

ネットワークアプリケーション

第5回 アプリケーションプロトコルの設計 (3)

石井 健太郎

(423研究室・オフィスアワー水3限)

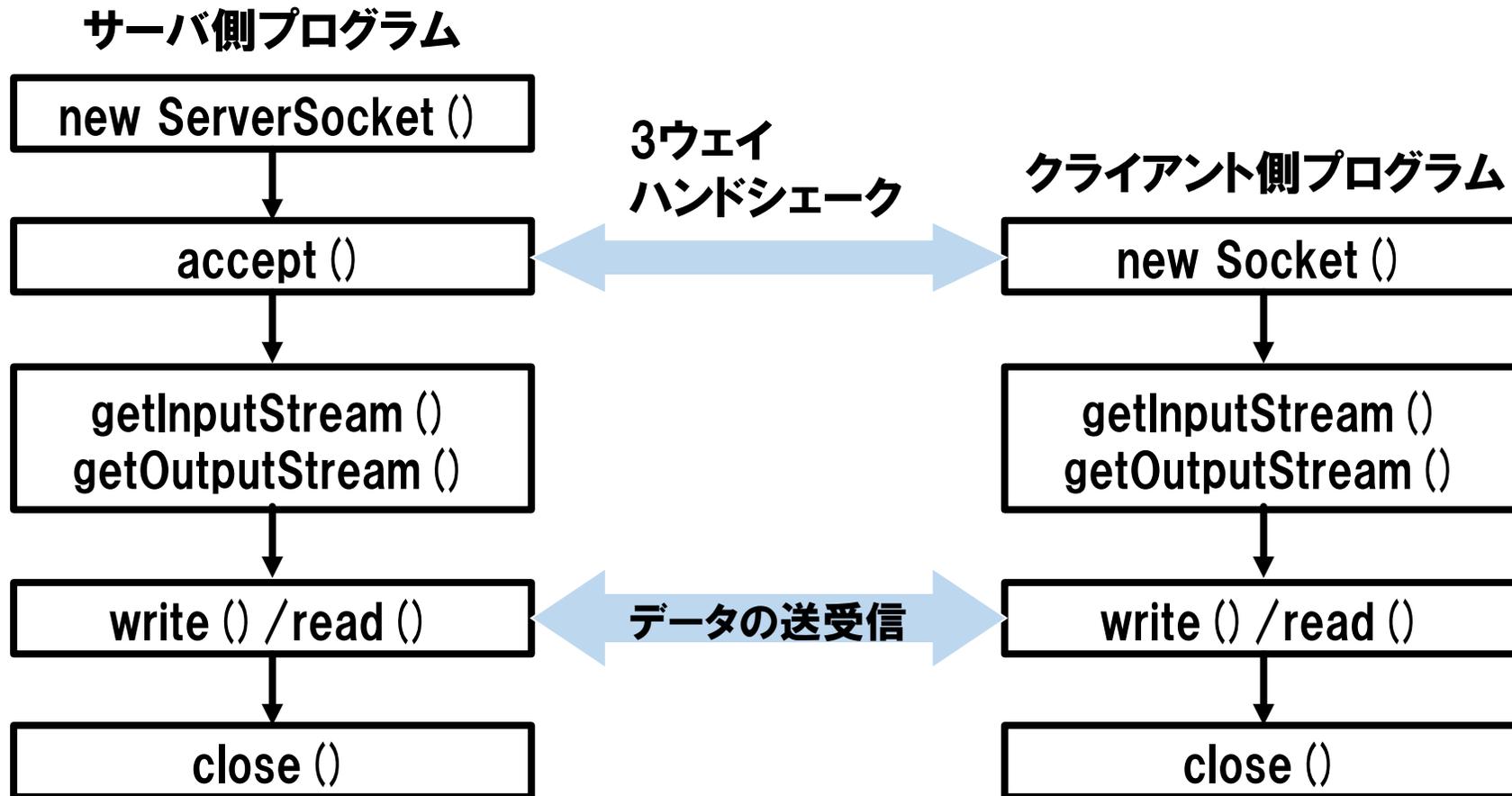
スケジュール

- 9月15日 第1回「TCP/IPプロトコルスイート」
- 9月29日 第2回「ネットワークアプリケーションのプログラミングモデル」
- 10月6日 第3回「アプリケーションプロトコルの設計(1)」
- 10月13日 第4回「アプリケーションプロトコルの設計(2)」
- 10月20日 第5回「アプリケーションプロトコルの設計(3)」 **演習(第3演習室)**
- 10月27日 第6回「アプリケーションプロトコルの設計(4)」 **演習(第3演習室)**
- 11月10日 第7回「サーバサイドウェブプログラミング(1)」
- 11月17日 第8回「サーバサイドウェブプログラミング(2)」

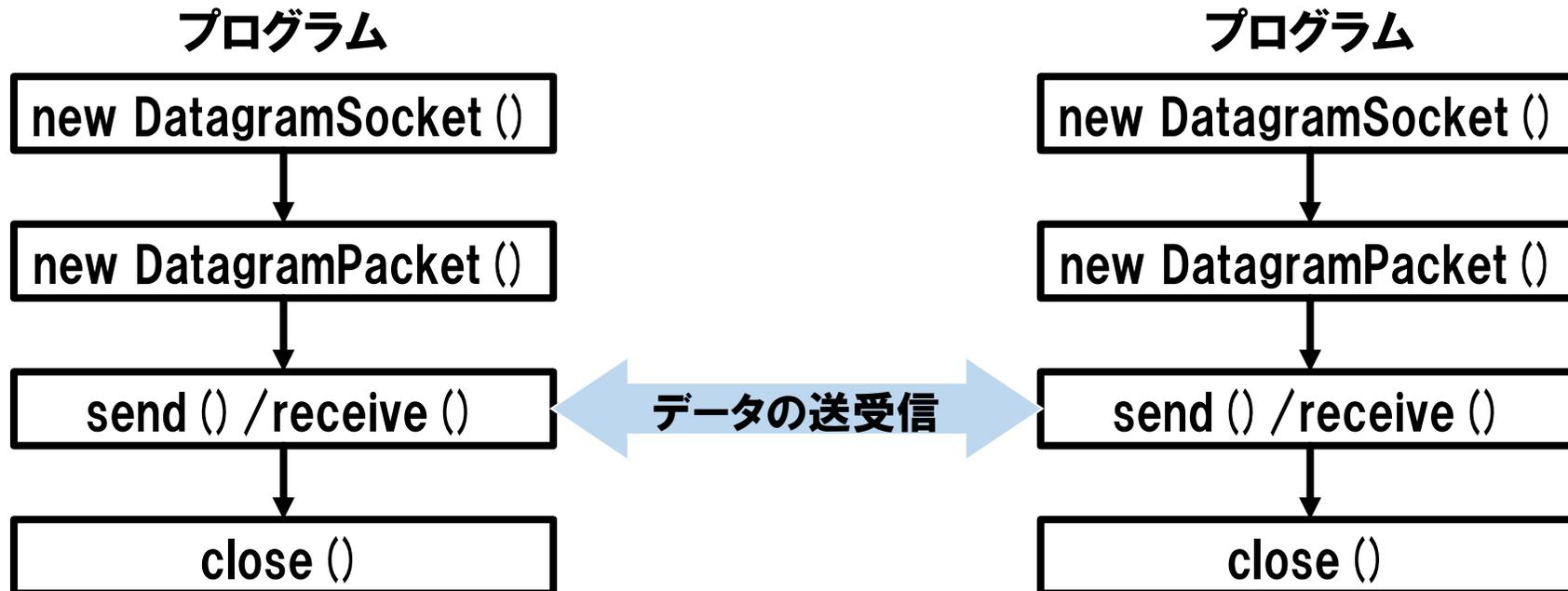
スケジュール

11月24日	第9回「サーバサイドウェブプログラミング(3)」	演習(第3演習室)
12月1日	第10回「サーバサイドウェブプログラミング(4)」	演習(第3演習室)
12月8日	第11回「クライアントサイドウェブプログラミング(1)」	
12月15日	第12回「クライアントサイドウェブプログラミング(2)」	
12月22日	第13回「クライアントサイドウェブプログラミング(3)」	演習(第3演習室)
1月12日	第14回「クライアントサイドウェブプログラミング(4)」	演習(第3演習室)
1月19日	第15回「まとめと演習」	演習(第3演習室)

(参考) TCPによるデータの送受信(Java編)



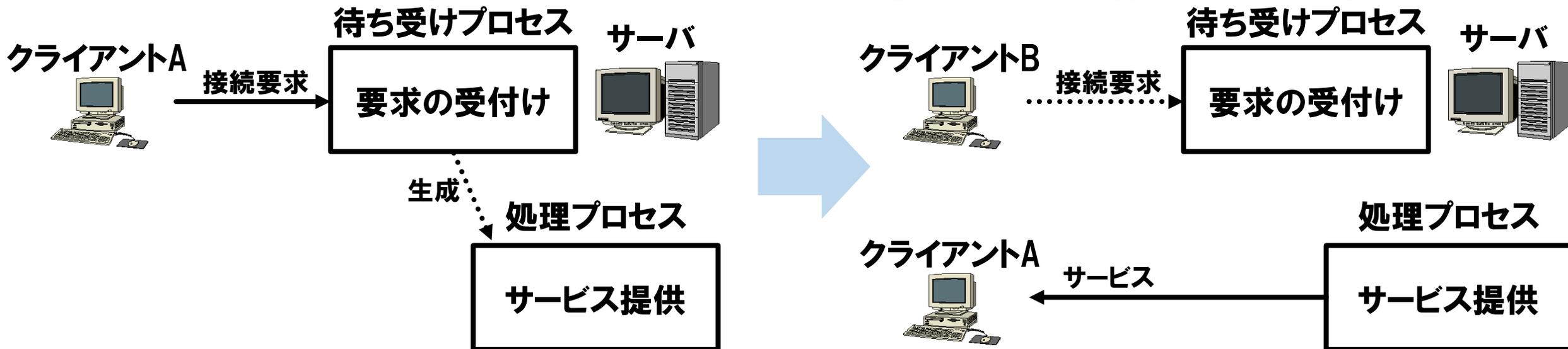
(参考) UDPによるデータの送受信(Java編)



マルチプロセス化・マルチスレッド化

- `accept ()` , `recv ()` , `recvfrom ()` はブロッキングコール
 - あるプログラムの接続・データを待っているとほかの処理を一切行えない

- マルチプロセス化 (fork) ・マルチスレッド化による並行処理が必要



プログラム

- **SimpleTCPTextServerApplication**
 - **SimpleTCPTextServerApplication.java**
 - ウィンドウの描画・キーボード/マウス処理・main関数
 - **SimpleTCPTextServer.java**
 - ソケットの接続待ち受け(サーバソケット)
 - **SimpleTCPTextSocket.java**
 - クライアントとの通信
- **SimpleTCPTextClientApplication**
 - **SimpleTCPTextClientApplication.java**
 - ウィンドウの描画・キーボード/マウス処理・main関数
 - **SimpleTCPTextSocket.java**
 - サーバとの通信

```
Java - SimpleTCPTextServerApplication/src/SimpleTCPTextServerApplication.java - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) プロジェクト(P) 実行(R) ウィンドウ(W) ヘルプ(H)
SimpleTCPTextServerApplication.java  SimpleTCPTextServer.java  SimpleTCPTextSocket.java  SimpleTCPTextClientApplication.java  SimpleTCPTextSocket.java
2 // SimpleTCPTextServerApplication.java
3 import java.awt.BasicStroke;
4
5 // -----
6
7
8
9
10
11 // -----
12
13
14
15
16
17
18
19
20
21 // -----
22 final class SimpleTCPTextServerApplication extends JFrame {
23
24     // -----
25     private static final int INITIAL_WIDTH = 1280;
26     private static final int INITIAL_HEIGHT = 720;
27
28     // -----
29     private static final int SERVER_PORT = 7777;
30
31     // -----
32     private JPanel panel;
33
34     // -----
35     private SimpleTCPTextServer server;
36
37     // -----
38     private int mouseX;
39     private int mouseY;
40
41     private int number0;
42     private int number1;
43     private String string;
44
45     // -----
46     SimpleTCPTextServerApplication(int serverPort) {
47         this.panel = new JPanel() {
48             @Override
49             protected void paintComponent(Graphics g) {
50                 paintPanel((Graphics2D) g);
51             }
52         };
53         this.panel.setBackground(Color.WHITE);
54         this.panel.setBackground(Color.BLACK);
55
56         MouseAdapter mouseAdapter = new MouseAdapter() {
57             @Override
58             public void mouseMoved(MouseEvent me) {
59                 mouseX = me.getX();
60                 mouseY = me.getY();
61             }
62             // implement here
63         };
64         panel.addMouseListener(mouseAdapter);
65
66         @Override
67         public void mousePressed(MouseEvent me) {
68             mouseX = me.getX();
69             mouseY = me.getY();
70             server.sendNumber(mouseX, mouseY);
71             // implement here
72         }
73         panel.addMouseListener(mouseAdapter);
74
75         // -----
76
77     // -----
78     }
79     this.panel.repaint();
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     private synchronized void paintPanel(Graphics2D g) {
151         g.setColor(this.panel.getBackground());
152         g.fillRect(0, 0, this.panel.getWidth(), this.panel.getHeight());
153         g.setStroke(new BasicStroke(3.0f, BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND));
154         g.setColor(this.panel.getForeground());
155         g.drawString("0 + this.number0, this.panel.getWidth() / 2 - 50, this.panel.getHeight() / 2 - 25);
156         g.drawString("1 + this.number1, this.panel.getWidth() / 2 + 50, this.panel.getHeight() / 2 - 25);
157         g.drawString(this.string, this.panel.getWidth() / 2, this.panel.getHeight() / 2 + 25);
158         g.drawString(this.mouseX + ", " + this.mouseY, 25, 25);
159         // implement here
160     }
161
162     // -----
163
164     // -----
165
166     // -----
167
168     // -----
169     public static void main(String args[]) {
170         EventQueue.invokeLater(new Runnable() {
171             @Override
172             public void run() {
173                 try {
174                     SimpleTCPTextServerApplication frame = new SimpleTCPTextServerApplication(SERVER_PORT);
175                     frame.setVisible(true);
176                 } catch (Exception e) {
177                     e.printStackTrace();
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
1005     // -----
1006
1007     // -----
1008
1009     // -----
1010
1011     // -----
1012
1013     // -----
1014
1015     // -----
1016
1017     // -----
1018
1019     // -----
1020
1021     // -----
1022
1023     // -----
1024
1025     // -----
1026
1027     // -----
1028
1029     // -----
1030
1031     // -----
1032
1033     // -----
1034
1035     // -----
1036
1037     // -----
1038
1039     // -----
1040
1041     // -----
1042
1043     // -----
1044
1045     // -----
1046
1047     // -----
1048
1049     // -----
1050
1051     // -----
1052
1053     // -----
1054
1055     // -----
1056
1057     // -----
1058
1059     // -----
1060
1061     // -----
1062
1063     // -----
1064
1065     // -----
1066
1067     // -----
1068
1069     // -----
1070
1071     // -----
1072
1073     // -----
1074
1075     // -----
1076
1077     // -----
1078
1079     // -----
1080
1081     // -----
1082
1083     // -----
1084
1085     // -----
1086
1087     // -----
1088
1089     // -----
1090
1091     // -----
1092
1093     // -----
1094
1095     // -----
1096
1097     // -----
1098
1099     // -----
1100
1101     // -----
1102
1103     // -----
1104
1105     // -----
1106
1107     // -----
1108
1109     // -----
1110
1111     // -----
1112
1113     // -----
1114
1115     // -----
1116
1117     // -----
1118
1119     // -----
1120
1121     // -----
1122
1123     // -----
1124
1125     // -----
1126
1127     // -----
1128
1129     // -----
1130
1131     // -----
1132
1133     // -----
1134
1135     // -----
1136
1137     // -----
1138
1139     // -----
1140
1141     // -----
1142
1143     // -----
1144
1145     // -----
1146
1147     // -----
1148
1149     // -----
1150
1151     // -----
1152
1153     // -----
1154
1155     // -----
1156
1157     // -----
1158
1159     // -----
1160
1161     // -----
1162
1163     // -----
1164
1165     // -----
1166
1167     // -----
1168
1169     // -----
1170
1171     // -----
1172
1173     // -----
1174
1175     // -----
1176
1177     // -----
1178
1179     // -----
1180
1181     // -----
1182
1183     // -----
1184
1185     // -----
1186
1187     // -----
1188
1189     // -----
1190
1191     // -----
1192
1193     // -----
1194
1195     // -----
1196
1197     // -----
1198
1199     // -----
1200
1201     // -----
1202
1203     // -----
1204
1205     // -----
1206
1207     // -----
1208
1209     // -----
1210
1211     // -----
1212
1213     // -----
1214
1215     // -----
1216
1217     // -----
1218
1219     // -----
1220
1221     // -----
1222
1223     // -----
1224
1225     // -----
1226
1227     // -----
1228
1229     // -----
1230
1231     // -----
1232
1233     // -----
1234
1235     // -----
1236
1237     // -----
1238
1239     // -----
1240
1241     // -----
1242
1243     // -----
1244
1245     // -----
1246
1247     // -----
1248
1249     // -----
1250
1251     // -----
1252
1253     // -----
1254
1255     // -----
1256
1257     // -----
1258
1259     // -----
1260
1261     // -----
1262
1263     // -----
1264
1265     // -----
1266
1267     // -----
1268
1269     // -----
1270
1271     // -----
1272
1273     // -----
1274
1275     // -----
1276
1277     // -----
1278
1279     // -----
1280
1281     // -----
1282
1283     // -----
1284
1285     // -----
1286
1287     // -----
1288
1289     // -----
1290
1291     // -----
1292
1293     // -----
1294
1295     // -----
1296
1297     // -----
1298
1299     // -----
1300
1301     // -----
1302
1303     // -----
1304
1305     // -----
1306
1307     // -----
1308
1309     // -----
1310
1311     // -----
1312
1313     // -----
1314
1315     // -----
1316
1317     // -----
1318
1319     // -----
1320
1321     // -----
1322
1323     // -----
1324
1325     // -----
1326
1327     // -----
1328
1329     // -----
1330
1331     // -----
1332
1333     // -----
1334
1335     // -----
1336
1337     // -----
1338
1339     // -----
1340
1341     // -----
1342
1343     // -----
1344
1345     // -----
1346
1347     // -----
1348
1349     // -----
1350
1351     // -----
1352
1353     // -----
1354
1355     // -----
1356
1357     // -----
1358
1359     // -----
1360
1361     // -----
1362
1363     // -----
1364
1365     // -----
1366
1367     // -----
1368
1369     // -----
1370
1371     // -----
1372
1373     // -----
1374
1375     // -----
1376
1377     // -----
1378
1379     // -----
1380
1381     // -----
1382
1383     // -----
1384
1385     // -----
1386
1387     // -----
1388
1389     // -----
1390
1391     // -----
1392
1393     // -----
1394
1395     // -----
1396
1397     // -----
1398
1399     // -----
1400
1401     // -----
1402
1403     // -----
1404
1405     // -----
1406
1407     // -----
1408
1409     // -----
1410
1411     // -----
1412
1413     // -----
1414
1415     // -----
1416
1417     // -----
1418
1419     // -----
1420
1421     // -----
1422
1423     // -----
1424
1425     // -----
1426
1427     // -----
1428
1429     // -----
1430
1431     // -----
1432
1433     // -----
1434
1435     // -----
1436
1437     // -----
1438
1439     // -----
1440
1441     // -----
1442
1443     // -----
1444
1445     // -----
1446
1447     // -----
1448
1449     // -----
1450
1451     // -----
1452
1453     // -----
1454
1455     // -----
1456
1457     // -----
1458
1459     // -----
1460
1461     // -----
1462
1463     // -----
1464
1465     // -----
1466
1467     // -----
1468
1469     // -----
1470
1471     // -----
1472
1473     // -----
1474
1475     // -----
1476
1477     // -----
1478
1479     // -----
1480
1481     // -----
1482
1483     // -----
1484
1485     // -----
1486
1487     // -----
1488
1489     // -----
1490
1491     // -----
1492
1493     // -----
1494
1495     // -----
1496
1497     // -----
1498
1499     // -----
1500
1501     // -----
1502
1503     // -----
1504
1505     // -----
1506
1507     // -----
1508
1509     // -----
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
21
```

はじめに

- クラスライブラリをいろいろ参照しながら理解する必要がある
 - <http://docs.oracle.com/javase/jp/8/api/>

SimpleTCPTextServerApplication.java

```
SimpleTCPTextServerApplication.java | SimpleTCPTTextServer.java | SimpleTCPTTextSocket.java
44 <
45 > // -----
46 > SimpleTCPTTextServerApplication(int serverPort) {
47 >     this.panel = new JPanel();
48 >     @Override
49 >     protected void paintComponent(Graphics g) {
50 >         paintPanel((Graphics2D)g);
51 >     }
52 > }
53 > this.panel.setForeground(Color.WHITE);
54 > this.panel.setBackground(Color.BLACK);
55 <
56 > MouseAdapter mouseAdapter = new MouseAdapter() {
57 >     @Override
58 >     public void mouseMoved(MouseEvent me) {
59 >         mouseX = me.getX();
60 >         mouseY = me.getY();
61 <
62 >         // implement here
63 <
64 >         panel.repaint();
65 >     }
66 <
67 >     @Override
68 >     public void mousePressed(MouseEvent me) {
69 >         mouseX = me.getX();
70 >         mouseY = me.getY();
71 >         server.sendNumbers(mouseX, mouseY);
72 <
73 >         // implement here
74 <
75 >         panel.repaint();
76 >     }
77 > };
78 > this.panel.addMouseListener(mouseAdapter);
79 > this.panel.addMouseMotionListener(mouseAdapter);
on
```

画面の描画

マウスの処理

mouseMoved () は
動かしたときの処理

mousePressed () は
クリックしたときの処理

```
80 <
81 >     this.server = new SimpleTCPTTextServer(this, serverPort);
82 >     this.server.start();
83 <
84 >     this.number0 = -1;
85 >     this.number1 = -1;
86 >     this.string = "no data";
87 <
88 >     setTitle(getClass().getName());
89 >     setSize(INITIAL_WIDTH + 16, INITIAL_HEIGHT + 39);
90 >     setBackground(Color.BLACK);
91 >     setContentPane(this.panel);
92 <
93 >     addWindowListener(new WindowAdapter() {
94 >         @Override
95 >         public void windowClosing(WindowEvent we) {
96 >             quit();
97 >         }
98 >     });
99 <
100 >     addKeyListener(new KeyAdapter() {
101 >         @Override
102 >         public void keyPressed(KeyEvent ke) {
103 >             switch (ke.getKeyCode()) {
104 >                 case KeyEvent.VK_ESCAPE:
105 >                 case KeyEvent.VK_Q:
106 >                     quit();
107 >                     break;
108 <
109 >                 // implement here
110 <
111 >                 default:
112 >                     break;
113 >             }
114 >         }
115 >     });
116 > }
117 <
118 <
```

キーボードの処理

ke.getKeyCode () で
入力されたキーを
取得できる

SimpleTCPTextServerApplication.java

- **画面の描画**

- 次ページ

- **マウスの処理**

- <http://docs.oracle.com/javase/jp/8/api/java/awt/event/MouseEvent.html>
- <http://docs.oracle.com/javase/jp/8/api/java/awt/event/MouseListener.html>
- <http://docs.oracle.com/javase/jp/8/api/java/awt/event/MouseMotionListener.html>

- **キーボードの処理**

- <http://docs.oracle.com/javase/jp/8/api/java/awt/event/KeyEvent.html>
- <http://docs.oracle.com/javase/jp/8/api/java/awt/event/KeyListener.html>

SimpleTCPTextServerApplication.java

- paintPanel () で実際に何を描画するかを書いている

<http://docs.oracle.com/javase/jp/8/api/java/awt/Graphics.html>

```
// ----- ↵
private synchronized void paintPanel(Graphics2D g) { ↵
> g.setColor(this.panel.getBackground()); ↵
> g.fillRect(0, 0, this.panel.getWidth(), this.panel.getHeight()); ↵
> g.setStroke(new BasicStroke(3.0f, BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND)); ↵
> g.setColor(this.panel.getForeground()); ↵
> g.drawString("" + this.number0, this.panel.getWidth() / 2 - 50, this.panel.getHeight() / 2 - 25); ↵
> g.drawString("" + this.number1, this.panel.getWidth() / 2 + 50, this.panel.getHeight() / 2 - 25); ↵
> g.drawString(this.string, this.panel.getWidth() / 2, this.panel.getHeight() / 2 + 25); ↵
> g.drawString(this.mouseX + ", " + this.mouseY, 25, 25); ↵
> // implement here ↵
} ↵
```

SimpleTCPTextServer.java

- クライアントの待ち受け(`this.serverSocket.accept()`)
- クライアントの相手をするスレッドの作成(`client.start()`)
 - <http://docs.oracle.com/javase/jp/8/api/java/net/ServerSocket.html>
 - <http://docs.oracle.com/javase/jp/8/api/java/lang/Thread.html>

```
// -----  
@Override  
public void run() {  
    try {  
        System.out.println("started");  
        this.serverSocket = new ServerSocket(this.port);  
    } catch (IOException ioe) {  
        ioe.printStackTrace();  
        return;  
    }  
  
    this.enabled = true;  
    while (this.enabled) {  
        try {  
            System.out.println("listening...");  
            SimpleTCPTextSocket client = new SimpleTCPTextSocket(this.application, this.serverSocket.accept());  
            System.out.println("accepted");  
  
            client.start();  
            this.clients.add(client);  
        } catch (IOException ioe) {  
            ioe.printStackTrace();  
        }  
    }  
}
```

SimpleTCPTextServer.java

- ちなみに、複数クライアントを管理しており、すべてのクライアントにデータを送信する関数を用意されている
 - ので、自信のある人は、複数クライアントに挑戦するとよい

```
// ----- ↵
synchronized void sendNumbers(int numbers0, int numbers1) { ↵
>   for (SimpleTCPTextSocket client : this.clients) ↵
>     client.sendNumbers(numbers0, numbers1); ↵
} ↵

// ----- ↵
synchronized void sendString(String string) { ↵
>   for (SimpleTCPTextSocket client : this.clients) ↵
>     client.sendString(string); ↵
} ↵

// ----- ↵
synchronized void sendHoge() { ↵
>   for (SimpleTCPTextSocket client : this.clients) ↵
>     client.sendHoge(); ↵
} ↵
```


SimpleTCPTextSocket.java

- run () で行われているのが受信
 - this.reader.readLine ()

```
// -----  
@Override  
public void run() {  
    > if (this.socket == null || this.reader == null || this.writer == null) {  
    >     > return;  
    > }  
  
    > this.enabled = true;  
  
    > try {  
    >     > while (this.enabled) {  
    >     >     > String line = this.reader.readLine();  
    >     >     > if (line == null) {  
    >     >     >     > break;  
    >     >     > }  
  
    >     >     > StringTokenizer tokens = new StringTokenizer(line);  
    >     >     > if (!tokens.hasMoreTokens()) {  
    >     >     >     > continue;  
    >     >     > }  
  
    >     >     > String command = tokens.nextToken();  
    >     >     > if (command.equals("Numbers")) {  
    >     >     >     > if (tokens.countTokens() < 2) {  
    >     >     >     >     > continue;  
    >     >     >     > }  
    >     >     >     > int number0 = Integer.parseInt(tokens.nextToken());  
    >     >     >     > int number1 = Integer.parseInt(tokens.nextToken());  
  
    >     >     >     > this.application.setNumbers(number0, number1);  
    >     >     >     > sendString("received! " + line + " ");  
    >     >     > } else if (command.equals("String")) {  
    >     >     >     > if (tokens.countTokens() < 1) {  
    >     >     >     >     > continue;  
    >     >     >     > }  
    >     >     >     > String string = line.substring("String ".length());  
  
    >     >     >     > this.application.setString(string);  
    >     >     > } else if (command.equals("Hoge")) {  
    >     >     >     > // implement here  
  
    >     >     >     > this.application.setHoge();  
    >     >     > } else {  
    >     >     >     > System.err.println("Invalid command! : " + command);  
    >     >     > }  
    >     > }  
    > } catch (IOException ioe) {  
    >     > ioe.printStackTrace();  
    > }  
    > }  
}
```

- <http://docs.oracle.com/javase/jp/8/api/java/io/BufferedReader.html>

SimpleTCPTextSocket.java

- 個別の関数で行われているのが送信
 - `this.writer.println ()`

```
// -----  
synchronized void sendNumbers(int numbers0, int numbers1) {  
>   if (this.writer == null) {  
>     return;  
>   }  
>   String line = "Numbers " + numbers0 + " " + numbers1;  
>   this.writer.println(line);  
}  
  
// -----  
synchronized void sendString(String string) {  
>   if (this.writer == null) {  
>     return;  
>   }  
>   String line = "String " + string;  
>   this.writer.println(line);  
}  
  
// -----  
synchronized void sendHoge() {  
>   if (this.writer == null) {  
>     return;  
>   }  
>   String line = "Hoge ";  
>   this.writer.println(line);  
}
```

- <http://docs.oracle.com/javase/jp/8/api/java/io/PrintWriter.html>

SimpleTCPTextClientApplication.java

- (このプログラムでは,)基本的にはサーバと同じ動作
 - 次ページ以降

SimpleTCPTextClientApplication.java

```
// -----  
SimpleTCPTextClientApplication(String serverHost, int serverPort) {  
    this.panel = new JPanel();  
    @Override  
    > protected void paintComponent(Graphics g) {  
    >     paintPanel((Graphics2D) g);  
    > }  
    > this.panel.setForeground(Color.WHITE);  
    > this.panel.setBackground(Color.BLACK);  
    > MouseAdapter mouseAdapter = new MouseAdapter() {  
    >     @Override  
    >     > public void mouseMoved(MouseEvent me) {  
    >         mouseX = me.getX();  
    >         mouseY = me.getY();  
    >         // implement here  
    >         panel.repaint();  
    >     }  
    >     @Override  
    >     > public void mousePressed(MouseEvent me) {  
    >         mouseX = me.getX();  
    >         mouseY = me.getY();  
    >         socket.sendNumbers(mouseX, mouseY);  
    >         // implement here  
    >         panel.repaint();  
    >     }  
    > }  
    > this.panel.addMouseListener(mouseAdapter);  
    > this.panel.addMouseMotionListener(mouseAdapter);  
    > this.socket = new SimpleTCPTextSocket(this, serverHost, serverPort);  
    > this.socket.start();  
}
```

画面の描画

マウスの処理

mouseMoved () は
動かしたときの処理

mousePressed () は
クリックしたときの処理

```
> this.number0 = -1;  
> this.number1 = -1;  
> this.string = "no data";  
    > setTitle(getClass().getName());  
    > setSize(INITIAL_WIDTH + 16, INITIAL_HEIGHT + 39);  
    > setBackground(Color.BLACK);  
    > setContentPane(this.panel);  
    > addWindowListener(new WindowAdapter() {  
    >     @Override  
    >     > public void windowClosing(WindowEvent we) {  
    >         > quit();  
    >     }  
    > });  
    > addKeyListener(new KeyAdapter() {  
    >     @Override  
    >     > public void keyPressed(KeyEvent ke) {  
    >         > switch (ke.getKeyCode()) {  
    >         > case KeyEvent.VK_ESCAPE  
    >         > case KeyEvent.VK_Q  
    >         >         > quit();  
    >         >         > break;  
    >         > // implement here  
    >         > default:  
    >         >         > break;  
    >         > }  
    >     }  
    > });  
}
```

キーボードの処理

ke.getKeyCode () で
入力されたキーを
取得できる

SimpleTCPTextClientApplication.java

- **画面の描画**

- 次ページ

- **マウスの処理**

- <http://docs.oracle.com/javase/jp/8/api/java/awt/event/MouseEvent.html>
- <http://docs.oracle.com/javase/jp/8/api/java/awt/event/MouseListener.html>
- <http://docs.oracle.com/javase/jp/8/api/java/awt/event/MouseMotionListener.html>

- **キーボードの処理**

- <http://docs.oracle.com/javase/jp/8/api/java/awt/event/KeyEvent.html>
- <http://docs.oracle.com/javase/jp/8/api/java/awt/event/KeyListener.html>

SimpleTCPTextClientApplication.java

- paintPanel () で実際に何を描画するかを書いている

<http://docs.oracle.com/javase/jp/8/api/java/awt/Graphics.html>

```
// ----- ↵
private synchronized void paintPanel(Graphics2D g) { ↵
> g.setColor(this.panel.getBackground()); ↵
> g.fillRect(0, 0, this.panel.getWidth(), this.panel.getHeight()); ↵
> g.setStroke(new BasicStroke(3.0f, BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND)); ↵
> g.setColor(this.panel.getForeground()); ↵
> g.drawString("" + this.number0, this.panel.getWidth() / 2 - 50, this.panel.getHeight() / 2 - 25); ↵
> g.drawString("" + this.number1, this.panel.getWidth() / 2 + 50, this.panel.getHeight() / 2 - 25); ↵
> g.drawString(this.string, this.panel.getWidth() / 2, this.panel.getHeight() / 2 + 25); ↵
> g.drawString(this.mouseX + ", " + this.mouseY, 25, 25); ↵
> // implement here ↵
} ↵
```

SimpleTCPTTextSocket.java

- 接続して通信に必要な `BufferedReader` と `PrintWriter` を生成する

- <http://docs.oracle.com/javase/jp/8/api/java/net/Socket.html>
- <http://docs.oracle.com/javase/jp/8/api/java/io/BufferedReader.html>
- <http://docs.oracle.com/javase/jp/8/api/java/io/PrintWriter.html>

```
SimpleTCPTTextSocket(SimpleTCPTTextClientApplication application, Socket socket) {  
>   this.enabled = false;  
>  
>   try {  
>     this.socket = socket;  
>     this.reader = new BufferedReader(new InputStreamReader(this.socket.getInputStream()));  
>     this.writer = new PrintWriter(this.socket.getOutputStream(), true);  
>   } catch (IOException ioe) {  
>     ioe.printStackTrace();  
>  
>     if (this.writer != null) {  
>       this.writer.close();  
>       this.writer = null;  
>     }  
>  
>     if (this.reader != null) {  
>       try {  
>         this.reader.close();  
>       } catch (IOException ioe2) {  
>         ioe2.printStackTrace();  
>       } finally {  
>         this.reader = null;  
>       }  
>     }  
>  
>     if (this.socket != null) {  
>       try {  
>
```

```
SimpleTCPTTextSocket(SimpleTCPTTextClientApplication application, String hostname, int port) {  
>   this.enabled = false;  
>  
>   try {  
>     this.socket = new Socket(hostname, port);  
>     this.reader = new BufferedReader(new InputStreamReader(this.socket.getInputStream()));  
>     this.writer = new PrintWriter(this.socket.getOutputStream(), true);  
>   } catch (IOException ioe) {  
>     ioe.printStackTrace();  
>  
>     if (this.writer != null) {  
>       this.writer.close();  
>       this.writer = null;  
>     }  
>  
>     if (this.reader != null) {  
>       try {  
>         this.reader.close();  
>       } catch (IOException ioe2) {  
>         ioe2.printStackTrace();  
>       } finally {  
>         this.reader = null;  
>       }  
>     }  
>  
>     if (this.socket != null) {  
>       try {  
>
```

クライアント側はこちらを使う

SimpleTCPTextSocket.java

- run () で行われているのが受信
 - this.reader.readLine ()

```
// -----  
@Override  
public void run() {  
    > if (this.socket == null || this.reader == null || this.writer == null) {  
    >     > return;  
    > }  
  
    > this.enabled = true;  
  
    > try {  
    >     > while (this.enabled) {  
    >         > String line = this.reader.readLine();  
    >         > if (line == null) {  
    >             > break;  
    >         > }  
  
    >         > StringTokenizer tokens = new StringTokenizer(line);  
    >         > if (!tokens.hasMoreTokens()) {  
    >             > continue;  
    >         > }  
  
    >         > String command = tokens.nextToken();  
    >         > if (command.equals("Numbers")) {  
    >             > if (tokens.countTokens() < 2) {  
    >                 > continue;  
    >             > }  
    >             > int number0 = Integer.parseInt(tokens.nextToken());  
    >             > int number1 = Integer.parseInt(tokens.nextToken());  
  
    >             > this.application.setNumbers(number0, number1);  
    >             > sendString("received! " + line + " ");  
    >         > } else if (command.equals("String")) {  
    >             > if (tokens.countTokens() < 1) {  
    >                 > continue;  
    >             > }  
    >             > String string = line.substring("String ".length());  
  
    >             > this.application.setString(string);  
    >         > } else if (command.equals("Hoge")) {  
    >             > // implement here  
  
    >             > this.application.setHoge();  
    >         > } else {  
    >             > System.err.println("Invalid command! : " + command);  
    >         > }  
    >     > }  
    > } catch (IOException ioe) {  
    >     > ioe.printStackTrace();  
    > }  
}
```

- <http://docs.oracle.com/javase/jp/8/api/java/io/BufferedReader.html>

SimpleTCPTextSocket.java

- 個別の関数で行われているのが送信
 - `this.writer.println ()`

```
// -----  
synchronized void sendNumbers(int numbers0, int numbers1) {  
>   if (this.writer == null)  
>     return;  
  
>   String line = "Numbers " + numbers0 + " " + numbers1;  
>   this.writer.println(line);  
}  
  
// -----  
synchronized void sendString(String string) {  
>   if (this.writer == null)  
>     return;  
  
>   String line = "String " + string;  
>   this.writer.println(line);  
}  
  
// -----  
synchronized void sendHoge() {  
>   if (this.writer == null)  
>     return;  
  
>   String line = "Hoge ";  
>   this.writer.println(line);  
}
```

- <http://docs.oracle.com/javase/jp/8/api/java/io/PrintWriter.html>

プログラム

- **SimpleTCPBinaryServerApplication**
 - SimpleTCPBinaryServerApplication.java
 - ウィンドウの描画・キーボード/マウス処理・main関数
 - SimpleTCPBinaryServer.java
 - ソケットの接続待ち受け(サーバソケット)
 - SimpleTCPBinarySocket.java
 - クライアントとの通信
- **SimpleTCPBinaryClientApplication**
 - SimpleTCPBinaryClientApplication.java
 - ウィンドウの描画・キーボード/マウス処理・main関数
 - SimpleTCPBinarySocket.java
 - サーバとの通信

SimpleBinary {Server,Client} について

- 通信方式以外は SimpleText {Server,Client} と同じ
 - 次ページ以降では異なる部分を説明する

SimpleTCPBinarySocket.java

- 接続して通信に必要な `DataInputStream` と `DataOutputStream` を生成する
 - <http://docs.oracle.com/javase/jp/8/api/java/net/Socket.html>
 - <http://docs.oracle.com/javase/jp/8/api/java/io/DataInputStream.html>
 - <http://docs.oracle.com/javase/jp/8/api/java/io/DataOutputStream.html>

```
SimpleTCPBinarySocket(SimpleTCPBinaryServerApplication application, Socket socket) {  
>   this.enabled = false;  
> }
```

```
> try {  
>   > this.socket = socket;  
>   > this.input = new DataInputStream(this.socket.getInputStream());  
>   > this.output = new DataOutputStream(this.socket.getOutputStream());  
> } catch (IOException ioe) {  
>   > ioe.printStackTrace();  
> }
```

```
> > if (this.output != null) {  
>   >   > try {  
>     >     > this.output.close();  
>     >     > } catch (IOException ioe2) {  
>     >     > ioe2.printStackTrace();  
>     >     > } finally {  
>     >     > this.output = null;  
>     >   }  
> > }
```

サーバ側はこちらを使う

```
> > if (this.input != null) {  
>   >   > try {  
>     >     > this.input.close();  
>     >     > } catch (IOException ioe2) {  
>     >     > ioe2.printStackTrace();  
>     >     > } finally {  
>     >     > this.input = null;  
>     >   }  
> > }
```

```
SimpleTCPBinarySocket(SimpleTCPBinaryServerApplication application, String hostname, int port) {  
>   this.enabled = false;  
> }
```

```
> try {  
>   > this.socket = new Socket(hostname, port);  
>   > this.input = new DataInputStream(this.socket.getInputStream());  
>   > this.output = new DataOutputStream(this.socket.getOutputStream());  
> } catch (IOException ioe) {  
>   > ioe.printStackTrace();  
> }
```

```
> > if (this.output != null) {  
>   >   > try {  
>     >     > this.output.close();  
>     >     > } catch (IOException ioe2) {  
>     >     > ioe2.printStackTrace();  
>     >     > } finally {  
>     >     > this.output = null;  
>     >   }  
> > }
```

クライアント側はこちらを使う

```
> > if (this.input != null) {  
>   >   > try {  
>     >     > this.input.close();  
>     >     > } catch (IOException ioe2) {  
>     >     > ioe2.printStackTrace();  
>     >     > } finally {  
>     >     > this.input = null;  
>     >   }  
> > }
```

SimpleTCPBinarySocket.java

- run () で行われているのが受信
 - this.input.readChar ()

```
// -----  
@Override  
public void run() {  
    > if (this.socket == null || this.input == null || this.output == null) {  
    >     > return;  
    > }  
  
    > this.enabled = true;  
  
    > try {  
    >     > while (this.enabled) {  
    >     >     > char command = this.input.readChar();  
    >     >     > if (command == 'N') {  
    >     >     >     > //implement here  
    >     >     > } else if (command == 'S') {  
    >     >     >     > //implement here  
    >     >     > } else if (command == 'H') {  
    >     >     >     > // implement here  
    >     >     > }  
    >     >     > > this.application.setHoge();  
    >     >     > } else {  
    >     >     >     > System.err.println("Invalid command! : " + command);  
    >     >     > }  
    >     > }  
    > } catch (IOException ioe) {  
    >     > ioe.printStackTrace();  
    > }  
} }
```

- <http://docs.oracle.com/javase/jp/8/api/java/io/DataInputStream.html>

SimpleTCPBinarySocket.java

- 個別の関数で行われているのが送信

- `this.output.writeChar ()`
- `this.output.writeInt ()`
- `this.output.write ()`

```
// -----  
synchronized void sendNumbers(int numbers0, int numbers1) {  
> if (this.output == null) {  
> > return; }  
  
> try {  
> > this.output.writeChar('N');  
> > this.output.writeInt(numbers0);  
> > this.output.writeInt(numbers1);  
> } catch (IOException ioe) {  
> > ioe.printStackTrace();  
> }  
}  
  
// -----  
synchronized void sendString(String string) {  
> if (this.output == null) {  
> > return; }  
  
> byte[] bytes = string.getBytes();  
> try {  
> > this.output.writeChar('S');  
> > this.output.writeInt(bytes.length);  
> > this.output.write(bytes);  
> } catch (IOException ioe) {  
> > ioe.printStackTrace();  
> }  
}  
  
// -----  
synchronized void sendHoge() {  
> if (this.output == null) {  
> > return; }  
  
> try {  
> > this.output.writeChar('H');  
> } catch (IOException ioe) {  
> > ioe.printStackTrace();  
> }  
}
```

- <http://docs.oracle.com/javase/jp/8/api/java/io/DataOutputStream.html>

提出課題

- SimpleBinaryServer / SimpleBinaryClient はバイナリデータを送受信することで、マウスの座標・メッセージを交換するプログラム(未完)である
 - サーバ側・クライアント側ともに、送信部分は実装されているが、受信部分は実装されていない
- 受信部分を実装し、SimpleTextServer / SimpleTextClient と同様に動作するように完成させる

- 提出期限: 12月22日(火)深夜まで
- 提出方法
 - サーバ・クライアントともに、1つのフォルダにいれて、<学籍番号>.zip というファイルに圧縮して提出
 - 課題提出用ドライブ(Xドライブ) SimpleTCPBinaryApplication フォルダに置く

提出課題

• ヒント

- SimpleBinarySocket.java の run () で受信部分を記述している
- this.input.readChar () で1文字受信できるが、int型のデータや文字データを受信するにはどうすればよいかをクラスライブラリのドキュメントを読んで考える

```
// -----  
@Override  
public void run() {  
    > if (this.socket == null || this.input == null || this.output == null)  
    >     return;  
  
    > this.enabled = true;  
  
    > try {  
    >     while (this.enabled) {  
    >         > char command = this.input.readChar();  
    >         >         if (command == 'N') {  
    >             >             //implement here  
  
    >             >         } else if (command == 'S') {  
    >                 >             //implement here  
  
    >             >         } else if (command == 'H') {  
    >                 >             // implement here  
  
    >             >         > this.application.setHoge();  
    >             >         } else {  
    >             >         System.err.println("Invalid command! : " + command);  
    >             >         }  
    >         }  
    >     } catch (IOException ioe) {  
    >         >         ioe.printStackTrace();  
    >     }  
    }  
}
```

- <http://docs.oracle.com/javase/jp/8/api/java/io/DataInputStream.html>

最終課題について(一般的な話)

• 選択制

- 2つのテーマの中から好きなものを選んで提出する
 - 独自のアプリケーションプロトコル
 - クライアントサイドウェブプログラミング
- 複数のテーマの課題が提出された場合、最も高い点数のテーマのみ採用する

• 先願優先主義

- 内容に重複があった場合、(特許と同じ)先願優先主義にて採点する
- つまり、同じような内容の課題が提出された場合、あとに提出された課題の評価を減点する

最終課題について(一般的な話)

- **プログラム実行に基づく評価**
 - プログラムのソースコードを提出
 - こちらでコンパイル・設置のうえ実行した結果により点数をつける
 - どんなに設計がよくできていても機能が充実していても、実行できない場合点数がつきません。
ただし、実行できない場合はメールにて連絡するようにします。
 - プログラムの起動方法・必要な外部ライブラリ・設定ファイルの配置方法など、プログラムを実行するために必要な説明を記した簡単なドキュメントも提出
 - その他任意で、ドキュメント以外の参考資料も提出可
 - 例えば、正しく動作しているときの動画などもOK
- **最終提出期限**
 - 試験期間の当日(その日の夜まで)にする予定(スケジュールが固まってきたら決定)

最終課題について(独自アプリケーション)

- **ソケット通信を行うアプリケーションを考え、通信プロトコルを各自設計の上、それを実装したプログラムを提出せよ**
 - Javaのサンプルプログラムを拡張して画面に何かを表示するプログラムを作成することを推奨する
 - C言語のプログラムでも可とするが、専門演習と全く同じプログラムは評価しない
 - 2人1組でサーバクライアントプログラムを書くことを可とする
 - ただし、プログラム同士連携のために繰り返し修正が必要なので、おすすめしない
- **提出方法**
 - ソースコード・実行するための説明を記したドキュメント・必要なライブラリ・参考資料を、すべて1つのフォルダにいれて、〈学籍番号〉.zip というファイルに圧縮して提出
 - 課題提出用ドライブ(Xドライブ) 最終課題(独自アプリケーション) フォルダに置く

- 来週10月27日(火)も**第3演習室**に集まってください