# Software Requirements Specification (SRS)

## Table of Contents

## 1. Introduction

### 1.1 Purpose

The purpose of this document is to provide a detailed description of the YearCalendar program. This document will cover the requirements, functionalities, design constraints, and other essential aspects needed for developing and understanding the system.

### 1.2 Scope

The YearCalendar program generates and displays a calendar for a user-specified year, detailing each month with day headers and correctly aligned dates. It is a console-based application developed in Java.

### 1.3 Definitions, Acronyms, and Abbreviations

- **SRS**: Software Requirements Specification
- **Java**: A high-level, class-based, object-oriented programming language
- **LocalDate**: A class in the java.time package representing a date without a time-zone
- **DateTimeFormatter**: A formatter for printing and parsing date-time objects

### 1.4 References

- Java SE Documentation
- ISO 8601 Date and Time Format Standard

### 1.5 Overview

This document includes a detailed description of the YearCalendar program, outlining the product perspective, functionality, user interactions, performance requirements, and design constraints.

---

## 2. Overall Description

### 2.1 Product Perspective

The YearCalendar program is a standalone console application. It does not interact with other systems or databases.

### 2.2 Product Functions

- Prompt user for a year input
- Generate and display the calendar for each month of the specified year
- Correctly align dates under respective day headers

### 2.3 User Classes and Characteristics

The primary users of this program are:

- General users who need to view a yearly calendar

### 2.4 Operating Environment

The program operates in a console environment on systems with Java Runtime Environment (JRE) installed.

### 2.5 Design and Implementation Constraints

- The program must be implemented in Java.
- It must use the java.time package for date and time manipulation.

### 2.6 Assumptions and Dependencies

- Users will have basic knowledge of operating a console application.
- The system running the program will have Java installed.

---

## 3. Specific Requirements

### 3.1 External Interface Requirements

#### 3.1.1 User Interfaces

- **Console Input**: The program will prompt the user to enter a year.
- **Console Output**: The program will display the calendar month-by-month.

#### 3.1.2 Hardware Interfaces

- Standard input and output devices (keyboard and monitor)

#### 3.1.3 Software Interfaces

- Java Runtime Environment (JRE) for executing the program

#### 3.1.4 Communications Interfaces

- No specific communication interfaces are required.

### 3.2 Functional Requirements

#### 3.2.1 Prompt for Year Input

- **Description**: The system shall prompt the user to input a year.
- **Rationale**: Required to generate the calendar for the specified year.
- **Source**: Developer

#### 3.2.2 Generate and Display Calendar

- **Description**: The system shall generate and display the calendar for each month of the specified year.
- **Rationale**: Allows users to view the calendar month-by-month.
- **Source**: Developer

### 3.2.3 Align Dates under Day Headers

- **Description**: The system shall align the dates correctly under the respective day headers (Sun, Mon, Tue, Wed, Thu, Fri, Sat).
- **Rationale**: Provides a clear and accurate visual representation of the calendar.
- **Source**: Developer

## 3.3 Performance Requirements

- The program should generate and display the calendar in less than 1 second after the year input is provided.

## 3.4 Logical Database Requirements

- Not applicable as the program does not interact with any databases.

## 3.5 Design Constraints

- Must use Java and its java.time package for date handling.

## 3.6 Software System Attributes

### 3.6.1 Reliability

- The system should reliably generate the correct calendar for any valid year input.

### 3.6.2 Availability

- The program should be available for use at any time on a system with Java installed.

### 3.6.3 Security

- The program does not handle sensitive information; basic input validation is sufficient.

### 3.6.4 Maintainability

- The code should be well-documented and modular to facilitate future maintenance.

### 3.6.5 Portability

- The program should run on any operating system with Java Runtime Environment installed.

---

## 4. Appendices

### 4.1 Appendix A: Glossary

- **LocalDate**: A date without a time-zone in the ISO-8601 calendar system, such as 2007-12-03.
- **DateTimeFormatter**: Formatter for printing and parsing date-time objects.
- **Scanner**: A simple text scanner which can parse primitive types and strings using regular expressions.

### 4.2 Appendix B: Analysis Models

- Use Case Diagrams
- Class Diagrams
- Sequence Diagrams

### 4.3 Appendix C: Issues List

- Known Issues
- Potential Enhancements

### 4.4 Appendix D: References

- Java SE Documentation: https://docs.oracle.com/javase/8/docs/api/

**Fields**

There are many fields in the Calendar class. Here are some of the most important ones:

- YEAR: The field indicating the year.
- MONTH: The field indicating the month.
- DAY_OF_MONTH: The field indicating the day of the month.
- DATE: Synonym for DAY_OF_MONTH.

- HOUR: The field indicating the hour of the day.
- MINUTE: The field indicating the minute within the hour.
- SECOND: The field indicating the second within the minute.
- MILLISECOND: The field indicating the millisecond within the second.

| No | Method | Description |
|---|---|---|
| 1. | public void add(int field, int amount) | Adds the specified (signed) amount of time to the given calendar field. |
| 2. | public boolean after (Object when) | The method Returns true if the time represented by this Calendar is after the time represented by when Object. |
| 3. | public boolean before(Object when) | The method Returns true if the time represented by this Calendar is before the time represented by when Object. |
| 4. | public final void clear(int field) | Set the given calendar field value and the time value of this Calendar undefined. |
| 5. | public Object clone() | Clone method provides the copy of the current object. |
| 6. | public int compareTo(Calendar anotherCalendar) | The compareTo() method of Calendar class compares the time values (millisecond offsets) between two calendar object. |
| 7. | protected void complete() | It fills any unset fields in the calendar fields. |
| 8. | protected abstract void computeFields() | It converts the current millisecond time value time to calendar field values in fields[]. |
| 9. | protected abstract void computeTime() | It converts the current calendar field values in fields[] to the millisecond time value time. |
| 10. | public boolean equals(Object object) | The equals() method compares two objects for equality and Returns true if they are equal. |
| 11. | public int get(int field) | In get() method fields of the calendar are passed as the parameter, and this method Returns the value of fields passed as the parameter. |
| 12. | public int getActualMaximum(int field) | Returns the Maximum possible value of the calendar field passed as the parameter to getActualMaximum() method. |
| 13. | public int getActualMinimum(int field) | Returns the Minimum possible value of the calendar field passed as parameter to getActualMinimum() methot. |
| 14. | public static Set<String> getAvailableCalendarTypes() | Returns a set which contains string set of all available calendar type supported by Java Runtime Environment. |

| | | |
|---|---|---|
| 15. | public static Locale[] getAvailableLocales() | Returns an array of all locales available in java runtime environment. |
| 16. | public String getCalendarType() | Returns in string all available calendar type supported by Java Runtime Environment. |
| 17. | public String getDisplayName(int field, int style, Locale locale) | Returns the String representation of the calendar field value passed as the parameter in a given style and local. |
| 18. | public Map<String,Integer> getDisplayNames(int field, int style, Locale locale) | Returns Map representation of the calendar field value passed as the parameter in a given style and local. |
| 19. | public int getFirstDayOfWeek() | Returns the first day of the week in integer form. |
| 20. | public abstract int getGreatestMinimum(int field) | This method returns the highest minimum value of Calendar field passed as the parameter. |
| 21. | public static Calendar getInstance() | This method is used with calendar object to get the instance of calendar according to current time zone set by java runtime environment |
| 22. | public abstract int getLeastMaximum(int field) | Returns smallest value from all maximum value for the field specified as the parameter to the method. |
| 23. | public abstract int getMaximum(int field) | This method is used with calendar object to get the maximum value of the specified calendar field as the parameter. |
| 24. | public int getMinimalDaysInFirstWeek() | Returns required minimum days in integer form. |
| 25. | public abstract int getMinimum(int field) | This method is used with calendar object to get the minimum value of specified calendar field as the parameter. |
| 26. | public final Date getTime() | This method gets the time value of calendar object and Returns date. |
| 27. | public long getTimeInMillis() | Returns the current time in millisecond. This method has long as return type. |
| 28. | public TimeZone getTimeZone() | This method gets the TimeZone of calendar object and Returns a TimeZone object. |
| 29. | public int getWeeksInWeekYear() | Return total weeks in week year. Weeks in week year is returned in integer form. |

| | | |
|---|---|---|
| 30. | public int getWeekYear() | This method gets the week year represented by current Calendar. |
| 31. | public int hashCode() | All other classes in Java overload hasCode() method. This method Returns the hash code for calendar object. |
| 32. | protected final int internalGet(int field) | This method returns the value of the calendar field passed as the parameter. |
| 33. | Public boolean isLenient() | Return Boolean value. True if the interpretation mode of this calendar is lenient; false otherwise. |
| 34. | public final boolean isSet(int field) | This method checks if specified field as the parameter has been set or not. If not set then it returns false otherwise true. |
| 35. | public boolean isWeekDateSupported() | Checks if this calendar supports week date. The default value is false. |
| 36. | public abstract void roll(int field, boolean up) | This method increase or decrease the specified calendar field by one unit without affecting the other field |
| 37. | public void set(int field, int value) | Sets the specified calendar field by the specified value. |
| 38. | public void setFirstDayOfWeek(int value) | Sets the first day of the week. The value which is to be set as the first day of the week is passed as parameter. |
| 39. | public void setMinimalDaysInFirstWeek(int value) | Sets the minimal days required in the first week. The value which is to be set as minimal days in first week is passed as parameter. |
| 40. | public final void setTime(Date date) | Sets the Time of current calendar object. A Date object id passed as the parameter. |
| 41. | public void setTimeInMillis(long millis) | Sets the current time in millisecond. |
| 42. | public void setTimeZone(TimeZone value) | Sets the TimeZone with passed TimeZone value (object) as the parameter. |
| 43. | public void setWeekDate(int weekYear, int weekOfYear, int dayOfWeek) | Sets the current date with specified integer value as the parameter. These values are weekYear, weekOfYear and dayOfWeek. |
| 44. | public final Instant toInstant() | The toInstant() method convert the current object to an instant. |

| | | |
|---|---|---|
| **45.** | **public String toString()** | **Returns string representation of the current object.** |

## CODE

```
import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

import java.time.DayOfWeek;

import java.util.Scanner;


public class YearCalendar {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a year: ");

        int year = sc.nextInt();

        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("MMMM
yyyy");


        for (int month = 1; month <= 12; month++) {

            LocalDate date = LocalDate.of(year, month, 1);

            System.out.println("\n" + date.format(formatter));


            System.out.println("Sun Mon Tue Wed Thu Fri Sat");


            int fw = date.getDayOfWeek().getValue();
            for (int i = 1; i < fw; i++) {
                System.out.print("    ");
            }
            int dm = date.lengthOfMonth();
            for (int day = 1; day <= dm; day++) {
                System.out.printf("%3d ", day);
                if ((day + fw - 1) % 7 == 0) {
                    System.out.println();
                }
            }
            System.out.println();
        }

    }
}
```

## Algorithm: YearCalendar

1. **Start**
2. **Initialize Scanner for user input**
    - Scanner sc = new Scanner(System.in);
3. **Prompt user to enter a year**
    - Print: "Enter a year: "
    - Read input year: int year = sc.nextInt();
4. **Initialize DateTimeFormatter for month-year format**
    - DateTimeFormatter formatter = DateTimeFormatter.ofPattern("MMMM yyyy");

5. **Loop through each month (1 to 12)**
   - **For month from 1 to 12**
     - **Create LocalDate object for the first day of the month**
       - LocalDate date = LocalDate.of(year, month, 1);
     - **Print the formatted month and year**
       - Print: date.format(formatter)
     - **Print the days of the week header**
       - Print: "Sun Mon Tue Wed Thu Fri Sat"
     - **Determine the day of the week for the first day of the month**
       - int fw = date.getDayOfWeek().getValue();
     - **Adjust Sunday to be represented as 0 instead of 7**
       - If fw == 7, then set fw = 0
     - **Print leading spaces for days before the first day**
       - **For i from 0 to fw-1**
         - Print: " "
     - **Determine the number of days in the month**
       - int dm = date.lengthOfMonth();
     - **Print each day of the month**
       - **For day from 1 to dm**
         - Print the day: System.out.printf("%3d ", day)
         - **Check if the current position is the end of the week**
           - If (day + fw) % 7 == 0, then print a newline
     - **Print a newline after printing all days of the month**
6. **End Loop**
7. **End**

## Pseudocode

```
START

  Initialize Scanner for user input

  Prompt user to enter a year

  Read input year



  Initialize DateTimeFormatter for month-year format
```

```
FOR month from 1 to 12 DO
  Create LocalDate object for the first day of the month

  Print formatted month and year
  Print days of the week header

  Determine day of the week for the first day of the month

  IF day of the week is Sunday THEN
    Adjust Sunday to be represented as 0

  FOR each day before the first day of the month DO
    Print leading spaces

  Determine number of days in the month

  FOR day from 1 to the number of days in the month DO
    Print the day

    IF current position is the end of the week THEN
      Print a newline

  Print a newline after printing all days of the month
  END FOR

END
```

## Explanation

1. **Initialization**:
   - A Scanner object is created to read user input.
   - The user is prompted to enter a year.
   - A DateTimeFormatter object is set up to format month and year.
2. **Loop Through Each Month**:
   - For each month, a LocalDate object is created for the first day.
   - The month and year are formatted and printed.
   - The day headers (Sun, Mon, etc.) are printed.
   - The day of the week for the first day of the month is determined. Sunday is adjusted to 0 for alignment purposes.
   - Leading spaces are printed to align the first day correctly under its header.
   - The number of days in the month is determined.
   - Each day of the month is printed, and a newline is inserted after every Saturday (end of the week).
   - A final newline is printed after all days of the month.

This algorithm provides a step-by-step approach to how the program operates and generates the calendar for the specified year.