

```
1 package hyperDap.base.testHelpers;
2
3 import static org.junit.Assert.assertEquals;
4 import java.math.BigDecimal;
5 import java.math.BigInteger;
6 import java.time.LocalDate;
7 import java.time.LocalDateTime;
8 import java.time.LocalTime;
9 import java.time.MonthDay;
10 import java.time.OffsetDateTime;
11 import java.time.OffsetTime;
12 import java.time.Year;
13 import java.time.YearMonth;
14 import java.time.ZoneId;
15 import java.time.ZoneOffset;
16 import java.time.ZonedDateTime;
17 import org.junit.jupiter.api.Test;
18 import hyperDap.base.helpers.Parser;
19
20 public class testParser {
21
22     @Test
23     void parseBoolean() {
24         Boolean test = true;
25         assertEquals(Parser.parse("true", Boolean.class), test);
26     }
27
28     @Test
29     void parseByte() {
30         Byte test = 123;
31         assertEquals(Parser.parse("123", Byte.class), test);
32     }
33
34     @Test
35     void parseShort() {
36         Short test = 123;
37         assertEquals(Parser.parse("123", Short.class), test);
38     }
39
40     @Test
41     void parseInt() {
42         Integer test = 123;
43         assertEquals(Parser.parse("123", Integer.class), test);
44     }
45
46     @Test
47     void parseLong() {
48         Long test = Long.valueOf(123);
49         assertEquals(Parser.parse("123", Long.class), test);
50     }
51
52     @Test
53     void parseBigInt() {
54         BigInteger test = BigInteger.valueOf(123);
55         assertEquals(Parser.parse("123", BigInteger.class), test);
56     }
57
58     @Test
59     void parseFloat() {
60         Float test = (float) 123.45;
61         assertEquals(Parser.parse("123.45", Float.class), test);
62     }
63
64     @Test
65     void parseDouble() {
66         Double test = 123.45;
67         assertEquals(Parser.parse("123.45", Double.class), test);
68     }
69
70     @Test
71     void parseBigDec() {
72         BigDecimal test = BigDecimal.valueOf(123.45);
73         assertEquals(Parser.parse("123.45", BigDecimal.class), test);
74     }
75 }
```

```

74     }
75
76     @Test
77     void parseString() {
78         String test = "Testing";
79         assertEquals(Parser.parse("Testing", String.class), test);
80     }
81
82     @Test
83     void parseYear() {
84         Year test = Year.parse("1996");
85         assertEquals(Parser.parse("1996", Year.class), test);
86     }
87
88     @Test
89     void parseYearMonth() {
90         YearMonth test = YearMonth.parse("1996-05");
91         assertEquals(Parser.parse("1996-05", YearMonth.class), test);
92     }
93
94     @Test
95     void parseMonthDay() {
96         MonthDay test = MonthDay.parse("--05-07");
97         assertEquals(Parser.parse("--05-07", MonthDay.class), test);
98     }
99
100    @Test
101    void parseDate() {
102        LocalDate test = LocalDate.parse("1996-05-07");
103        assertEquals(Parser.parse("1996-05-07", LocalDate.class), test);
104    }
105
106    @Test
107    void parseTime() {
108        LocalTime test = LocalTime.parse("22:58:03");
109        assertEquals(Parser.parse("22:58:03", LocalTime.class), test);
110    }
111
112    @Test
113    void parseZoneOffset() {
114        ZoneOffset test = ZoneOffset.of("-02:00");
115        assertEquals(Parser.parse("-02:00", ZoneOffset.class), test);
116    }
117
118    @Test
119    void parseTimeOffset() {
120        OffsetTime test = OffsetTime.parse("22:58:03-02:00");
121        assertEquals(Parser.parse("22:58:03-02:00", OffsetTime.class), test);
122    }
123
124    @Test
125    void parseDateTime() {
126        LocalDateTime test = LocalDateTime.parse("1996-05-07T22:58:03");
127        assertEquals(Parser.parse("1996-05-07T22:58:03", LocalDateTime.class), test);
128    }
129
130    @Test
131    void parseOffsetDateTime() {
132        OffsetDateTime test = OffsetDateTime.parse("1996-05-07T22:58:03-02:00");
133        assertEquals(Parser.parse("1996-05-07T22:58:03-02:00", OffsetDateTime.class),
134            test);
135    }
136
137    @Test
138    void parseZone() {
139        ZoneId test = ZoneId.of("Europe/Berlin");
140        assertEquals(Parser.parse("Europe/Berlin", ZoneId.class), test);
141    }
142
143    @Test
144    void parseZoneDateTime() {
145        ZonedDateTime test =
146            ZonedDateTime.parse("1996-05-07T22:58:03-02:00[Europe/Berlin]");

```

```
145     assertEquals(Parser.parse("1996-05-07T22:58:03-02:00[Europe/Berlin]",
146         ZonedDateTime.class),
147         test);
148 }
149
```