EOOP- Preliminary Project

Date: 18.04/2025 Semester: Spring 2025

Author and Group: Damian Piechocki 102

Subject (Keyword): Presidential Election

I. Description of the project

1. Overview of the project

Election stores the votes submitted by the voters. The votes are counted and converted to percentages in respect to the general population as well as individual voivodships. Candidates gain support by means of submitted votes. The winner of the election is determined according to the statistics, which are shown at the end. Such statistics include attendance and support by age groups for all registered candidates.

2. Class and data structures overview

There are the following classes: Election, Voter, Candidate, Voivodship.

The Election will contain a vector of voivodships. The main goal is to determine the winner of the election and express the Candidate's support in percentage with respect to the general population. The Election will also display the backing with respect to age, with the following categories: young adults <age 18-40), middle aged <40,65) and elders <65,). The election attendance will also be computed based on the voivodships.

The Voter will store basic information about voters like age, name, boolean value indicating vote submission, voivodship they live in and a helper boolean value indicating registration status(validity). It will most importantly include the functionality of submitting a vote. Only registered voters that meet the restrictions (see section I.3) are allowed to cast their vote on an existing, registered candidate. Vote submission status and validity can be modified in accordance with the principles of fair election.

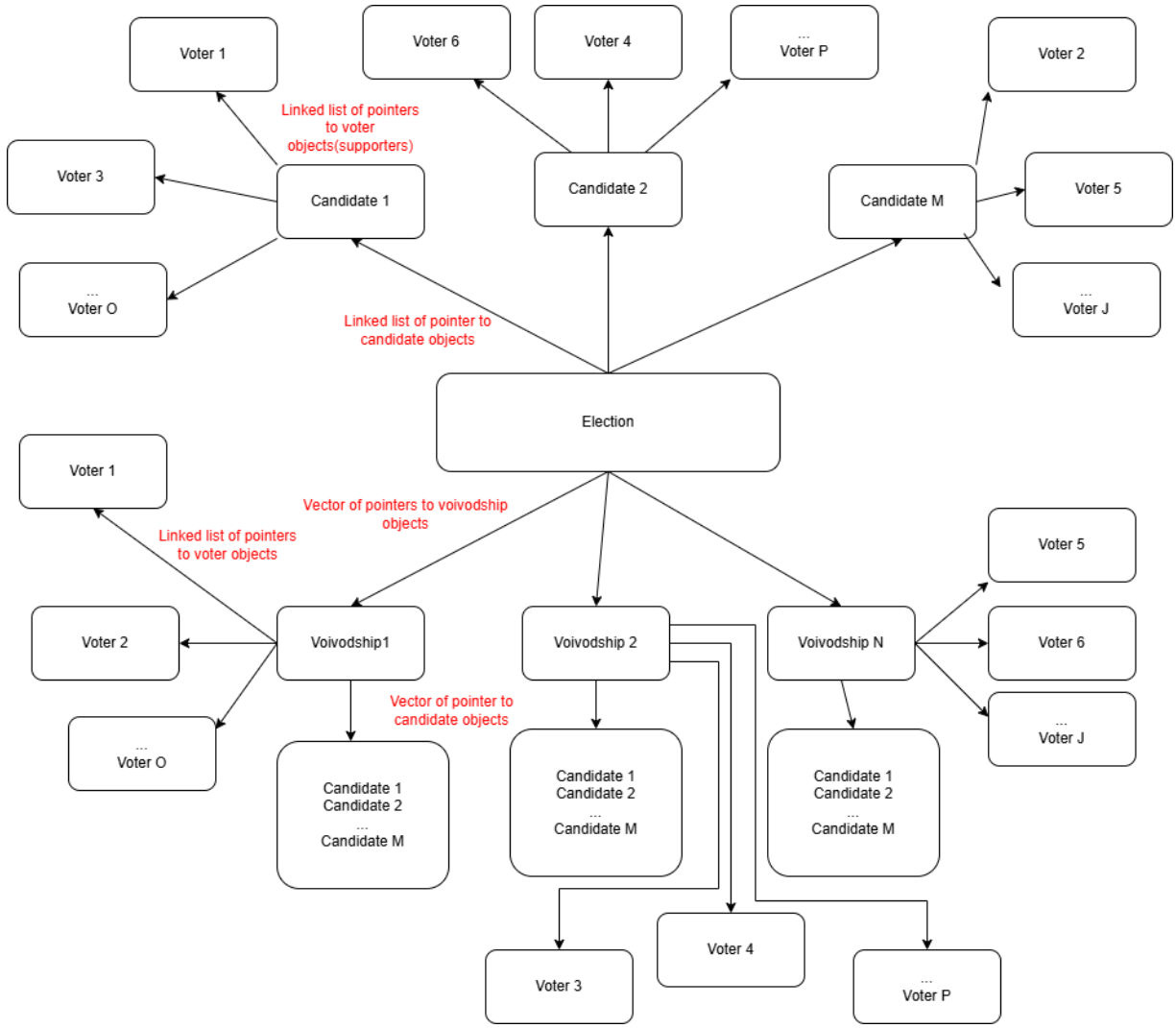
The Candidate will store information about the candidate, namely the support. It will inherit from the Voter, as candidates do have the voting rights. The backing increases, as the number of votes submitted in favor is greater. The Candidate will also store all the voters who submitted the vote on him/her. By default, the candidate can only vote for himself/herself.

The voivodship will include a singly linked list storing all the registered voters living within the voivodship and a vector of candidate objects. It will additionally store the number of citizens living within (not all people decide to vote) and a voivodship name. Its goal is to present the support of respective candidates in percentage solely by voters that live within the voivodship.

3. Restrictions, limits, assumptions

R1. Voter must be at least 18 years of age to vote.
R2. The number of voters cannot outnumber the number of citizens of voivodship.
R3. Each vote has the same weight and can be submitted only once.
R4. Candidates must be at least 35 years of age to part-take.
R5. Candidates can also be voters, they can vote.
R6. Candidates cannot have the number of votes greater than the total amount of votes.
R7. Only registered voters and candidates that meet the above assumptions can fulfill their intended functionality

II. Case study (a memory map)



Where M, N,P, O, J are definite numbers and well defined within the code.

III. Declaration of the classes

```cpp
#ifndef CLASSES_H
#define CLASSES_H
#include <vector>
#include <iostream>
#include <map>
using namespace std;

class Candidate;
class Voter;
class Election;
class Voivodship;

class Election{
    private:
        vector<Voivodship*> voivodships; //Vector containing Voivodship objects
        struct Candidates{  //Singly linked list of Candidate objects.
            Candidate* candidate; //Candidate object.
            Candidates* next; //Next Candidate instance.
        };
        Candidates* headC; //Head of the singly linked list for candidates.
    public:
        //Constructor for the election
        Election(const vector<Voivodship*> voivodship);
        //Destructor for Election class.
        ~Election();
        //Determines an election winner and displays total support of each candidate in percantages.
        void determine_winner();
        //Shows the backing of candidates with respect to the age group of voters. Age(18-39]= Young adults, Age[40,65)= Middle aged, Age(65,..)= Elders.
        void support_by_age_group();
        //Appends the candidate to the structure of Candidates following prior validations. Takes Candidate object to be added as an argument.
        bool register_candidate(Candidate* candidate);
        //Displays all registered candidates(without their support).
        void display_registered_candidates();
        //Sends the list of candidates to all the voivodships.
        void distribute_candidates_to_voivodships();
        //Returns the election attendance in percantage
        double election_attendance();
};
```

```
class Voter{
    private:
        bool validity;//boolean validity of a voter, indicating registartion status, by default false
        char* name;//Name of a voter.
        unsigned int age;//Age of a voter.
        bool vote;//Boolean status of vote submission.
        char* voivodship;//Name of a voivodship the voter lives in.
    public:
        //Constructor of voter, taking up the private variables. The vote submission status is assumed to be false at object creation.
        Voter(const char* name, const unsigned int age,const char* voivodship,bool vote=false,const bool validity=false);
        //Destructor for voter instance.
        ~Voter();
        //Copy contrcutor for voter instance
        Voter(const Voter& voter);
        //Submits the vote for a candidate, increasing his backing and setting the vote submission status to true.
        void submit_vote(Candidate& candidate);
        //Returns the age of a voter.
        unsigned int get_age()const;
        //Returns the vote submission status.
        bool& refValidity();
        //Displays the voting status
        bool& has_voted();
        //Displays the voters information.
        friend ostream& operator<<(ostream& os,const Voter& voter);
        //Returns the name of a voter
        char* get_name()const;
        //Returns the voivodship of voter
        char* get_voivodship()const;

};
```

```
//Candidate inherits from Voter class as candidates do have the voting rights, despite part-taking in an election.
class Candidate:public Voter{
    private:
        unsigned int support; //Numerical value representing the amount of votes  a candidate has gathered. At creation initialized to zero.
        struct Supporters{ //Singly linked list of voters who submitted the votes on the candidate.
            Voter* voter; //Voter instance.
            Supporters* next; //Next supporters instance.
        };
        Supporters* headS; //Head of the supporters structure.
    public:
        //Constructor for candidate object, inherits from voter instance
        Candidate(const char* name, const unsigned int age, const char* voivodship, const bool vote = false, const bool validity = false, const int support = 0);
        //Destructor for candidate instance.
        ~Candidate();
        //Vote submission, increasing backing
        void submit_vote();
        //Returns the support in a given voivodship
        unsigned int local_support(const char* voivodship);
        //Returns the candidate's support as a reference
        unsigned int& ref_support();
        //Displays all voters that submitted their vote on the candidate.
        void display_voters();
        //Appends a voter to the supporters list.
        void add_supporter(Voter* voter);
        //Frees the list of supporters
        void free_supporters();
        //Function returning support with respect to age group
        vector<int> age_distribution();
        //operator<< for candidate information
        friend ostream& operator<<(ostream& os,const Candidate& candidate);
};
```

```
 class Voivodship{
    private:
        char* name; //Name of the voivodship.
        unsigned int citizens; //Number of all citizens living within the voidvodship. The number voters can not exceed this number.
        struct Voters{ //Singly linked list of all registered voters(who submitted their vote).
            Voter* voter; //Voter instance.
            Voters* next; //Next Voters entry.
        };
        Voters* headV; //Head of singly linked list of all registered voters.
        vector<Candidate*> localVotes; //Map of Candidates and their backing WITHIN THE VOIVODSHIP.
    public:
        //Constructor for the voivodship instance.
        Voivodship(const char* name, const unsigned int citizens);
        //Destructor for voivodship object.
        ~Voivodship();
        //Appends the voter to the registerd voters.
        bool register_voter(Voter* voter);
        //Displays all registered voters within a voivodship.
        void display_registered_voters();
        //Iterates thorugh singly linked list and returns
        bool find(const char* name,const unsigned int age);
        //Displays the local support of each candidate in percantages.
        void display_local_support();
        //Returns the number of all registered voters.
        unsigned int number_of_voters();
        //Returns the number of citizens
        unsigned int number_of_citizens();
        //Returns the name of the voivodship(private memeber)
        char* get_name();
        //Constructs a local map for candidates and their backing.
        void register_candidate(Candidate* candidate);
 };

 #endif
```

IV. Functional test cases

```cpp
#include <iostream>
#include <vector>
#include <map>
#include "classes.h"
using namespace std;

/*
Assumptions and Restrictions:
R1. Voter must be at least 18 years of age to vote.
R2. The number of voters cannot outnumber the number of citizens of voivodship.
R3. Each vote has the same weight and can be submitted only once.
R4. Candidates must be at least 35 years of age to part-take.
R5. Candidates can also be voters, they can vote.
R6. Candidates cannot have the number of votes greater than the total amount of votes.
*/


int main(){
    Voter* voter1 = new Voter("Marvin Beak", 18, "Masovian");
    Voter* voter2 =new Voter("Agent 5.5",24,"Lesser Poland");
    Voter* voter4=new Voter("Cheese",16,"Lesser Poland");//Invalid args
    Voter* voter5=new Voter("Jim Pork",20,"Great Poland");
    Voter* voter6=new Voter("Bob Bacon",21,"test");
    Voivodship* v1 = new Voivodship("Masovian", 100);
    Voivodship* v2 = new Voivodship("Lesser Poland", 40);
    Voivodship* v3 = new Voivodship("Great Poland", 20);
    Voivodship* v4 = new Voivodship("test", 0);
    if(v4->register_voter(voter6)) cerr<<"Number of citizens is smaller than amount of registered voters"<<endl;
    if(!v1->register_voter(voter1)){
        cerr<<"Valid voter not registered correctly"<<endl;
    }else{
        if(v1->register_voter(voter1))
            cerr<<"Voter registered twice to a single voivodship"<<endl;
    }
    if(v2->register_voter(voter1)){
        cerr<<"Voter assigned to a different voivodship"<<endl;
    }
    if(v2->register_voter(voter4)){
        cerr<<"Voter of age<18 registered successfully RESTRICTION R1"<<endl;
    }
    v2->register_voter(voter2);
    v3->register_voter(voter5);
```

```cpp
    if(v1->number_of_voters()!=1) cerr<<"Invalid number of voters in v1"<<endl;
    if(v2->number_of_voters()!=1) cerr<<"Invalid number of voters in v2"<<endl;
    if(v3->number_of_voters()!=1) cerr<<"Invalid number of voters in v3"<<endl;

    v1->display_registered_voters(); //Expected information of voter1
    v2->display_registered_voters(); //Expected information of voter2
    v3->display_registered_voters(); //Expected information of voter5
    vector<Voivodship*> voivodships={v1,v2,v3};
    Election e(voivodships);
    Candidate* c1=new Candidate("Tim Cheese",36,"Masovian");
    Candidate* c2=new Candidate("John Pork",34,"Lesser Poland");
    Candidate* c3=new Candidate("Tiger Sam",40,"Great Poland");
    if(!e.register_candidate(c1)){
        cerr<<"Valid candidate not registered successfully"<<endl;
    }if(e.register_candidate(c2)){
        cerr<<"Candidate with invalid age registered successfully RESTRICTION R4"<<endl;
    }
    e.register_candidate(c3);
    e.display_registered_candidates(); //Expected information abot c1, c3
    e.distribute_candidates_to_voivodships();

    c1->submit_vote();
    if(c1->ref_support()!=1) cerr<<"Candidates support not incremented after self vote"<<endl;
    c1->submit_vote();
    if(c1->ref_support()==2) cerr<<"Candidates vote status not changed after initial self vote"<<endl;

    voter1->submit_vote(*c1);
    if(c1->ref_support()!=2) cerr<<"Invalid support of candidate c1 after valid Voter vote cast"<<endl;
    v1->display_local_support(); //Expected: Masovian: Tim Cheese: 100% Tiger Sam 0%
    voter2->submit_vote(*c3);
    v2->display_local_support(); //Expected: Lesser Poland: Tiger Sam: 100%
    voter5->submit_vote(*c1);
    v3->display_local_support();// Expected: Great Poland: Tim Cheese: 100%

    e.determine_winner(); //Expected: Tim Cheese:75%, Tiger Sam: 25%
    //Election attendance: 3/160 *100(This is in percantages,160 is sum of citizens, 3 is the counter), which is 1.875%
    e.support_by_age_group(); //Expected: Young adults:100% Middle aged:0% Elders:0% for all candidates
    return 0;
}
```