the roll and returns the default value to 0
12     Roll roll = roll();
13
14     // part of testRunner method, of class Test
15
16     // test runner
17
18     public void testResults() {
19         System.out.println("testResults");
20         Roll outcome = roll();
21         inting outcome = 0;
22         inting result = 0;
23         //outcome is initialized by result
24         //roll returns the number of the roll and returns the default value to 0
25         Roll roll = roll();
26
27         // part of testRunner method, of class Test
28
29     }
30
31     public void testResults() {
32         System.out.println("testResults");
33         Roll outcome = roll();
34         inting outcome = 0;
35         inting result = 0;
36         //outcome is initialized by result
37         //roll returns the number of the roll and returns the default value to 0
38         Roll roll = roll();
39
40     }
41
42     // part of testRunner method, of class Test
43
44     // test runner
45
46     public void testResults() {
47         System.out.println("testResults");
48
49     }
50
51     // part of testRunner method, of class Test
52
53     // test runner
54
55     public void testResults() {
56         System.out.println("testResults");
57
58     }
59
60     // part of testRunner method, of class Test
61
62     // test runner
63
64     public void testResults() {
65         System.out.println("testResults");
66
67     }
68
69     // part of testRunner method, of class Test
70
71     // test runner
72
73     public void testResults() {
74         System.out.println("testResults");
75
76     }
77
78     // part of testRunner method, of class Test
79
80     // test runner
81
82     public void testResults() {
83         System.out.println("testResults");
84
85     }
86
87     // part of testRunner method, of class Test
88
89     // test runner
90
91     public void testResults() {
92         System.out.println("testResults");
93
94     }
95
96     // part of testRunner method, of class Test
97
98     // test runner
99
100    public void testResults() {
101        System.out.println("testResults");
102
103    }
104
105    // part of testRunner method, of class Test
106
107    // test runner
108
109    public void testResults() {
110        System.out.println("testResults");
111
112    }
113
114    // part of testRunner method, of class Test
115
116    // test runner
117
118    public void testResults() {
119        System.out.println("testResults");
120
121    }
122
123    // part of testRunner method, of class Test
124
125    // test runner
126
127    public void testResults() {
128        System.out.println("testResults");
129
130    }
131
132    // part of testRunner method, of class Test
133
134    // test runner
135
136    public void testResults() {
137        System.out.println("testResults");
138
139    }
140
141    // part of testRunner method, of class Test
142
143    // test runner
144
145    public void testResults() {
146        System.out.println("testResults");
147
148    }
149
150    // part of testRunner method, of class Test
151
152    // test runner
153
154    public void testResults() {
155        System.out.println("testResults");
156
157    }
158
159    // part of testRunner method, of class Test
160
161    // test runner
162
163    public void testResults() {
164        System.out.println("testResults");
165
166    }
167
168    // part of testRunner method, of class Test
169
170    // test runner
171
172    public void testResults() {
173        System.out.println("testResults");
174
175    }
176
177    // part of testRunner method, of class Test
178
179    // test runner
180
181    public void testResults() {
182        System.out.println("testResults");
183
184    }
185
186    // part of testRunner method, of class Test
187
188    // test runner
189
190    public void testResults() {
191        System.out.println("testResults");
192
193    }
194
195    // part of testRunner method, of class Test
196
197    // test runner
198
199    public void testResults() {
200        System.out.println("testResults");
201
202    }
203
204    // part of testRunner method, of class Test
205
206    // test runner
207
208    public void testResults() {
209        System.out.println("testResults");
210
211    }
212
213    // part of testRunner method, of class Test
214
215    // test runner
216
217    public void testResults() {
218        System.out.println("testResults");
219
220    }
221
222    // part of testRunner method, of class Test
223
224    // test runner
225
226    public void testResults() {
227        System.out.println("testResults");
228
229    }
230
231    // part of testRunner method, of class Test
232
233    // test runner
234
235    public void testResults() {
236        System.out.println("testResults");
237
238    }
239
240    // part of testRunner method, of class Test
241
242    // test runner
243
244    public void testResults() {
245        System.out.println("testResults");
246
247    }
248
249    // part of testRunner method, of class Test
250
251    // test runner
252
253    public void testResults() {
254        System.out.println("testResults");
255
256    }
257
258    // part of testRunner method, of class Test
259
260    // test runner
261
262    public void testResults() {
263        System.out.println("testResults");
264
265    }
266
267    // part of testRunner method, of class Test
268
269    // test runner
270
271    public void testResults() {
272        System.out.println("testResults");
273
274    }
275
276    // part of testRunner method, of class Test
277
278    // test runner
279
280    public void testResults() {
281        System.out.println("testResults");
282
283    }
284
285    // part of testRunner method, of class Test
286
287    // test runner
288
289    public void testResults() {
290        System.out.println("testResults");
291
292    }
293
294    // part of testRunner method, of class Test
295
296    // test runner
297
298    public void testResults() {
299        System.out.println("testResults");
300
301    }
302
303    // part of testRunner method, of class Test
304
305    // test runner
306
307    public void testResults() {
308        System.out.println("testResults");
309
310    }
311
312    // part of testRunner method, of class Test
313
314    // test runner
315
316    public void testResults() {
317        System.out.println("testResults");
318
319    }
320
321    // part of testRunner method, of class Test
322
323    // test runner
324
325    public void testResults() {
326        System.out.println("testResults");
327
328    }
329
330    // part of testRunner method, of class Test
331
332    // test runner
333
334    public void testResults() {
335        System.out.println("testResults");
336
337    }
338
339    // part of testRunner method, of class Test
340
341    // test runner
342
343    public void testResults() {
344        System.out.println("testResults");
345
346    }
347
348    // part of testRunner method, of class Test
349
350    // test runner
351
352    public void testResults() {
353        System.out.println("testResults");
354
355    }
356
357    // part of testRunner method, of class Test
358
359    // test runner
360
361    public void testResults() {
362        System.out.println("testResults");
363
364    }
365
366    // part of testRunner method, of class Test
367
368    // test runner
369
370    public void testResults() {
371        System.out.println("testResults");
372
373    }
374
375    // part of testRunner method, of class Test
376
377    // test runner
378
379    public void testResults() {
380        System.out.println("testResults");
381
382    }
383
384    // part of testRunner method, of class Test
385
386    // test runner
387
388    public void testResults() {
389        System.out.println("testResults");
390
391    }
392
393    // part of testRunner method, of class Test
394
395    // test runner
396
397    public void testResults() {
398        System.out.println("testResults");
399
400    }
401
402    // part of testRunner method, of class Test
403
404    // test runner
405
406    public void testResults() {
407        System.out.println("testResults");
408
409    }
410
411    // part of testRunner method, of class Test
412
413    // test runner
414
415    public void testResults() {
416        System.out.println("testResults");
417
418    }
419
420    // part of testRunner method, of class Test
421
422    // test runner
423
424    public void testResults() {
425        System.out.println("testResults");
426
427    }
428
429    // part of testRunner method, of class Test
430
431    // test runner
432
433    public void testResults() {
434        System.out.println("testResults");
435
436    }
437
438    // part of testRunner method, of class Test
439
440    // test runner
441
442    public void testResults() {
443        System.out.println("testResults");
444
445    }
446
447    // part of testRunner method, of class Test
448
449    // test runner
450
451    public void testResults() {
452        System.out.println("testResults");
453
454    }
455
456    // part of testRunner method, of class Test
457
458    // test runner
459
460    public void testResults() {
461        System.out.println("testResults");
462
463    }
464
465    // part of testRunner method, of class Test
466
467    // test runner
468
469    public void testResults() {
470        System.out.println("testResults");
471
472    }
473
474    // part of testRunner method, of class Test
475
476    // test runner
477
478    public void testResults() {
479        System.out.println("testResults");
480
481    }
482
483    // part of testRunner method, of class Test
484
485    // test runner
486
487    public void testResults() {
488        System.out.println("testResults");
489
490    }
491
492    // part of testRunner method, of class Test
493
494    // test runner
495
496    public void testResults() {
497        System.out.println("testResults");
498
499    }
500
501    // part of testRunner method, of class Test
502
503    // test runner
504
505    public void testResults() {
506        System.out.println("testResults");
507
508    }
509
510    // part of testRunner method, of class Test
511
512    // test runner
513
514    public void testResults() {
515        System.out.println("testResults");
516
517    }
518
519    // part of testRunner method, of class Test
520
521    // test runner
522
523    public void testResults() {
524        System.out.println("testResults");
525
526    }
527
528    // part of testRunner method, of class Test
529
530    // test runner
531
532    public void testResults() {
533        System.out.println("testResults");
534
535    }
536
537    // part of testRunner
```

```

10  public void testPolynomial() {
11      System.out.println("Testing");
12      MathField instance = null;
13      String argument = "";
14      String result = instance.evaluatePolynomial(
15          argument);
16      // Print out the result of the polynomial evaluation
17      // and the argument used to evaluate it
18      System.out.println("Result: " + result + " for argument: " + argument);
19  }
20
21  // Part of getPolynomial method, of class MathField
22
23  // test
24  public void testGetPolynomial() {
25      System.out.println("Testing");
26      MathField instance = null;
27      String argument = "";
28      String result = instance.getPolynomial(
29          argument);
30      // Print out the result of the polynomial evaluation
31      // and the argument used to evaluate it
32      System.out.println("Result: " + result + " for argument: " + argument);
33  }
34
35  // Part of setPolynomial method, of class MathField
36
37  // test
38  public void testSetPolynomial() {
39      System.out.println("Testing");
40      String argument = "";
41      MathField instance = null;
42      instance.setPolynomial(argument);
43  }

```