Name:	QUID:

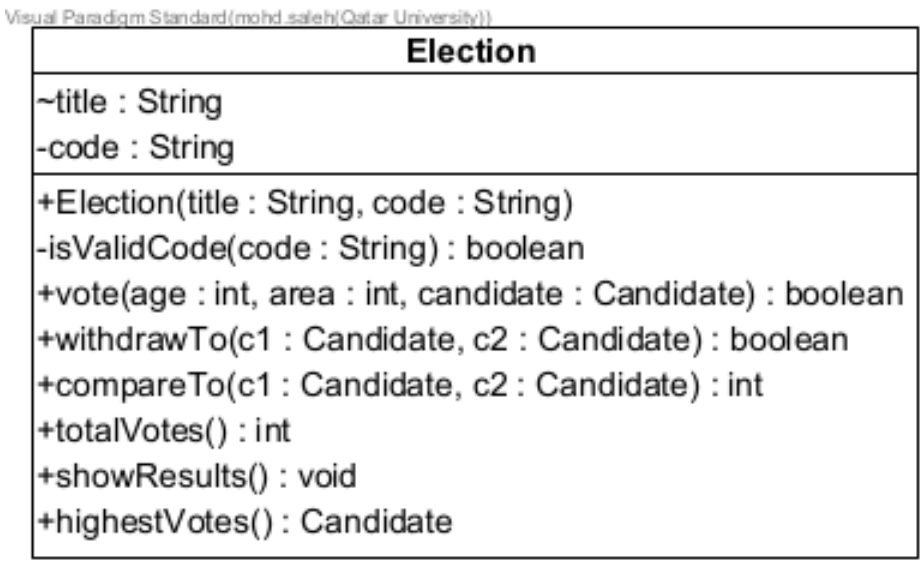
Some methods of the String, Character, and Integer classes

charAt(int index): char – String	Returns the char value at the specified index.
indexOf(int ch): int – String	Returns the index within this string of the first occurrence of the specified character.
indexOf(String str): int – String	Returns the index within this string of the first occurrence of the specified substring.
subString(int beginIndex): String – String	Returns a string that is a substring of this string.
subString(int beginIndex, int endIndex): String – String	Returns a string that is a substring of this string.
startsWith(String prefix): boolean – String	Tests if this string starts with the specified prefix.
endsWith(String suffix): boolean – String	Tests if this string ends with the specified suffix.
contains(String charSequence): boolean – String	Returns true if and only if this string contains the specified sequence of char values.
length(): int – String	Returns the length of this string.
isBlank(): boolean – String	Returns true if the string is empty or contains only white space codepoints, otherwise false.
isEmpty(): boolean – String	Returns true if, and only if, length() is 0.
strip(): String – String	Returns a string whose value is this string, with all leading and trailing white space removed.
trim(): String – String	Returns a string whose value is this string, with all leading and trailing space removed, where space is defined as any character whose codepoint is less than or equal to 'U+0020' (the space character).
toLowerCase(): String – String	Converts all of the characters in this String to lower case using the rules of the default locale.
toUpperCase(): String – String	Converts all of the characters in this String to upper case using the rules of the default locale.
static valueOf(double d): double - String	Returns the string representation of the double argument.
static valueOf(float d): float - String	Returns the string representation of the float argument.
static valueOf(int d): int - String	Returns the string representation of the int argument.
static valueOf(char d): char - String	Returns the string representation of the char argument.
static getNumericValue(char ch): int – Character	Returns the int value that the specified Unicode character represents.
static isDigit(char ch): boolean – Character	Determines if the specified character is a digit.
static isLetterOrDigit(char ch): boolean – Character	Determines if the specified character is a letter or digit.
static isLetter(char ch): boolean – Character	Determines if the specified character is a letter.
static isWhiteSpace(char ch): boolean – Character	Determines if the specified character is white space according to Java.
static isTitleCase(char ch): boolean – Character	Determines if the specified character is a titlecase character.
static isUpperCase(char ch): boolean – Character	Determines if the specified character is an uppercase character.
static isLowerCase(char ch): boolean – Character	Determines if the specified character is a lowercase character.
static parseInt(String str): int – Integer	Parses the string argument as a signed decimal integer.

1. **[15 POINTS]** Consider a very simplified election system in which there are predefined candidates in the enum **Candidate** who are running for elections in ***area*** number **3**. The Election class has a constructor to initialize the attributes **title** and a **code**. The **code** must be verified before it is set. The class **Election** has methods that you need code based on the comments provided for each method. The class **ElectionTester** runs randomized voting for an election. You need to provide the necessary code for each of the comments in the class **ElectionTester**.

```
public enum Status {
    ACTIVE,WITHDRAWN
}
```

```
public enum Candidate{
    CANDIDATE1("Huda",3),CANDIDATE2("Khalid",3),
    CANDIDATE3("Hind",3),CANDIDATE4("Ahmed",3),
    CANDIDATE5("Iman",7),CANDIDATE6("Qais",7);
    String name;
    int area;
    int votes;
    Status status = Status.ACTIVE;
    private Candidate(String name, int area) {
        this.name=name;
        this.area=area;
    }
}
```



Name:	QUID:

```
public class Election {
      String title;
      private String code;
      /*Code the fully parameterized constructor to initialize the attributes title
       * and code. The code must be verified so call setCode to takes care of that*/
```

6 Points

```
      /*Code a getter for the attribute code*/
```

3 Points

```
      public void setCode(String code) {
            if(isValidCode(code))
                  this.code=code;
      }//End of Method
      public boolean vote(int age,int area,Candidate candidate) {
      /*Code the body of this method. The age and area represent the age and
       * area of the voter, and the candidate is the one to vote for.
       * The voter's age must be above 18 and lives in the same area of the candidate
       * and if so, the votes of candidate are increased by one and true is
       * returned. Otherwise, voting is not successful and false is returned.
       */
```

6 Points

```
      }//End of Method
```

6 Points

```
public boolean withdrawTo(Candidate c1, Candidate c2) {
/*Code the body of this method which adds the votes of candidate c1 to
 * to candidate c2, sets the votes of c1 to zero, sets status of c1 to
 * WITHDRAWN and return true. Note that both c1 and c2 must be ACTIVE and
 * in the same area for this method to do that. Otherwise, false is returned.*/
```

```
}//End of Method
```

6 Points

```
public int compareTo(Candidate c1, Candidate c2) {
/*Code the body of this method which uses shorthand if to compare the
 * votes of c1 to that of c2*/
```

```
}//End of Method
```

6 Points

```
public int totalVotes() {
/*Code the body of this method that finds and returns the sum of all
 * valid votes.*/
```

```
}//End of Method
```

6 Points

```
public void showResults() {
/*Code the body of this method which loops on all candidates and prints
 * each candidate's info. including: name, area, and number of votes.
 * Finally, the method prints the sum of valid votes.
 * The format of printing candidates info. looks something like this:
 * Candidate Hind in area 3 has 8532 votes.
 */
     System.out.println("Election Results:");









}//End of Method
public Candidate highestVotes() {
/*Code the body of this method which loops on all candidates to find the
 * candidate having the highest number of votes and returns that candidate.*/









} //End of Method
```

6 Points

10 Points

```
public void showAreaWinner(int area) {
/* Code the body of this method which loops on all candidates to find the
 * ACTIVE candidate having the highest number of votes in the specified
 * area and displays the information in the formats:
 * "Qais won the elections in area 7 having 17000 votes, 51.73% of all votes".
 * Display error message if no such ACTIVE candidate in the specified area.
 * You cannot assume knowledge of internal source code of the enum Candidate.*/









}//End of Method
```

6 Points

```
    private boolean isValidCode(String code) {
    /* Code the body of this method. In a valid code, the first character is an
     * upper case letter followed by one lower case letter, followed by five digits,
     * followed by the string HCq00, followed by 3 to 8 alphanumeric characters*/









    }//End of Method
}//End of Class Election
```

```
import java.util.Random;
import java.util.Scanner;
public class ElectionTester {
      public ElectionTester() {
```

6 Points

```
/*Create a Scanner object kb and use it to read the title and code strings*/
```

3 Points

```
/*Create election as an object of Election using the read title and code*/
```

3 Points

```
/*Retrieve the values of title and code from the object election and display
* on one line the title 30 spaces and the code 15 spaces both left justified*/
```

```
      // Randomize voting attempts of 1000000 voters
            Random r = new Random();
            int age, area;
            Candidate candidate;
            for (int i = 0; i < 1000000; i++) {
```

6 Points

```
/*Use the Random object r to create random data for age [10,100] and area [1,9].
* I have already used r to randomly select a candidate. Use all of this random
 * data to call the method vote using the object election.*/
                  int index = r.nextInt(0, Candidate.values().length);
                  candidate = Candidate.values()[index];
```

```
            } //End of for loop
```

3 Points
```
/*Call an appropriate method of object election to display the results*/
```

3 Points
```
/*Call an appropriate method of object election get the candidate Huda to
* withdraw to the candidate Hind*/
```

3 Points
```
/*Call an appropriate method of object election to display the results*/
```

6 Points
```
/*Call an appropriate method of object election to get the candidate with the
* highest number of votes then display the name, area, and votes of for it*/
```

3 Points
```
/*Call an appropriate method of object election to display information about the
 * winner of area 7*/
```

```
    }//End of Constructor
    public static void main(String[] args) {
```

3 Points
```
    /*Create an anonymous object of the class ElectionTester*/
```

```
    }//End of Method
}//End of Class ElectionTester
```