Lab 4 - Exceptions Handling

Introduction

This project is an extension of Lab 3 which was about designing an object-oriented voting system, written in java, that will illustrate the principles of object-oriented programming. With this lab we will illustrate the use of exceptions.

Classes

We will be enhancing the classes which we created before **(changes in highlighted in yellow)**, as well as adding new classes for exceptions (this will hold the custom exceptions we defined).

Person (No changes)

This will be an abstract class. In this class we have used the concept of inheritance as the candidate class and voter class are child classes so they can access the methods of parent class.

Must have the following variables:

- age (int) - private
- gender (char) - private, will hold values like M, F, T, U
- firstName (string) - protected
- lastName (string) - protected
- politicalParty (string) - protected, will hold values like Democrat, Republican, Non-Affiliate

Must have the defined constructor:

- Person ( )
- Person (int age, char gender, String firstName, String lastName, String politicalParty) Which will set the passed properties to our defined variables

Will implement the following methods:

- getAge() - returns int
- getGender() - returns char

Will define the following abstract methods:

- getFullName() - returns string

Candidate

This class holds the information about the candidate it will extend the class Person.

Must have the following variables:

- voteCount (int) - private, when the class is instantiated this should default to 0

Must have the defined constructor:

- Candidate ( )
- Candidate (int age, char gender, String firstName, String lastName, String politicalParty) Which will set the passed properties to our defined variables and super class
- Throws exception **MinimumAgeException** if candidate's age is less than 25
    - The message is "Candidate's age cannot be less than 25"

Which will set the passed properties to our defined variables and super class

Will implement the following public methods:

- getVoteCount() - returns int
- incrementVoteCount() - void, increments the voteCount
- getFullName() - returns string
    - If the candidates firstName = "John" and lastName = "Smith" and political party is "Democrat" then it returns "John Smith - D"
    - If the candidates firstName = "John" and lastName = "Smith" and political party is "Republican" then it returns "John Smith - R"
    - If the candidates firstName = "John" and lastName = "Smith" and political party is "Non-Affiliate" then it returns "John Smith - NA"
    - Otherwise "John Smith"

Voter

This class will gather the information about the voter it will extend the class Person.

Must have the following variables:pub

- voterId (int) - private

● voted (boolean) - private, when the class is instantiated this should default to false

Must have the defined constructor:

● Voter ( )

● Voter (int voterId, int age, char gender, String firstName, String lastName, String politicalParty)

● Throws exception **MinimumAgeException** if voter's age is less than 18

- The message is "Voter's age cannot be less than 18"

Which will set the passed properties to our defined variables and super class

Will implement the following public methods:

● getVoterId() - returns int

● hasVoted() - returns boolean

● voted() - void, sets the voted flag to true

● getFullName() - returns string - e.g. firstName = "John" and lastName = "Smith" then it returns "John Smith"

Voting Machine

This class is used to manipulate and print the results (i.e. who is the winner and how many votes he/she gained). This class has crucial role in whole system

Must have the following variables:

~~● candidates (Candidate[ ]) - public, array of all candidates in the race~~

● candidates (**ArrayList<Candidate>**) – public ArrayList of all candidates in the race

Must have the defined constructor:

~~● VotingMachine (Candidate candidates[])~~

● VotingMachine (**ArrayList<Candidate>** candidates)

● Throws exception **MissingCandidateException** when candidates is null or empty. The message will be "The candidate list cannot be null or empty."

Which will set the passed properties to our defined variables

Will implement the following public methods:

● vote(Voter v, Candidate c) - void, counts vote (increments candidate count) and marks voter as voted

- Will throw **CandidateNotFoundException** if voter votes for a candidate not in the race. Message will be "Candidate does not exists in the candidates collection."
- We will not handle the scenario that a voter votes for a candidate not in the election

● vote(Voter v) - void, if voter has not voted, marks voter as voted.

● tally() - void, prints results and winner.

For example we have the following candidates

(Sam Smith - R, Joe Dean - D, Jane Doe - NA) the out print will be like

Sam Smith - R has 5 votes.

Joe Dean - D has 2 votes.

Jane Doe - NA has 7 votes.

Jane Doe - NA won with 7 votes.

We will not handle the case for ties at the moment, so the first candidate to have the most votes wins.

MinimumAgeException

Will be in the exceptions package. This will extend Exception.

Must have the defined constructor:

- MinimumAgeException (String message)

CandidateNotFoundException

Will be in the exceptions package. This will extend Exception.

Must have the defined constructor:

- CandidateNotFoundException(String message)

MissingCandidatesException

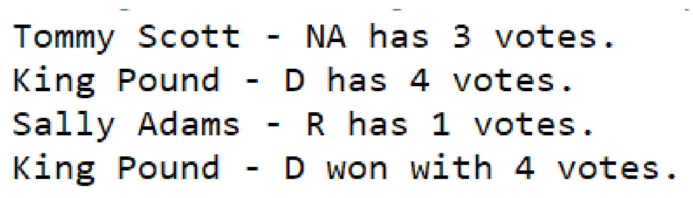
Will be in the exceptions package. This will extend Exception.

Must have the defined constructor:

- MissingCandidatesException(String message)

Tests

You can test your codes by compiler and execute the provided file-
ElectionLab4.java You will see the printout as



```
Tommy Scott - NA has 3 votes.
King Pound - D has 4 votes.
Sally Adams - R has 1 votes.
King Pound - D won with 4 votes.
```

Submission

After completing this lab, zip up all the .class, and .java files in the name of the zip file must be COP3809_SYY_Lab4_XXXX.zip; where YY is 01, 02 or 03 and XXXX is your student Id.