

Section A

Q1	10/10
Q2	6 /6
Q3	5 /5
Q4	8 /8
Q5	15 /15
Q6	6 /6

Compilation penalty

Style penalty (capped at -3)



Total for Section A

50 / 50

Section A**Tests (1/1)****jdk21**

Username: jlk21

Compilation: 1 / 1

Model Answer's Tests - Question1Tests: 10 / 10

Model Answer's Tests - Question2Tests: 7 / 7

Model Answer's Tests - Question3Tests: 11 / 11

Model Answer's Tests - Question4Tests: 3 / 3

Model Answer's Tests - Question5Tests: 9 / 9

Model Answer's Tests - Question6Tests: 3 / 3

Model Answer's Tests - GoodPracticesTestsNote: 4 / 4

Model Answer's Tests - GoodPracticesTestsStandardTune: 2 / 2

Model Answer's Tests - GoodPracticesTestsTransposedTune: 2 / 2

Model Answer's Tests - GoodPracticesTestsTune: 1 / 1

No Google style violations - excellent!

Style penalty (capped at -3): 0

Note: if it is below the cap, your total style penalty could be higher if the marker has stylistic concerns that go beyond what Checkstyle identifies automatically.

```
1: package tunes;
2:
3: public final class Note {
4:     private final int pitch;
5:     private final int duration;
6:
7:     private static final int MIN_PITCH = 0;
8:     private static final int MAX_PITCH = 200;
9:
10:    private static final int MIN_DURATION = 1;
11:    private static final int MAX_DURATION = 64;
12:
13:    private static final int NUM_NOTES_PER_OCTAVE = 12;
14:
15:    public Note(int pitch, int duration) {
16:        if (pitch < MIN_PITCH || pitch > MAX_PITCH) {
17:            throw new IllegalArgumentException("Pitch is out of bounds.");
18:        }
19:        if (duration < MIN_DURATION || duration > MAX_DURATION) {
20:            throw new IllegalArgumentException("Duration is out of bounds.");
21:        }
22:        this.pitch = pitch;
23:        this.duration = duration;
24:    }
25:
26:    public boolean hasNoteAbove() {
27:        return pitch < MAX_PITCH;
28:    }
29:
30:    public boolean hasNoteBelow() {
31:        return pitch > MIN_PITCH;
32:    }
33:
34:    public Note noteAbove() {
35:        // PRE: There is a note above
36:        return new Note(pitch + 1, duration);
37:    }
38:
39:    public Note noteBelow() {
40:        // PRE: There is a note below
41:        return new Note(pitch - 1, duration);
42:    }
43:
44:    public int getDuration() {
45:        return duration;
46:    }
47:
48:    @Override
49:    public String toString() {
50:        int value = pitch % NUM_NOTES_PER_OCTAVE;
51:        String valueName = switch (value) {
52:            case 0 -> "C";
53:            case 1 -> "C#";
54:            case 2 -> "D";
55:            case 3 -> "D#";
56:            case 4 -> "E";
57:            case 5 -> "F";
58:            case 6 -> "F#";
59:            case 7 -> "G";
60:            case 8 -> "G#";
61:            case 9 -> "A";
62:            case 10 -> "A#";
63:            case 11 -> "B";
64:            default -> throw new RuntimeException(
65:                "Value out of bounds; check NUM_NOTES_PER_OCTAVE definition.");
66:        };
67:        int octave = pitch / NUM_NOTES_PER_OCTAVE;
68:        StringBuilder sb = new StringBuilder();
69:        sb.append(valueName);
70:        sb.append(octave);
71:        sb.append("(");
72:        sb.append(duration);
73:        sb.append(")");
74:        return sb.toString();
75:    }
76:
77:    @Override
78:    public boolean equals(Object that) {
79:        if (!that instanceof Note thatNote) {
80:            return false;
81:        }
```

```
82:        return thatNote.pitch == pitch && thatNote.duration == duration;
83:    }
84:
85:    @Override
86:    public int hashCode() {
87:        return pitch * duration;
88:    }
89: }
```

```
1: package tunes;
2:
3: import java.util.ArrayList;
4: import java.util.List;
5:
6: public class StandardTune implements Tune {
7:     private final List<Note> notes; ✓
8:
9:     public StandardTune() {
10:         notes = new ArrayList<>();
11:     }
12:
13:     @Override
14:     public List<Note> getNotes() {
15:         return new ArrayList<>(notes); ✓
16:     }
17:
18:     @Override
19:     public void addNote(Note note) {
20:         notes.add(note); ✓
21:     }
22: }
```

```
1: package tunes;
2:
3: import java.util.List;
4:
5: public class TransposedTune implements Tune {
6:     private final Tune targetTune;
7:     private final int pitchOffset; ✓
8:
9:     public TransposedTune(Tune targetTune, int pitchOffset) {
10:         this.targetTune = targetTune;
11:         this.pitchOffset = pitchOffset; ✓
12:     }
13:
14:     @Override
15:     public List<Note> getNotes() {
16:         return targetTune
17:             .getNotes()
18:             .stream()
19:             .map(note -> shift(note, pitchOffset))
20:             .toList(); ✓
21:     }
22:
23:     @Override
24:     public void addNote(Note note) {
25:         targetTune.addNote(shift(note, -pitchOffset));
26:     } ✓
27:
28:     private Note shift(Note note, int offset) {
29:         if (offset > 0 && note.hasNoteAbove()) {
30:             return shift(note.noteAbove(), offset - 1);
31:         } else if (offset < 0 && note.hasNoteBelow()) {
32:             return shift(note.noteBelow(), offset + 1);
33:         } else {
34:             return note;
35:         }
36:     }
37: }
```

good for
avoiding duplicate
code -

```
1: package tunes;
2:
3: import java.util.List;
4:
5: public interface Tune {
6:     List<Note> getNotes();
7:
8:     void addNote(Note note);
9:
10:     default int getTotalDuration() {
11:         return getNotes()
12:             .stream()
13:             .mapToInt(Note::getDuration)
14:             .reduce(0, Integer::sum);
15:     }
16: }
```

Section A	Output (1/2)	j1k21	Section A	Output (2/2)	j1k21
	1: Model Answer's Tests - Question1Tests works!		82: Model Answer's Tests - GoodPracticesTestsTune works!		
	2:		83:		
	3: JUnit version 4.12		84: JUnit version 4.12		
	4:		85: .		
	5: Time: 0.004		86: Time: 0.014		
	6:		87:		
	7: OK (10 tests)		88: OK (1 test)		
	8:		89:		
	9:		90:		
	10: Model Answer's Tests - Question2Tests works!				
	11:				
	12: JUnit version 4.12				
	13:				
	14: Time: 0.015				
	15:				
	16: OK (7 tests)				
	17:				
	18:				
	19: Model Answer's Tests - Question3Tests works!				
	20:				
	21: JUnit version 4.12				
	22:				
	23: Time: 0.016				
	24:				
	25: OK (11 tests)				
	26:				
	27:				
	28: Model Answer's Tests - Question4Tests works!				
	29:				
	30: JUnit version 4.12				
	31: ...				
	32: Time: 0.016				
	33:				
	34: OK (3 tests)				
	35:				
	36:				
	37: Model Answer's Tests - Question5Tests works!				
	38:				
	39: JUnit version 4.12				
	40:				
	41: Time: 0.033				
	42:				
	43: OK (9 tests)				
	44:				
	45:				
	46: Model Answer's Tests - Question6Tests works!				
	47:				
	48: JUnit version 4.12				
	49: ...				
	50: Time: 0.02				
	51:				
	52: OK (3 tests)				
	53:				
	54:				
	55: Model Answer's Tests - GoodPracticesTestsNote works!				
	56:				
	57: JUnit version 4.12				
	58:				
	59: Time: 0.015				
	60:				
	61: OK (4 tests)				
	62:				
	63:				
	64: Model Answer's Tests - GoodPracticesTestsStandardTune works!				
	65:				
	66: JUnit version 4.12				
	67: ..				
	68: Time: 0.016				
	69:				
	70: OK (2 tests)				
	71:				
	72:				
	73: Model Answer's Tests - GoodPracticesTestsTransposedTune works!				
	74:				
	75: JUnit version 4.12				
	76: ..				
	77: Time: 0.016				
	78:				
	79: OK (2 tests)				
	80:				
	81:				