

# 40009 ExerciseTypes.PPT11

## Kotlin Journey Planning

### Submitters

---

**anb122**

Adithya Narayanan

# Emarking

Final Tests TestSummary.txt: 1/1 Adithya Narayanan - anb122:j1:24

```
1: Final Tests: Summary for anb122 of j1
2: PPT 24
3: -----
4:
5:   Public Tests:
6:     Compiles:    1 / 1
7:     Tests Pass:  1 / 1
8:     Style Check: 1 / 1 ✓
9:
10: Git Repo: git@gitlab.doc.ic.ac.uk:lab2324_autumn/kotlinjourneyplanner_anb122.git
11: Commit ID: 986a1
```

Tests: 3/4  
Correctness: 3/3  
Quality: 1/3

7/10

- 1 for not add tests to test all the code you've written
- 1 for very long lines. try to stick to the 80 char limit.
- 1 for no comments.

```

1: package journeyplan
2:
3: import java.util.*
4: import kotlin.collections.ArrayList
5:
6: class SubwayMap(var subwaySegmentList: List<Segment>) {
7:     fun routesFrom(origin: Station, destination: Station): List<Route> {
8:         return recursiveHelper(Route(listOf()), origin, destination).sortedBy { ↗
9:             it.duration() }
10:     }
11:     private fun recursiveHelper(routeToPass: Route, origin: Station, destination: ↗
12:         Station): List<Route> {
13:         if (origin == destination) {
14:             return listOf(routeToPass)
15:         }
16:         var nextSegments = this.subwaySegmentList
17:             .filter { x -> x.station1 == origin }
18:             .filter { x -> !(x in routeToPass.segmentList) }
19:
20:         if (nextSegments.size == 0) {
21:             return emptyList()
22:         }
23:
24:         return nextSegments.flatMap { x -> ↗
25:             recursiveHelper(routeToPass.add(Route(listOf(x))), x.station2, destination) }
26:     }
27:
28: class Route(var segmentList: List<Segment>) {
29:     fun add(route2: Route): Route {
30:         var arraylistcurrentlist = ArrayList<Segment>(segmentList)
31:         var arraylistroutetoadd = ArrayList<Segment>(route2.segmentList)
32:         var listsum = arraylistcurrentlist + arraylistroutetoadd
33:         return Route(listsum.toList())
34:     }
35:
36:     fun listOfLines(): ArrayList<Segment> {
37:         var listOfLines = ArrayList<Segment>()
38:         listOfLines.add((this.segmentList.first()))
39:         for (i in 1..this.segmentList.size - 1) {
40:             if (segmentList[i].lineOfSegment == segmentList[i - 1].lineOfSegment) {
41:                 var segmentToManipulate = Segment(
42:                     listOfLines.last().station1,
43:                     segmentList[i].station2,
44:                     segmentList[i].lineOfSegment,
45:                     listOfLines.last().minutes + segmentList[i].minutes
46:                 )
47:                 listOfLines.removeAt(listOfLines.size - 1)
48:                 listOfLines.add(segmentToManipulate)
49:             } else {
50:                 listOfLines.add(segmentList[i])
51:             }
52:         }
53:         return listOfLines
54:     }
55:
56:     override fun toString(): String {
57:
58:         var listOfLines = listOfLines()
59:
60:         var OutputString =
61:             this.segmentList.first().station1.toString() + " to " + ↗
62:             this.segmentList.last().station2.toString() + " - " + this.duration() + " minutes, " + ↗
63:             this.numChanges() + " changes"

```

*can make this private?*

*- 1 for long lines.*

```

62:
63:     for (i in listOfLines) {
64:         OutputString += "\n - " + i.station1.toString() + " to " + ↗
65:         i.station2.toString() + " by " + i.lineOfSegment.toString()
66:     }
67:     return OutputString
68: }
69:
70: fun numChanges(): Int {
71:     return listOfLines().size - 1
72: }
73:
74: fun duration(): Int {
75:     var count = 0
76:     for (i in this.segmentList) {
77:         count += i.minutes
78:     }
79:     return count
80: }
81: }
82:
83: fun londonUnderground(): SubwayMap {
84:     val northernLine = Line("Northern")
85:     val victoriaLine = Line("Victoria")
86:     val centralLine = Line("Central")
87:
88:     val highgate = Station("Highgate")
89:     val archway = Station("Archway")
90:     val tufnellPark = Station("Tufnell Park")
91:     val kentishTown = Station("Kentish Town")
92:     val camden = Station("Camden Town")
93:     val euston = Station("Euston")
94:     val warrenStreet = Station("Warren Street")
95:     val oxfordCircus = Station("Oxford Circus")
96:     val bondStreet = Station("Bond Street")
97:
98:     return SubwayMap(
99:         listOf(
100:             Segment(highgate, archway, victoriaLine, 4),
101:             Segment(archway, tufnellPark, victoriaLine, 6),
102:             Segment(tufnellPark, kentishTown, northernLine, 8),
103:             Segment(tufnellPark, camden, victoriaLine, 5),
104:             Segment(camden, euston, centralLine, 6),
105:             Segment(camden, warrenStreet, northernLine, 7),
106:             Segment(euston, oxfordCircus, centralLine, 10),
107:             Segment(camden, bondStreet, victoriaLine, 7),
108:
109:             Segment(archway, highgate, victoriaLine, 4),
110:             Segment(tufnellPark, archway, victoriaLine, 6),
111:             Segment(kentishTown, archway, northernLine, 8),
112:             Segment(camden, tufnellPark, victoriaLine, 5),
113:             Segment(euston, camden, centralLine, 6),
114:             Segment(warrenStreet, camden, northernLine, 7),
115:             Segment(oxfordCircus, euston, centralLine, 10),
116:             Segment(bondStreet, camden, victoriaLine, 7)
117:         )
118:     )
119: }
120:
121: // Garbage code
122: // for (i in segmentList) {
123: //     if (i.station1 == origin) {
124: //         routeList.add(Route(listOf(i)))
125: //     }
126: // }

```

Final Tests                      RoutePlanner.kt: 3/3    Adithya Narayanan - anb122:j1:24

```
127: // println(routeList)
128: // if (routeList[0].segmentList[0].station1 == origin && ✓
routeList[0].segmentList[0].station2 == destination) {
129: //     return routeList.toList()
130: // } else {
131: //     return (routeList + routesFrom())
132: // }
133: //
134: // private fun routesFrom2(origin: Station, destination: Station, routesList: ✓
ArrayList<Route>): List<Route> {
135: //     var routeList = ArrayList<Route>()
136: //     var stationList = ArrayList<Station>()
137: //     for (i in routesList) {
138: //         if (i.segmentList)
139: //     }
140: //     for (i: Segment in segmentList){
141: //         if (i.station1 == origin && i.station2 == destination){
142: //             return listOf(Route(listOf(i)))
143: //         } else if (i.station1 == origin && (segmentList.map {x -> x.station2 in ✓
}).isEmpty()){
144: //         }
145: //     }
146: // }
147:
148: fun main() {
149:     var fakeUnderground = londonUnderground()
150:     println(fakeUnderground.routesFrom(Station("Highgate"), Station("Kentish ✓
Town")))
151: }
```

Final Tests                      TravelModel.kt: 1/1    Adithya Narayanan - anb122:j1:24

```
1: package journeyplan
2:
3: class Station(var n: String) {
4:     override fun toString(): String {
5:         return n
6:     } ✓
7: }
8:
9: class Line(var lineName: String) {
10:     override fun toString(): String {
11:         return lineName + " Line"
12:     } ✓
13: }
14:
15: class Segment(var station1: Station, var station2: Station, var lineOfSegment: ✓
Line, var minutes: Int)
```

# Final Tests RoutePlannerTest.kt: 1/2 Adithya Narayanan - anb122:j1:24

```

1: package journeyplan
2:
3: import org.junit.Test
4: import kotlin.test.assertEquals
5:
6: class RoutePlannerTest {
7:
8:     val northernLine = Line("Northern")
9:     val victoriaLine = Line("Victoria")
10:    val centralLine = Line("Central")
11:
12:    val highgate = Station("Highgate")
13:    val archway = Station("Archway")
14:    val tufnellPark = Station("Tufnell Park")
15:    val kentishTown = Station("Kentish Town")
16:    val camden = Station("Camden Town")
17:    val euston = Station("Euston")
18:    val warrenStreet = Station("Warren Street")
19:    val oxfordCircus = Station("Oxford Circus")
20:    val bondStreet = Station("Bond Street")
21:
22:    val tufnellParkToHighgate =
23:        Route(
24:            listOf(
25:                Segment(tufnellPark, archway, northernLine, 3),
26:                Segment(archway, highgate, northernLine, 3)
27:            )
28:        )
29:
30:    val highgateToOxfordCircus =
31:        Route(
32:            listOf(
33:                Segment(highgate, archway, northernLine, 3),
34:                Segment(archway, kentishTown, northernLine, 3),
35:                Segment(kentishTown, camden, northernLine, 3),
36:                Segment(camden, euston, northernLine, 3),
37:                Segment(euston, warrenStreet, victoriaLine, 3),
38:                Segment(warrenStreet, oxfordCircus, victoriaLine, 3)
39:            )
40:        )
41:
42:    val camdenToBondStreet =
43:        Route(
44:            listOf(
45:                Segment(camden, euston, northernLine, 3),
46:                Segment(euston, warrenStreet, victoriaLine, 3),
47:                Segment(warrenStreet, oxfordCircus, victoriaLine, 3),
48:                Segment(oxfordCircus, bondStreet, centralLine, 2)
49:            )
50:        )
51:
52:    @Test
53:    fun `can calculate number of changes`() {
54:        assertEquals(0, tufnellParkToHighgate.numChanges())
55:        assertEquals(1, highgateToOxfordCircus.numChanges())
56:        assertEquals(2, camdenToBondStreet.numChanges())
57:    }
58:
59:    @Test
60:    fun `can calculate total duration`() {
61:        assertEquals(6, tufnellParkToHighgate.duration())
62:        assertEquals(18, highgateToOxfordCircus.duration())
63:        assertEquals(11, camdenToBondStreet.duration())
64:    }
65:
66:    @Test

```

# Final Tests RoutePlannerTest.kt: 2/2 Adithya Narayanan - anb122:j1:24

```

67:    fun `toString omits calling points`() {
68:        assertEquals(
69:            """
70:                Tufnell Park to Highgate - 6 minutes, 0 changes
71:                - Tufnell Park to Highgate by Northern Line
72:            """.trimIndent(),
73:            tufnellParkToHighgate.toString()
74:        )
75:    }
76:
77:    @Test
78:    fun `toString shows changes`() {
79:        assertEquals(
80:            """
81:                Highgate to Oxford Circus - 18 minutes, 1 changes
82:                - Highgate to Euston by Northern Line
83:                - Euston to Oxford Circus by Victoria Line
84:            """.trimIndent(),
85:            highgateToOxfordCircus.toString()
86:        )
87:    }
88: }

```

- Add tests for routes from.

- Add tests for list of lines().

**Final Tests****TravelModelTest.kt: 1/1 Adithya Narayanan - anb122:j1:24**

```
1: package journeyplan
2:
3: import org.junit.Test
4: import kotlin.test.assertEquals
5:
6: class TravelModelTest {
7:
8:     @Test
9:     fun `printing stations shows their names`() {
10:         assertEquals("South Kensington", Station("South Kensington").toString())
11:         assertEquals("Knightsbridge", Station("Knightsbridge").toString())
12:     }
13:
14:     @Test
15:     fun `printing lines shows their names`() {
16:         assertEquals("District Line", Line("District").toString())
17:         assertEquals("Circle Line", Line("Circle").toString())
18:     }
19: }
```

## Final Tests

testResults.txt: 1/1

Adithya Narayanan - anb122:j1:24

```
1: ----- Test Output -----
2: Running LabTS build... (Fri 24 Nov 18:28:25 UTC 2023)
3:
4: Submission summary...
5: You made 6 commits
6:   - 7675a0c PPT completed [4 files changed, 257 insertions, 97 deletions]
7:   - 1799567 Style corrections [2 files changed, 12 insertions, 13 deletions]
8:   - 6ebb700 Style changes [1 file changed, 2 deletions]
9:   - d2f21f1 Style changed [2 files changed, 11 insertions, 2 deletions]
10:  - 0674f14 Return sorted list and style changes [1 file changed, 5 insertions, 11 deletions]
11:  - 986a1f0 Trying to fix style [1 file changed, 1 insertion, 1 deletion]
12:
13: Preparing...
14:
15: BUILD SUCCESSFUL in 681ms
16:
17: Compiling...Path for java installation '/usr/lib/jvm/openjdk-17' (Common Linux Locations) does not contain a java executable
18:
19: BUILD SUCCESSFUL in 10s
20:
21: Running tests...Path for java installation '/usr/lib/jvm/openjdk-17' (Common Linux Locations) does not contain a java executable
22:
23:
24: journeyplan.RoutePlannerTest > toString shows changes PASSED
25:
26: journeyplan.RoutePlannerTest > toString omits calling points PASSED
27:
28: journeyplan.RoutePlannerTest > can calculate total duration PASSED
29:
30: journeyplan.RoutePlannerTest > can calculate number of changes PASSED
31:
32: journeyplan.TravelModelTest > printing lines shows their names PASSED
33:
34: journeyplan.TravelModelTest > printing stations shows their names PASSED
35:
36: BUILD SUCCESSFUL in 1s
37:
38: Checking code style...
39: BUILD SUCCESSFUL in 2s
40: Finished auto test. (Fri 24 Nov 18:29:03 UTC 2023)
41:
42: ----- Test Errors -----
43:
```

**Test Preview****TestSummary.txt: 1/1 Adithya Narayanan - anb122:j1:24**

```
1: Test Preview: Summary for anb122 of j1
2: PPT 24
3: -----
4:
5:   Public Tests:
6:     Compiles:    1 / 1
7:     Tests Pass:  1 / 1
8:     Style Check: 1 / 1
9:
10: Git Repo: git@gitlab.doc.ic.ac.uk:lab2324_autumn/kotlinjourneyplanner_anb122.git
11: Commit ID: 986a1
```



```

1: package journeyplan
2:
3: import java.util.*
4: import kotlin.collections.ArrayList
5:
6: class SubwayMap(var subwaySegmentList: List<Segment>) {
7:     fun routesFrom(origin: Station, destination: Station): List<Route> {
8:         return recursiveHelper(Route(listOf()), origin, destination).sortedBy { ↵
it.duration() }
9:     }
10:
11:     private fun recursiveHelper(routeToPass: Route, origin: Station, destination: ↵
Station): List<Route> {
12:         if (origin == destination) {
13:             return listOf(routeToPass)
14:         }
15:
16:         var nextSegments = this.subwaySegmentList
17:             .filter { x -> x.station1 == origin }
18:             .filter { x -> !(x in routeToPass.segmentList) }
19:
20:         if (nextSegments.size == 0) {
21:             return emptyList()
22:         }
23:
24:         return nextSegments.flatMap { x -> ↵
recursiveHelper(routeToPass.add(Route(listOf(x))), x.station2, destination) }
25:     }
26: }
27:
28: class Route(var segmentList: List<Segment>) {
29:     fun add(route2: Route): Route {
30:         var arraylistcurrentlist = ArrayList<Segment>(segmentList)
31:         var arraylistroutetoadd = ArrayList<Segment>(route2.segmentList)
32:         var listsum = arraylistcurrentlist + arraylistroutetoadd
33:         return Route(listsum.toList())
34:     }
35:
36:     fun listOfLines(): ArrayList<Segment> {
37:         var listOfLines = ArrayList<Segment>()
38:         listOfLines.add((this.segmentList.first()))
39:         for (i in 1..this.segmentList.size - 1) {
40:             if (segmentList[i].lineOfSegment == segmentList[i - 1].lineOfSegment) {
41:                 var segmentToManipulate = Segment(
42:                     listOfLines.last().station1,
43:                     segmentList[i].station2,
44:                     segmentList[i].lineOfSegment,
45:                     listOfLines.last().minutes + segmentList[i].minutes
46:                 )
47:                 listOfLines.removeAt(listOfLines.size - 1)
48:                 listOfLines.add(segmentToManipulate)
49:             } else {
50:                 listOfLines.add(segmentList[i])
51:             }
52:         }
53:         return listOfLines
54:     }
55:
56:     override fun toString(): String {
57:
58:         var listOfLines = listOfLines()
59:
60:         var OutputString =
61:             this.segmentList.first().station1.toString() + " to " + ↵
this.segmentList.last().station2.toString() + " - " + this.duration() + " minutes, " + ↵
this.numChanges() + " changes"

```

```

62:
63:     for (i in listOfLines) {
64:         OutputString += "\n - " + i.station1.toString() + " to " + ↵
i.station2.toString() + " by " + i.lineOfSegment.toString()
65:     }
66:
67:     return OutputString
68: }
69:
70: fun numChanges(): Int {
71:     return listOfLines().size - 1
72: }
73:
74: fun duration(): Int {
75:     var count = 0
76:     for (i in this.segmentList) {
77:         count += i.minutes
78:     }
79:     return count
80: }
81: }
82:
83: fun londonUnderground(): SubwayMap {
84:     val northernLine = Line("Northern")
85:     val victoriaLine = Line("Victoria")
86:     val centralLine = Line("Central")
87:
88:     val highgate = Station("Highgate")
89:     val archway = Station("Archway")
90:     val tufnellPark = Station("Tufnell Park")
91:     val kentishTown = Station("Kentish Town")
92:     val camden = Station("Camden Town")
93:     val euston = Station("Euston")
94:     val warrenStreet = Station("Warren Street")
95:     val oxfordCircus = Station("Oxford Circus")
96:     val bondStreet = Station("Bond Street")
97:
98:     return SubwayMap(
99:         listOf(
100:             Segment(highgate, archway, victoriaLine, 4),
101:             Segment(archway, tufnellPark, victoriaLine, 6),
102:             Segment(tufnellPark, kentishTown, northernLine, 8),
103:             Segment(tufnellPark, camden, victoriaLine, 5),
104:             Segment(camden, euston, centralLine, 6),
105:             Segment(camden, warrenStreet, northernLine, 7),
106:             Segment(euston, oxfordCircus, centralLine, 10),
107:             Segment(camden, bondStreet, victoriaLine, 7),
108:
109:             Segment(archway, highgate, victoriaLine, 4),
110:             Segment(tufnellPark, archway, victoriaLine, 6),
111:             Segment(kentishTown, archway, northernLine, 8),
112:             Segment(camden, tufnellPark, victoriaLine, 5),
113:             Segment(euston, camden, centralLine, 6),
114:             Segment(warrenStreet, camden, northernLine, 7),
115:             Segment(oxfordCircus, euston, centralLine, 10),
116:             Segment(bondStreet, camden, victoriaLine, 7)
117:         )
118:     )
119: }
120:
121: // Garbage code
122: // for (i in segmentList) {
123: //     if (i.station1 == origin) {
124: //         routeList.add(Route(listOf(i)))
125: //     }
126: // }

```

**Test Preview** **RoutePlanner.kt: 3/3 Adithya Narayanan - anb122:j1:24**

```
127: // println(routeList)
128: // if (routeList[0].segmentList[0].station1 == origin && ✓
routeList[0].segmentList[0].station2 == destination) {
129: //     return routeList.toList()
130: // } else {
131: //     return (routeList + routesFrom())
132: // }
133: //
134: // private fun routesFrom2(origin: Station, destination: Station, routesList: ✓
ArrayList<Route>): List<Route> {
135: //     var routeList = ArrayList<Route>()
136: //     var stationList = ArrayList<Station>()
137: //     for (i in routesList) {
138: //         if (i.segmentList)
139: //     }
140: //     for (i: Segment in segmentList){
141: //         if (i.station1 == origin && i.station2 == destination){
142: //             return listOf(Route(listOf(i)))
143: //         } else if (i.station1 == origin && (segmentList.map {x -> x.station2 in ✓
}).isEmpty()){
144: //         }
145: //     }
146: // }
147:
148: fun main() {
149:     var fakeUnderground = londonUnderground()
150:     println(fakeUnderground.routesFrom(Station("Highgate"), Station("Kentish ✓
Town")))
151: }
```

**Test Preview** **TravelModel.kt: 1/1 Adithya Narayanan - anb122:j1:24**

```
1: package journeyplan
2:
3: class Station(var n: String) {
4:     override fun toString(): String {
5:         return n
6:     }
7: }
8:
9: class Line(var lineName: String) {
10:     override fun toString(): String {
11:         return lineName + " Line"
12:     }
13: }
14:
15: class Segment(var station1: Station, var station2: Station, var lineOfSegment: ✓
Line, var minutes: Int)
```

## Test Preview

RoutePlannerTest.kt: 1/2 Adithya Narayanan - anb122:j1:24

```

1: package journeyplan
2:
3: import org.junit.Test
4: import kotlin.test.assertEquals
5:
6: class RoutePlannerTest {
7:
8:     val northernLine = Line("Northern")
9:     val victoriaLine = Line("Victoria")
10:    val centralLine = Line("Central")
11:
12:    val highgate = Station("Highgate")
13:    val archway = Station("Archway")
14:    val tufnellPark = Station("Tufnell Park")
15:    val kentishTown = Station("Kentish Town")
16:    val camden = Station("Camden Town")
17:    val euston = Station("Euston")
18:    val warrenStreet = Station("Warren Street")
19:    val oxfordCircus = Station("Oxford Circus")
20:    val bondStreet = Station("Bond Street")
21:
22:    val tufnellParkToHighgate =
23:        Route(
24:            listOf(
25:                Segment(tufnellPark, archway, northernLine, 3),
26:                Segment(archway, highgate, northernLine, 3)
27:            )
28:        )
29:
30:    val highgateToOxfordCircus =
31:        Route(
32:            listOf(
33:                Segment(highgate, archway, northernLine, 3),
34:                Segment(archway, kentishTown, northernLine, 3),
35:                Segment(kentishTown, camden, northernLine, 3),
36:                Segment(camden, euston, northernLine, 3),
37:                Segment(euston, warrenStreet, victoriaLine, 3),
38:                Segment(warrenStreet, oxfordCircus, victoriaLine, 3)
39:            )
40:        )
41:
42:    val camdenToBondStreet =
43:        Route(
44:            listOf(
45:                Segment(camden, euston, northernLine, 3),
46:                Segment(euston, warrenStreet, victoriaLine, 3),
47:                Segment(warrenStreet, oxfordCircus, victoriaLine, 3),
48:                Segment(oxfordCircus, bondStreet, centralLine, 2)
49:            )
50:        )
51:
52:    @Test
53:    fun `can calculate number of changes`() {
54:        assertEquals(0, tufnellParkToHighgate.numChanges())
55:        assertEquals(1, highgateToOxfordCircus.numChanges())
56:        assertEquals(2, camdenToBondStreet.numChanges())
57:    }
58:
59:    @Test
60:    fun `can calculate total duration`() {
61:        assertEquals(6, tufnellParkToHighgate.duration())
62:        assertEquals(18, highgateToOxfordCircus.duration())
63:        assertEquals(11, camdenToBondStreet.duration())
64:    }
65:
66:    @Test

```

## Test Preview

RoutePlannerTest.kt: 2/2 Adithya Narayanan - anb122:j1:24

```

67:    fun `toString omits calling points`() {
68:        assertEquals(
69:            """
70:                Tufnell Park to Highgate - 6 minutes, 0 changes
71:                - Tufnell Park to Highgate by Northern Line
72:            """.trimIndent(),
73:            tufnellParkToHighgate.toString()
74:        )
75:    }
76:
77:    @Test
78:    fun `toString shows changes`() {
79:        assertEquals(
80:            """
81:                Highgate to Oxford Circus - 18 minutes, 1 changes
82:                - Highgate to Euston by Northern Line
83:                - Euston to Oxford Circus by Victoria Line
84:            """.trimIndent(),
85:            highgateToOxfordCircus.toString()
86:        )
87:    }
88: }

```

```
1: package journeyplan
2:
3: import org.junit.Test
4: import kotlin.test.assertEquals
5:
6: class TravelModelTest {
7:
8:     @Test
9:     fun `printing stations shows their names`() {
10:         assertEquals("South Kensington", Station("South Kensington").toString())
11:         assertEquals("Knightsbridge", Station("Knightsbridge").toString())
12:     }
13:
14:     @Test
15:     fun `printing lines shows their names`() {
16:         assertEquals("District Line", Line("District").toString())
17:         assertEquals("Circle Line", Line("Circle").toString())
18:     }
19: }
```

## Test Preview

testResults.txt: 1/1

Adithya Narayanan - anb122:j1:24

```
1: ----- Test Output -----
2: Running LabTS build... (Fri 24 Nov 18:28:25 UTC 2023)
3:
4: Submission summary...
5: You made 6 commits
6:   - 7675a0c PPT completed [4 files changed, 257 insertions, 97 deletions]
7:   - 1799567 Style corrections [2 files changed, 12 insertions, 13 deletions]
8:   - 6ebb700 Style changes [1 file changed, 2 deletions]
9:   - d2f21f1 Style changed [2 files changed, 11 insertions, 2 deletions]
10:  - 0674f14 Return sorted list and style changes [1 file changed, 5 insertions, 11 deletions]
11:  - 986a1f0 Trying to fix style [1 file changed, 1 insertion, 1 deletion]
12:
13: Preparing...
14:
15: BUILD SUCCESSFUL in 681ms
16:
17: Compiling...Path for java installation '/usr/lib/jvm/openjdk-17' (Common Linux Locations) does not contain a java executable
18:
19: BUILD SUCCESSFUL in 10s
20:
21: Running tests...Path for java installation '/usr/lib/jvm/openjdk-17' (Common Linux Locations) does not contain a java executable
22:
23:
24: journeyplan.RoutePlannerTest > toString shows changes PASSED
25:
26: journeyplan.RoutePlannerTest > toString omits calling points PASSED
27:
28: journeyplan.RoutePlannerTest > can calculate total duration PASSED
29:
30: journeyplan.RoutePlannerTest > can calculate number of changes PASSED
31:
32: journeyplan.TravelModelTest > printing lines shows their names PASSED
33:
34: journeyplan.TravelModelTest > printing stations shows their names PASSED
35:
36: BUILD SUCCESSFUL in 1s
37:
38: Checking code style...
39: BUILD SUCCESSFUL in 2s
40: Finished auto test. (Fri 24 Nov 18:29:03 UTC 2023)
41:
42: ----- Test Errors -----
43:
```