# Assignment Description

Create a class called Date that has integer data members to store month, day, and year. The class will only have a single 3-parameter constructor which allows the date to be created by specifying the numerical month, day, and year. If the user creates a Date object when any of the values passed are invalid, an invalid argument exception should be thrown with appropriate text:

* The month must be a valid month (1-12),
* The day must be in the appropriate range for that month (do not worry about leap years), and
* The year must be between 1900 and 2020.

Include mutator methods for setting the month, day, and year. These should also throw invalid parameter exceptions if invalid parameters are provided. Include a toString() method that returns the date in the format “Month DD, YYYY” (ex: August 22, 1984).

Create a program that utilizes this class to create and display dates based on user input. Utilize try-catch blocks to validate the user input. Ensure that the exception thrown includes appropriate text to describe the problem (e.g. “Invalid Value – there are only 28 days in February” or “There are only 12 months in the year”). Demonstrate usage of the class and its embedded exception handling in a program that prompts users for initial date values, creates a date object, and then prompts the user to change the month, change the day, and change the year.

# 1 Readme Documentation

This program will prompt the user to input the year, month, and day of a date between January 1st, 1900, and December 31st, 2020. User input will be validated using try/catch blocks and invalid input will be discarded. The user can then edit their date or exit the program.

# 2 Flowchart Screen Shots

# 3 UML and Use Case Diagrams

# 4 Source Code of All files (.h, .cpp)

#include <iostream>

#include <iomanip>

#include <string>

#include <cctype>

#include <exception>

#include <map>

using namespace std;

/\*

Name: Date Input Validation

Author: Wesley Hixon

Date Last Updated: 11/2/2024

Purpose: Validate input of dates using try/catch blocks and custom exceptions

\*/

// Map which tells how many days are in each month

map<int, int> daysInMonth{

    {1, 31},

    {2, 28},

    {3, 31},

    {4, 30},

    {5, 31},

    {6, 30},

    {7, 31},

    {8, 31},

    {9, 30},

    {10, 31},

    {11, 30},

    {12, 31}

};

// Custom exception class which handles all out of range exceptions for month, day, year etc.

class outOfRangeException{

    private:

        string message;

    public:

        outOfRangeException(int userInteger, int beginRange, int endRange){

            if(userInteger < beginRange || userInteger > endRange){

                message = "Input must be between " + to\_string(beginRange) + " and " + to\_string(endRange);

            }

        }

        string what(){

            return message;

        }

};

class Date{

    private:

        int month;

        int day;

        int year;

    public:

        // Default constructor initializes as 1/1/1900

        Date(){

            day = 1;

            month = 1;

            year = 1900;

        }

        // Constructor which gets input for year, month, day

        Date(int inputYear, int inputMonth, int inputDay){

            setYear(inputYear);

            setMonth(inputMonth);

            setDay(inputDay);

        }

        // Converts an int month to a string (e.g. 12 -> "December")

        string monthToString(int month){

            string monthString;

            // Switch statement for int to month

            switch(month){

                case 1:

                    monthString = "January";

                    break;

                case 2:

                    monthString = "Feburary";

                    break;

                case 3:

                    monthString = "March";

                    break;

                case 4:

                    monthString = "April";

                    break;

                case 5:

                    monthString = "May";

                    break;

                case 6:

                    monthString = "June";

                    break;

                case 7:

                    monthString = "July";

                    break;

                case 8:

                    monthString = "August";

                    break;

                case 9:

                    monthString = "September";

                    break;

                case 10:

                    monthString = "October";

                    break;

                case 11:

                    monthString = "November";

                    break;

                case 12:

                    monthString = "December";

                    break;

            }

            return monthString;

        }

        // Converts current date into a string (e.g. 3/12/2020 -> "12th of March, 2020")

        string dateToString(){

            string daySuffix;

            switch(day){

                case 1: case 21: case 31:

                    daySuffix = "st";

                    break;

                case 2: case 22:

                    daySuffix = "nd";

                    break;

                case 3: case 23:

                    daySuffix = "rd";

                    break;

                default:

                    daySuffix = "th";

                    break;

            }

            string monthString = monthToString(month);

            string dateString = to\_string(day) + daySuffix + " of " + monthString + ", " + to\_string(year);

            return dateString;

        }

        // Getter functions

        int getYear(){

            return year;

        }

        int getMonth(){

            return month;

        }

        int getDay(){

            return day;

        }

        // Setter functions

        void setDay(int inputDay){

            // Get num of days in month

            int numDaysInMonth = daysInMonth[month];

            // Check that day is in range of the num of days in month

            try{

                if(inputDay < 1 || inputDay > numDaysInMonth) throw(outOfRangeException(inputDay, 1, numDaysInMonth));

                day = inputDay;

            }

            catch(outOfRangeException e){

                cerr << e.what() << endl;

            }

        }

        void setMonth(int inputMonth){

            // Check that month is > 1 and < 12

            try{

                if(inputMonth < 1 || inputMonth > 12) throw(outOfRangeException(inputMonth, 1, 12));

                month = inputMonth;

            }

            catch(outOfRangeException e){

                cerr << endl << e.what() << endl;

            }

        }

        void setYear(int inputYear){

            // Check that year is between 1900 and 2020

            try{

                if(inputYear < 1900 || inputYear > 2020) throw(outOfRangeException(inputYear, 1900, 2020));

                year = inputYear;

            }

            catch(outOfRangeException e){

                cerr << endl << e.what() << endl;

            }

        }

};

void menu(Date date);

int inputYear();

int inputMonth();

int inputDay(int month);

int main(){

    // Output introduction

    cout << "This program will allow you to store a date between January 1st, 1900 and December 31st, 2020." << endl;

    cout << "An error will be thrown if an invalid date is inputted, and you will be asked to re-enter your date." << endl;

    // Use input functions to get valid year, month, and day

    int year = inputYear();

    int month = inputMonth();

    int day = inputDay(month);

    // Create userDate object with year, month, and day

    Date userDate(year, month, day);

    // Menu lets user change year, month, or day

    menu(userDate);

    return 0;

}

// Menu to edit date or exit program

void menu(Date date){

    bool running = true;

    int menuChoice;

    while(running){

        // Outputting menu

        cout << "Your date is: " << date.dateToString() << endl;

        cout << "Select from the following menu options:" << endl

        << "1. Change Year" << endl

        << "2. Change Month" << endl

        << "3. Change Day" << endl

        << "4. Exit Program" << endl;

        bool valid = false;

        while(!valid){

            try{

                // Get input

                cin >> menuChoice;

                // In case of cin error or out of range error

                if(cin.fail()){

                    cin.clear();

                    cin.ignore(numeric\_limits<streamsize>::max(), '\n');

                    throw(string("Please enter a valid integer."));

                }

                if(menuChoice > 4 || menuChoice < 1) throw(outOfRangeException(menuChoice, 1, 4));

                valid = true;

            }

            // Output error messages and try again

            catch(outOfRangeException e){

                cerr << endl << e.what() << endl;

            }

            catch(string s){

                cerr << endl << s << endl;

            }

        }

        // Menu options

        switch(menuChoice){

            case 1:

                date.setYear(inputYear());

                break;

            case 2:

                date.setMonth(inputMonth());

                break;

            case 3:

                date.setDay(inputDay(date.getMonth()));

                break;

            case 4:

                running = false;

                break;

        }

    }

}

// Function to get input for a valid year

int inputYear(){

    bool valid = false;

    int year;

    cout << "Input a year for your date: ";

    // Validate input

    while(!valid){

        try{

            cin >> year;

            // In case of input failure

            if(cin.fail()){

                cin.clear();

                cin.ignore(numeric\_limits<streamsize>::max(), '\n');

                throw(string("Please enter a valid integer."));

            }

            // In case year out of range

            if(year < 1900 || year > 2020) throw(outOfRangeException(year, 1900, 2020));

            valid = true;

        }

        catch(const string& s){

            cerr << endl << s << endl;

        }

        catch(outOfRangeException& e){

            cerr << endl << e.what() << endl;

        }

    }

    return year;

}

// Function to get input for a valid month

int inputMonth(){

    bool valid = false;

    int month;

    cout << "Input a month for your date: ";

    // Validate input

    while(!valid){

        try{

            cin >> month;

            // In case of input failure

            if(cin.fail()){

                cin.clear();

                cin.ignore(numeric\_limits<streamsize>::max(), '\n');

                throw(string("Please enter a valid integer."));

            }

            // In case month out of range

            if(month < 1 || month > 12) throw(outOfRangeException(month, 1, 12));

            valid = true;

        }

        catch(string s){

            cerr << endl << s << endl;

        }

        catch(outOfRangeException e){

            cerr << endl << e.what() << endl;

        }

    }

    return month;

}

// Function to get input for a valid day in month

int inputDay(int month){

    bool valid = false;

    // Getting number of days in current month from map

    int numDaysInMonth = daysInMonth[month];

    int day;

    cout << "Input a day for your date: ";

    // Validating input

    while(!valid){

        try{

            cin >> day;

            // In case of input failure

            if(cin.fail()){

                cin.clear();

                cin.ignore(numeric\_limits<streamsize>::max(), '\n');

                throw(string("Please enter a valid integer."));

            }

            // In case day out of range of days in month

            if(day < 1 || day > numDaysInMonth) throw(outOfRangeException(day, 1, numDaysInMonth));

            valid = true;

        }

        catch(string s){

            cerr << endl << s << endl;

        }

        catch(outOfRangeException e){

            cerr << endl << e.what() << endl;

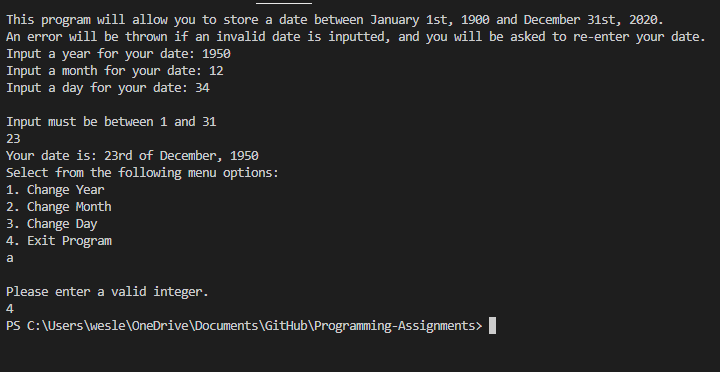
        }

    }

    return day;

}

# 5 Three Use Case Screen Shots



A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated