Plan Perfect

Seam: Database

Integration Test:

```
@Before
public void SQLDBsetUp() {
    String url = "jdbc:mysql://localhost:3306/CA_Public_Holidays";
    String user = "root";
    String password = "EECS2311"; // replace ... with your password

    events = new ArrayList<>();

    try (Connection con = DriverManager.getConnection(url, user, password)) {
        String[] queries = { "SELECT * FROM 2023_Holidays;", "SELECT * FROM 2024_Holidays;", "SELECT * FROM 2025_Holidays;"

        for (String query : queries) {
            try (Statement statement = con.createStatement();
                    ResultSet result = statement.executeQuery(query)) {
                while (result.next()) {
                    String holiday_Name = result.getString(columnLabel:"Holiday_Name");
                    int day = result.getInt(columnLabel:"day");
                    int month = result.getInt(columnLabel:"month");
                    int year = result.getInt(columnLabel:"year");

                    events.add(new CalendarEvent(LocalDate.of(year, month, day), LocalTime.of(hour:8, minute:0),
                            LocalTime.of(hour:9, minute:0), holiday_Name));
                }
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

@Test
public void testDayCalendar() {
    DayCalendar cal = new DayCalendar(events);
    assertNotNull(cal);
}
Test
public void testWeekCalendar() {
    WeekCalendar cal = new WeekCalendar(events);
    assertNotNull(cal);
}
```

Seam: Week / Day Calendar View

Integration Test:

```java
@Test
public void testGetStart() {
  LocalTime start = LocalTime.of(9, 0);
  LocalTime end = LocalTime.of(10, 0);
  TimeSlot ts = new TimeSlot(start, end);
  assertEquals(start, ts.getStart());
}

@Test
public void testGetEnd() {
  LocalTime start = LocalTime.of(9, 0);
  LocalTime end = LocalTime.of(10, 0);
  TimeSlot ts = new TimeSlot(start, end);
  assertEquals(end, ts.getEnd());
}

@Test
public void testGetStartOfWeek() {
    // Test Monday
    LocalDate date = LocalDate.of(2023, 4, 10); // Sunday
    LocalDate expected = LocalDate.of(2023, 4, 10); // Monday
    LocalDate actual = Week.getStartOfWeek(date);
    assertEquals(expected, actual);

    // Test Friday
    date = LocalDate.of(2023, 4, 14); // Friday
    expected = LocalDate.of(2023, 4, 10); // Monday
    actual = Week.getStartOfWeek(date);
    assertEquals(expected, actual);
}

@Test
public void testGetDay() {
    Week week = new Week(LocalDate.of(2023, 4, 10)); // Week starting from Monday, Apr 10
    LocalDate expected = LocalDate.of(2023, 4, 10); // Monday
    LocalDate actual = week.getDay(DayOfWeek.MONDAY);
    assertEquals(expected, actual);

    expected = LocalDate.of(2023, 4, 15); // Saturday
    actual = week.getDay(DayOfWeek.SATURDAY);
    assertEquals(expected, actual);
}

@Test
public void testNextWeek() {
    Week week = new Week(LocalDate.of(2023, 4, 10)); // Week starting from Monday, Apr 10
    Week expected = new Week(LocalDate.of(2023, 4, 21)); // Week starting from Monday, Apr 17
    Week actual = week.nextWeek();
    assertEquals(expected.toString(), actual.toString());
}
```

Diagram: