Franklin Marathon Test-Driven Development Plan

TDD Requirement 1.1:

Given a date stamp in the format of YYYYMMDD, the system shall be able to determine the location of that time within the race calendar year, according to the following table:

Dates	Race Calendar Period
1 Jun – 30 Sep	Registration Not Open
Oct 1 – Oct 31	Super Early
Nov 1 – Feb 28/29	Early
Mar 1 – Apr 1	Baseline
Apr 2 – TDay	Late
TDay – 31 May	Registration Closed

where TDay is the Thursday before the first Saturday of May.

Test Case 1
- Input: 20250930
- Expected Output: Registration Not Open
- Actual Output: Registration Not Open
- Pass/Fail: Pass

Test Case 2
- Input: 20251001
- Expected Output: Super Early
- Actual Output: Super Early
- Pass/Fail: Pass

Test Case 3
- Input: 20251031
- Expected Output: Super Early
- Actual Output: Super Early
- Pass/Fail: Pass

Test Case 4
- Input: 20251101
- Expected Output: Early
- Actual Output: Early
- Pass/Fail: Pass

Test Case 5
- Input: 20260228
- Expected Output: Early
- Actual Output: Early
- Pass/Fail: Pass

Test Case 6
- Input: 20260301
- Expected Output: Baseline
- Actual Output: Baseline
- Pass/Fail: Pass

Test Case 7
- Input: 20260401
- Expected Output: Baseline
- Actual Output: Baseline
- Pass/Fail: Pass

Test Case 8
- Input: 20260402
- Expected Output: Late
- Actual Output: Late
- Pass/Fail: Pass

Test Case 9
- Input: 20260430
- Expected Output: Late
- Actual Output: Late
- Pass/Fail: Pass

Test Case 10
- Input: 20250501
- Expected Output: Registration Not Open
- Actual Output: Registration Not Open
- Pass/Fail: Pass

```java
import java.time.DayOfWeek;
import java.time.LocalDate;
import java.time.Month;
import java.time.format.DateTimeFormatter;
import java.util.Scanner;

public class req1 {
    private static final DateTimeFormatter INPUT_FMT =
DateTimeFormatter.ofPattern("yyyyMMdd");

    public static void main(String[] args) {
        String input;
        System.out.print("Enter date (YYYYMMDD): ");
        Scanner sc = new Scanner(System.in);
        input = sc.next().trim();
        sc.close();

        LocalDate date;
        date = LocalDate.parse(input, INPUT_FMT);

        String period = determineRacePeriod(date);
        System.out.println(period);
    }

    public static String determineRacePeriod(LocalDate d) {
        int raceYear = d.getYear() + (d.getMonthValue() >= 6 ? 1 : 0);

        LocalDate superEarly = LocalDate.of(raceYear - 1, Month.OCTOBER, 1);
        LocalDate early = LocalDate.of(raceYear - 1, Month.NOVEMBER, 1);
        LocalDate baseline = LocalDate.of(raceYear, Month.MARCH, 1);
        LocalDate late = LocalDate.of(raceYear, Month.APRIL, 2);
        LocalDate registrationEnd = LocalDate.of(raceYear, Month.MAY, 31);

        if(d.isBefore(superEarly) || d.isAfter(registrationEnd)){
            return "Registration Not Open";
        }else if(!d.isBefore(superEarly) && d.isBefore(early)){
            return "Super Early";
        }else if(!d.isBefore(early) && d.isBefore(baseline)){
            return "Early";
        }else if(!d.isBefore(baseline) && d.isBefore(late)){
            return "Baseline";
        }else {
            return "Late";
        }
    }
```

```
    public static LocalDate computeTDay(int year) {
        LocalDate firstOfMay = LocalDate.of(year, Month.MAY, 1);
        DayOfWeek dow = firstOfMay.getDayOfWeek();
        int daysToAdd = (DayOfWeek.SATURDAY.getValue() - dow.getValue() + 7) % 7;
        LocalDate firstSaturday = firstOfMay.plusDays(daysToAdd);
        return firstSaturday.minusDays(2);
    }
}
```

TDD Requirement 1.2:

Given a date of birth, the system shall be able to calculate the age of a runner on race day. (Race day is TDay + 2 for the 5K and 10K, and TDay + 3 for the full and half marathons.)

Test Case 1

Input: 19980502
Expected Output: 27 sat, 27 sun
Actual Output: 27 sat, 27 sun
Pass/Fail: Pass

Test Case 2

Input: 19980503
Expected Output: 26 sat, 27 sun
Actual Output: 26 sat, 27 sun
Pass/Fail: Pass

```java
import java.time.LocalDate;
import java.time.Period;
import java.time.format.DateTimeFormatter;
import java.util.Scanner;

public class req2 {
    private static final DateTimeFormatter DOB_FMT =
DateTimeFormatter.ofPattern("yyyyMMdd");

    public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    LocalDate dob = LocalDate.parse(sc.next().trim(), DOB_FMT);
    sc.close();

    LocalDate today = LocalDate.now();
    int raceYear = today.getYear() + ((today.getMonthValue() >= 10 &&
today.getDayOfMonth() >= 1) ? 1 : 0);
        sc.close();

        LocalDate tday = req1.computeTDay(raceYear);
        LocalDate raceDayShort = tday.plusDays(2);
        LocalDate raceDayLong = tday.plusDays(3);

        int ageShort = calculateAge(dob, raceDayShort);
        int ageLong = calculateAge(dob, raceDayLong);

        System.out.println(ageShort);
        System.out.println(ageLong);
    }

    public static int calculateAge(LocalDate dob, LocalDate onDate) {
        return Period.between(dob, onDate).getYears();
    }
}
```