Franklin Marathon Test-Driven Development Plan

TDD Requirement 2.1:

Given the information in the column on the left, the system shall add a runner to the race roster that includes the information in the column on the right

Info from Registration	Info Stored in Race Roster
a. First Name	a. First Name
b. Last Name	b. Last Name
c. Date of Birth	c. Date of Birth
d. Gender	d. Gender
e. Email Address	e. Email Address
f. Registration Timestamp	f. Registration Timestamp

Test Case 1

Input	First: John Last: Smith DOB: 2000-05-20 Gender: M Email: john.smith@email.com
Expected Output	First: John Last: Smith Age: 25 Gender: M Email: john.smith@email.com
Actual Output	First: John Last: Smith Age: 25 Gender: M Email: john.smith@email.com
Pass/Fail	Pass

Test Case 2

Input	First: John Last: Smith DOB: 2000-05-20 Gender: M Email: john.smith@email.com
Expected Output	Duplicate entry. Not added to roster.
Actual Output	Duplicate entry. Not added to roster.
Pass/Fail	Pass

Test Case 3

Input	First: Sarah Last: Jones DOB: 2001-05-20 Gender: F Email: sarah.jones@email.com
Expected Output	First: Sarah Last: Jones Age: 24 Gender: F Email: sarah.jones@email.com
Actual Output	First: Sarah Last: Jones Age: 24 Gender: F Email: sarah.jones@email.com
Pass/Fail	Pass

```java
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.Scanner;

public class req21 {
    private static final DateTimeFormatter DOB_FMT =
DateTimeFormatter.ofPattern("yyyyMMdd");
    private static final String ROSTER_FILE = "RaceRoster";

    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter first name :");
        String firstName = sc.next().trim();
        System.out.print("Enter last name :");
        String lastName = sc.next().trim();
        System.out.print("Enter date of birth (YYYYMMDD) :");
        String dobStr = sc.next().trim();
        System.out.print("Enter gender :");
        String gender = sc.next().trim();
        System.out.print("Enter email :");
        String email = sc.next().trim();
        System.out.print("Enter registration timestamp (YYYYMMDDHHMMSS) :");
        String regTs = sc.next().trim();
        sc.close();

        LocalDate dob = LocalDate.parse(dobStr, DOB_FMT);
        LocalDate regDate = LocalDate.parse(regTs.substring(0, 8), DOB_FMT);
        int raceYear = regDate.getYear() + (regDate.getMonthValue() >= 6 ? 1 : 0);
        LocalDate tday = req11.computeTDay(raceYear);
        LocalDate raceDayShort = tday.plusDays(2);

        int ageOnRaceDay = req12.calculateAge(dob, raceDayShort);

        String entry = firstName + "\n" + lastName + "\n" + ageOnRaceDay + "\n" +
gender + "\n" + email + "\n" + regTs;
```

```
        if (!isDuplicate(entry)) {
            try (FileWriter fw = new FileWriter(ROSTER_FILE, true);
                 PrintWriter pw = new PrintWriter(fw)) {
                pw.println(firstName);
                pw.println(lastName);
                pw.println(ageOnRaceDay);
                pw.println(gender);
                pw.println(email);
                pw.println(regTs);
            }
            System.out.println(firstName);
            System.out.println(lastName);
            System.out.println(ageOnRaceDay);
            System.out.println(gender);
            System.out.println(email);
            System.out.println(regTs);
        } else {
            System.out.println("Duplicate entry. Not added to roster.");
        }
    }

    private static boolean isDuplicate(String entry) throws IOException {
        File file = new File(ROSTER_FILE);
        if (!file.exists()) {    return false;    }

        try (BufferedReader br = new BufferedReader(new FileReader(file))) {
            StringBuilder existingEntry = new StringBuilder();
            String line;
            int lineCount = 0;

            while ((line = br.readLine()) != null) {
                existingEntry.append(line);
                lineCount++;
                if (lineCount == 6) {
                    if (existingEntry.toString().equals(entry.replace("\n", ""))) {
                        return true;
                    }
                    existingEntry.setLength(0);
                    lineCount = 0;
                }
            }
        }
        return false;
    }
}
```

TDD Requirement 2.2:
The system shall allow a runner to sign up for any one of four races (5K, 10K, Half Marathon, and Full Marathon) and add the runner's information to the correct roster.

Test Case 1

Input	First: John Last: Smith DOB: 2000-05-20 Gender: M Email: john.smith@email.com Race: 5k
Expected Output	First: John Last: Smith Age: 25 Gender: M Email: john.smith@email.com Race: 5k
Actual Output	First: John Last: Smith Age: 25 Gender: M Email: john.smith@email.com Race: 5k
Pass/Fail	Pass

Test Case 2

Input	First: John Last: Smith DOB: 2000-05-20 Gender: M Email: john.smith@email.com Race: 5k
Expected Output	Duplicate entry. Not added to roster.
Actual Output	Duplicate entry. Not added to roster.
Pass/Fail	Pass

Test Case 3

Input	First: Sarah Last: Jones DOB: 2001-05-20 Gender: F Email: sarah.jones@email.com Race: 10k
Expected Output	First: Sarah Last: Jones Age: 24 Gender: F Email: sarah.jones@email.com Race: 10k
Actual Output	First: Sarah Last: Jones Age: 24 Gender: F Email: sarah.jones@email.com Race: 10k
Pass/Fail	Pass

Test Case 4

Input	James O'Connor 1985-06-01 M james.oconnor@email.com Half Marathon
Expected Output	James O'Connor Age: 40 M james.oconnor@email.com Half Marathon
Actual Output	James O'Connor Age: 40 M james.oconnor@email.com Half Marathon
Pass/Fail	Pass

Test Case 5

Input	Olivia Davis 1997-11-30 F olivia.davis@email.com Full Marathon
Expected Output	Olivia Davis Age: 28 F olivia.davis@email.com Full Marathon
Actual Output	Olivia Davis Age: 28 F olivia.davis@email.com Full Marathon
Pass/Fail	Pass

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.Scanner;

public class req22 {
    private static final DateTimeFormatter DOB_FMT =
DateTimeFormatter.ofPattern("yyyyMMdd");
    private static final String _5K_ROSTER_FILE = "5k_RaceRoster";
    private static final String _10K_ROSTER_FILE = "10k_RaceRoster";
    private static final String HALF_ROSTER_FILE = "Half_RaceRoster";
    private static final String FULL_ROSTER_FILE = "Full_RaceRoster";

    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter first name :");
        String firstName = sc.next().trim();
        System.out.print("Enter last name :");
        String lastName = sc.next().trim();
        System.out.print("Enter date of birth (YYYYMMDD) :");
        String dobStr = sc.next().trim();
        System.out.print("Enter gender :");
        String gender = sc.next().trim();
        System.out.print("Enter email :");
        String email = sc.next().trim();
        System.out.print("Enter registration timestamp (YYYYMMDDHHMMSS) :");
        String regTs = sc.next().trim();
        System.out.println("Enter race category :");
        String raceCategory = sc.next().trim();
        sc.close();

        LocalDate dob = LocalDate.parse(dobStr, DOB_FMT);
        LocalDate regDate = LocalDate.parse(regTs.substring(0, 8), DOB_FMT);
        int raceYear = regDate.getYear() + (regDate.getMonthValue() >= 6 ? 1 : 0);
        LocalDate tday = req11.computeTDay(raceYear);
        //  TODO this will be adjusted later to determine age on the day of the
different races
        LocalDate raceDayShort = tday.plusDays(2);
        int ageOnRaceDay = req12.calculateAge(dob, raceDayShort);
```

```
        String entry = firstName + "\n" + lastName + "\n" + ageOnRaceDay + "\n" +
gender + "\n" + email + "\n" + regTs;

        String ROSTER_FILE;
        switch (raceCategory.toLowerCase()) {
            case "5k":
                ROSTER_FILE = _5K_ROSTER_FILE;
                break;
            case "10k":
                ROSTER_FILE = _10K_ROSTER_FILE;
                break;
            case "half":
                ROSTER_FILE = HALF_ROSTER_FILE;
                break;
            case "full":
                ROSTER_FILE = FULL_ROSTER_FILE;
                break;
            default:
                System.out.println("Invalid race category.");
                return;
        }

        if (!isDuplicate(entry)) {
            try (FileWriter fw = new FileWriter(ROSTER_FILE, true);
                 PrintWriter pw = new PrintWriter(fw)) {
                pw.println(firstName);
                pw.println(lastName);
                pw.println(ageOnRaceDay);
                pw.println(gender);
                pw.println(email);
                pw.println(regTs);
            }
            System.out.println(firstName);
            System.out.println(lastName);
            System.out.println(ageOnRaceDay);
            System.out.println(gender);
            System.out.println(email);
            System.out.println(regTs);
            System.out.println(raceCategory);
        } else {
            System.out.println("Duplicate entry. Not added to roster.");
        }
    }
```

```
    private static boolean isDuplicate(String entry) throws IOException {
        File[] files = {new File(_5K_ROSTER_FILE), new File(_10K_ROSTER_FILE), new
File(HALF_ROSTER_FILE), new File(FULL_ROSTER_FILE)};

        for (File file : files) {
            if (!file.exists()) {   continue;   }
            try (BufferedReader br = new BufferedReader(new FileReader(file))) {
                StringBuilder existingEntry = new StringBuilder();
                String line;
                int lineCount = 0;
                while ((line = br.readLine()) != null) {
                    existingEntry.append(line);
                    lineCount++;
                    if (lineCount == 6) {
                        if (existingEntry.toString().equals(entry.replace("\n", ""))) {
                            return true;
                        }
                        existingEntry.setLength(0);
                        lineCount = 0;
                    }
                }
            }
        }
        return false;
    }
}
```