CENTURY CALENDER



An

Object-Oriented Programming through Java Course Project bReport in partial fulfillment of the degree

## Bachelor of Technology

in

## Electronic and Communication Engineering

**By**

S.SRIRAM REDDY 2105A41114

**Submitted to**





# DEPARTMENT OF ELECTRONIC AND COMMUNICATION

**ENGINEERING**

**CERTIFICATE**

This is to certify that the **Object Oriented Programming through Java - Course Project** Report entitled **“Century Calender”** is a record of bonafide work carried out by the student S.Sriram Reddy bearing Roll No(s)\_2105A41114\_ during the academic year 2023 in partial fulfillment of the award of the degree of ***Bachelor of Technology*** in **Electronic and Communication**

**Engineering** by the SR University , Ananthsagar ,Warangal.

**Lab In-charge Head of the Department**

# Table of Contents

|  |  |  |
| --- | --- | --- |
| **Chapter**  **No.** | **Title** | **Page No.** |
| 1 | OBJECTIVE OF THE PROJECT | 1 |
| **2** | DEFINITIONSOF THE ELEMENTS USED IN THE PROJECT | 2 |
| 3 | IMPLEMENTATION-CODE | 4 |
| 4 | CONCLUSION | 9 |

**ABSTRACT**

A century calendar is a way of organizing the years into groups of 100, starting from a certain point in time. Different calendar systems may have different starting points and ways of counting centuries. A century calendar is useful for historical and cultural purposes, as it allows people to identify and compare different periods of time. For example, the 20th century is often associated with the rise of modernity, while the 21st century is marked by globalization and digitalization. To use a century calendar, one needs to know the starting point and the numbering system of the calendar. For example, the Gregorian calendar, which is the most widely used calendar in the world, starts from the year 1 AD and counts centuries as 1-100, 101-200, and so on. Therefore, the year 2023 belongs to the 21st century in the Gregorian calendar. Other calendars, such as the Hindu calendar or the Aztec calendar, may have different starting points and groupings of years.

2

# OBJECTIVE

* It’s a Century calender code
* You can Print any Year Calender with this code
* You get any day with date with this code
* You save save the Incidents with date with code

# DEFINITIONS OF THE ELEMENTS USED IN THE PROJECT

**Class:** A class is a user-defined blueprint or prototype from which objects are created. It represents a collection of related data (attributes) and functions (methods) that operate on the data. For example, in a car rental system, a "Car" class can be created to represent the general attributes and behaviors of a car.

**Object:** An object is an instance of a class. It is a runtime entity that represents a specific occurrence of the class, with its own set of data and methods. For instance, if the "Car" class is defined, objects of the class could be "Toyota Corolla," "Ford Mustang," or any other specific car.

**Swings**: Java swing is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java Unlike AWT, Java Swing provides platform- independent and lightweight components.

**Java AWT (Abstract Window Toolkit)**: The Abstract Window Toolkit (AWT) was introduced as part of Java’s standard library in its early days and was the first GUI library for Java. [AWT provides a set of basic components and widgets, such as buttons, labels, text fields, and more, that developers can use to create simple GUI interfaces2](https://iq.opengenus.org/gui-in-java/).

**JavaFX**: JavaFX is a software platform for creating and delivering desktop applications, as well as rich internet applications (RIAs) that can run across a wide variety of devices. JavaFX is intended to replace Swing as the standard GUI library for Java.

**Inheritance:** Inheritance is a key feature of OOP that allows a class (subclass or derived class) to inherit properties and behaviors from another class. It promotes code re usability and the creation of a hierarchical class structure. For example, a "SUV" class can inherit attributes and methods from the "Car" class.

**Encapsulation:** Encapsulation is the concept of bundling data (attributes) and methods (functions) that operate on the data, ensuring that they are accessed and modified only through well-defined interfaces. It helps in data hiding and protecting the integrity of the data.

**Polymorphic:** Polymorphic allows objects of different classes to be treated as objects of a common super class through the use of inheritance. It enables a single interface to be used for entities of different types. This facilitates flexibility and extensibility in the design of the software.

**Composition**:In object-oriented programming (OOP), composition is a design principle that enables creating complex objects by combining simpler objects or component

**Method overloading**: is a feature in object-oriented programming that allows a class to have multiple methods with the same name but different parameters. This enables a class to perform a similar operation in different ways based on the number or type of parameters. Method overloading is a form of compile-time polymorphism where the decision on which method to call is made during the compilation phase based on the number and type of arguments provided.

**Inteafaces:**

[In Object-Oriented Programming (OOP), an interface is a blueprint or a contract that defines a set of methods that a class must implement](https://www.javatpoint.com/interface-in-java). [It provides a standard way to define the behavior of a group of related classes](https://users.cs.utah.edu/~germain/PPS/Topics/interfaces.html). [An interface is a description of all functions that an object must have in order to be an "X"](https://users.cs.utah.edu/~germain/PPS/Topics/interfaces.html). [In Java, an interface is a mechanism to achieve abstraction3](https://www.geeksforgeeks.org/interfaces-in-java/). [An interface in Java is a blueprint of a behavior3](https://www.geeksforgeeks.org/interfaces-in-java/). [A Java interface contains static constants and abstract methods](https://www.geeksforgeeks.org/interfaces-in-java/).

# IMPLEMENTATION-CODE:

package Century\_Calender\_1114;

public class main {

public static void main(String[] args){

{

new GUI\_1114();

}

}

}

package Century\_Calender\_1114;

import javax.swing.\*;

import java.awt.event.\*;

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

import java.time.format.DateTimeParseException;

import java.util.HashMap;

import java.util.Scanner;

public class GUI\_1114 {

JFrame frame;

JComboBox<Integer> options;

public GUI\_1114() {

System.out.println("ENTER WHAT DO U NEED");

System.out.println("1.PRINT THE YEAR 2.ENTER DATE TO GET DAY 3.SAVE A INCIDENT WITH DATE:");

frame = new JFrame("Switch Case GUI");

options = new JComboBox<>(new Integer[]{1, 2, 3});

options.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

Integer selectedOption = (Integer) options.getSelectedItem();

switch (selectedOption) {

case 1 -> case1();

case 2 -> case2();

case 3 -> case3();

}

}

});

frame.add(options);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.pack();

frame.setVisible(true);

}

static void case1(){

Scanner input= new Scanner(System.in);

int Year;

System.out.println("ENTER THE YEAR:");

Year=input.nextInt();

YEAR\_1114 test=new YEAR\_1114(Year);

test.p(Year);

System.out.println();

System.out.println();

}

static void case2(){

System.out.println("ENTER THE DATE IN 00/00/0000:");

Scanner ram=new Scanner(System.in);

String date = ram.nextLine();

FINALDATE\_1114 Date = new FINALDATE\_1114();

LocalDate localDate = Date.getDate(date);

String dayOfWeek =Date.getDayOfWeek(localDate);

System.out.println("THE DAY IS :" + dayOfWeek);

System.out.println();

System.out.println();

}

static void case3(){

HashMap<LocalDate, String> incidentMap = new HashMap<>();

Scanner scanner = new Scanner(System.in);

while (true) {

try {

System.out.println("ENTER THE DATE IN FORMAT YYYY-MM-DD:");

String dateStr = scanner.nextLine();

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");

LocalDate date1 = LocalDate.parse(dateStr, formatter);

System.out.println("ENTER THE INCIDENT");

String incident = scanner.nextLine();

incidentMap.put(date1, incident);

System.out.println("DO YOU WANT TO ENTER ANOTHER INCIDENT(YES/NO)");

String response = scanner.nextLine();

if (!response.equalsIgnoreCase("YES")) {

break;

}

} catch (DateTimeParseException e) {

System.out.println("INVALID DATA TRY AGAIN");

}

}

System.out.println("INCIDENTS:");

for (LocalDate date1 : incidentMap.keySet()) {

System.out.println("DATE: " + date1 + ", INCIDENT: " + incidentMap.get(date1));

}

System.out.println();

System.out.println();

}

}

public class YEAR\_1114 {

private static int year = 0;

public YEAR\_1114(int year) {

YEAR\_1114.year =year;

}

void p(int y)

{

int year, daycode, leapyear;

year = inputyear(y);

daycode = determinedaycode(year);

determineleapyear(year);

calendar(year, daycode);

System.out.println(" ");

}

int[] daysinmonth = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

String[] months =

{

" ",

"\n\n\nJANUARY",

"\n\n\nFEBRUARY",

"\n\n\nMARCH",

"\n\n\nAPRIL",

"\n\n\nMAY",

"\n\n\nJUNE",

"\n\n\nJULY",

"\n\n\nAUGUST",

"\n\n\nSEPTEMBER",

"\n\n\nOCTOBER",

"\n\n\nNOVEMBER",

"\n\n\nDECEMBER"

};

int inputyear (int y)

{

return y;

}

int determinedaycode ( int year)

{

int daycode;

int d1, d2, d3;

d1 = (int) ((year - 1) / 4.0);

d2 = (int) ((year - 1) / 100.);

d3 = (int) ((year - 1) / 400.);

daycode = (year + d1 - d2 + d3) % 7;

return daycode;

}

void determineleapyear (int year)

{

if (year % 4 == 0 && year % 100 != 0 || year % 400 == 0) {

daysinmonth[2] = 29;

} else {

daysinmonth[2] = 28;

}

}

void calendar ( int year, int daycode) {

int month, day;

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

System.out.println(months[month]);

System.out.print("\n\nSUN MON TUE WED THUS FRI SAT\n");

for (day = 1; day <= 1 + daycode \* 5; day++) {

System.out.print(" ");

}

for (day = 1; day <= daysinmonth[month]; day++) {

System.out.print(day);

if ((day + daycode) % 7 > 0)

System.out.print(" ");

else

System.out.println(" ");

}

daycode = (daycode + daysinmonth[month]) % 7;

}

}

}

abstract class ABSTRACTDATE\_1114 implements DAYTODATE\_1114 {

abstract String getDayOfWeek(LocalDate date);

}

public interface DAYTODATE\_1114 {

LocalDate getDate(String date);

}

package Century\_Calender\_1114;

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

class FINALDATE\_1114 extends ABSTRACTDATE\_1114 {

@Override

public LocalDate getDate(String date) {

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");

return LocalDate.parse(date, formatter);

}

@Override

public String getDayOfWeek(LocalDate date) {

return date.getDayOfWeek().toString();

}

}

# CONCLUSION:

By this project I have Learned all oop Concepts using the Java programming. I made this project by all java oop concepts.This project Name is Century calender I learned classes and how to create objects with them.Learned How to ude Inheritence How to Create parent Classes and child Class,I Learned interfaces Gui And Theards , Expection

Handling and many key words also.Finally by this Project I learned Many Things about Java and Gained Much Experience in coding java