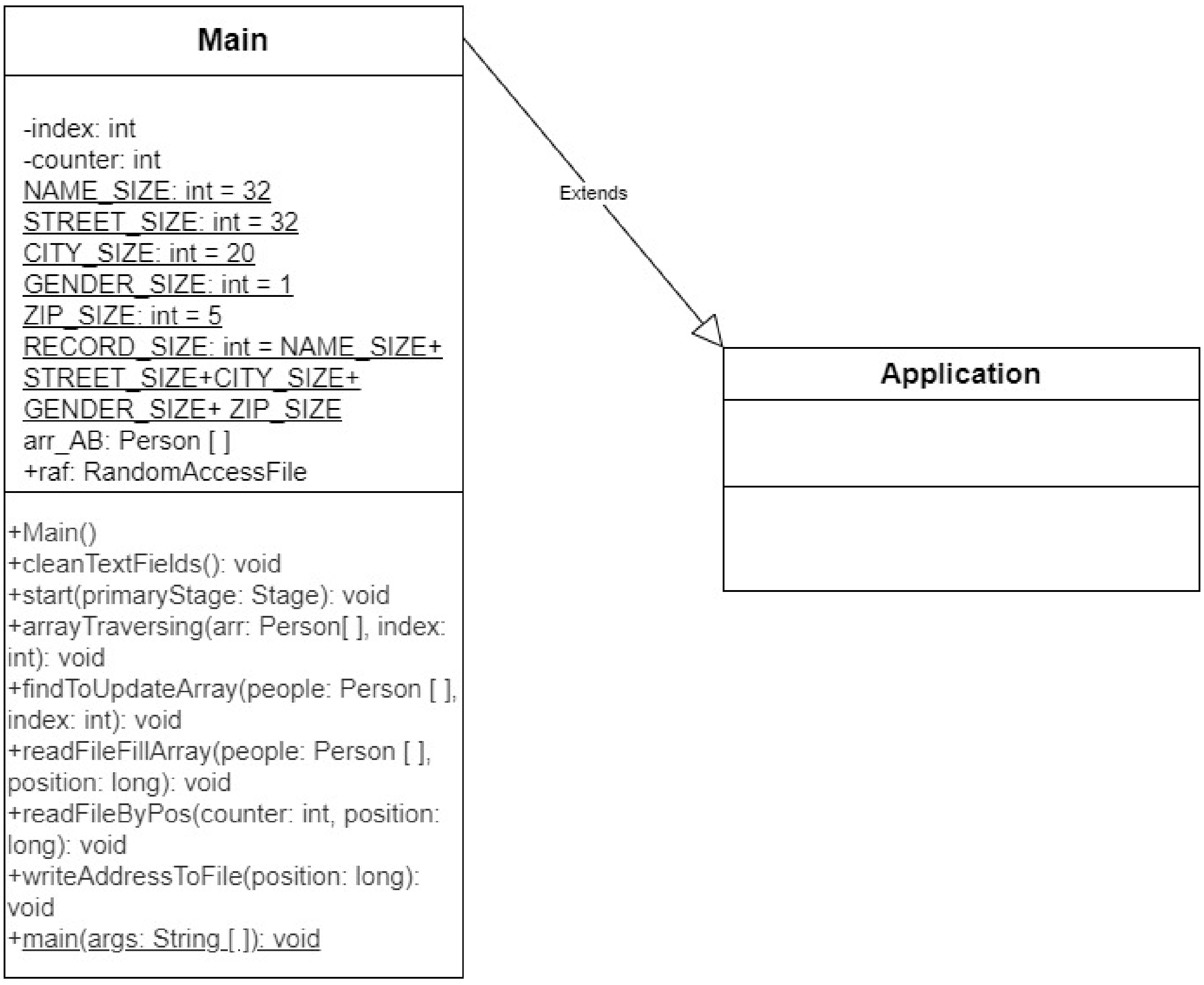
Aydın Adnan Menderes University
Department of Computer Engineering, 2020-2021/Fall
CSE203 Object Oriented Programming
Student ID: 181805031 / Formal Education
Student Name Surname: Fatma Elvan Sarıaydın
Student ID: 191805070/ Formal Education
Student Name Surname: Seray Karaefe

Object Oriented Programming Fall Semester Project Report

Aydın Adnan Menderes University
Department of Computer Engineering, 2020-2021/Fall
CSE203 Object Oriented Programming
Student ID: 181805031 / Formal Education
Student Name Surname: Fatma Elvan Sarıaydın
Student ID: 191805070/ Formal Education
Student Name Surname: Seray Karaefe

Introduction
Introduction in this report, we will try to explain the classes that construct our project, how we create the members of these classes and how they connect with each other. We used UML diagrams for the visualization of classes.

Classes
Main Class
This is the class which includes public static void main (String [ ] args) method, data fields and other methods to create and operate the GUI.


Main
-index: int
-counter: int
NAME_SIZE: int = 32
STREET_SIZE: int = 32
CITY_SIZE: int = 20
GENDER_SIZE: int = 1
ZIP_SIZE: int = 5
RECORD_SIZE: int = NAME_SIZE+ STREET_SIZE+CITY_SIZE+ GENDER_SIZE+ ZIP_SIZE
arr_AB: Person [ ]
+raf: RandomAccessFile
+Main()
+cleanTextFields(): void
+start(primaryStage: Stage): void
+arrayTraversing(arr: Person[ ], index: int): void
+findToUpdateArray(people: Person [ ], index: int): void
+readFileFillArray(people: Person [ ], position: long): void
+readFileByPos(counter: int, position: long): void
+writeAddressToFile(position: long): void
+main(args: String [ ]): void
Extends
Application

Aydın Adnan Menderes University
Department of Computer Engineering, 2020-2021/Fall
CSE203 Object Oriented Programming
Student ID: 181805031 / Formal Education
Student Name Surname: Fatma Elvan Sarıaydın
Student ID: 191805070/ Formal Education
Student Name Surname: Seray Karaefe

1.Data Fields (Attributes)

1.1 private int index
This data field keeps the number of records and it helps for traversing in array with private int counter variable.

1.2 private int counter
This data field helps for traversing the array. Its value changes when click the buttons (first,next,previous,last).

1.3 Person [ ] arr_AB
This array is created for objects. Once the program runs, this array is filled by the data from the file and used for when traversing and changing the data.

1.4 final static int NAME_SIZE
This data field keeps the size of the name.

1.5 final static int STREET_SIZE
This data field keeps the size of the street.

1.6 final static int CITY_SIZE
This data field keeps the size of the city.

1.7 final static int GENDER_SIZE
This data field keeps the size of the gender.

1.8 final static int ZIP_SIZE
This data field keeps the size of the zip.

1.9 final static int RECORD_SIZE
This data field keeps the size of the record.

1.10 public RandomAccessFile raf
This data field is our file. We will write the data into it and read the data from it.

2. Methods (Behaviors)

2.1 public Main()
This is our constructor. It has raf assignment and arr_AB assignment wrapped by a try catch block in it.

2.2 public void cleanTextFields()
This method deletes the data on the text fields.

2.3 public void start(Stage primaryStage)

This method has a try catch block and in try block we set the text fields' properties such as width and hight. We create a GridPane object and design inside of it. Also we create a HBox object and design inside of it. Then we construct grid pane and horizontal box together by a border pane. After that, we create an Scene object and put border pane inside of it. Our scene is ready. Finally, we set scene for the stage. Method also has one more try catch block which runs when the program starts and calls a method to fill the array with the data which is read from the file. Besides, it calls another method to show first record on the screen. Then, we set our buttons' actions. We use an event handler called setOnAction(e->{}); for the buttons' events. Also there is a counter for buttons to use. btAdd's action is adding one more record into the file and array. If next adding is out of the boundry of the array an alert shows up and button action does not add any records and text fields are cleaned. Also if the text fields for adding are empty, an alert shows up and warns for required text fields which are name,street and gender. btFirst's action is always showing the first record. btNext's action is showing the next record. If there is not any records in next, an alert shows up an the same record remains. btPrevious's action is showing the previous record on the screen.If there is not any record before in use record, an alert shows up and the same record remains. btLast's action is showing the last record on the screen.
btSearchByID button takes the id from tfSearchUpdateID text field to find the relevant record. If the id searched by the user is bigger than the index value an alert shows up.
btFirst,btNext,btPrevious,btLast and btSearchByID buttons call the arrayTraversing(Person [ ] arr, int index) method to make their actions. btUpdateByID calls the findToUpdateArray(Person [ ] people,int index) method and sends arr_AB and the relevant record's id. Finally btCleantextFields button's action is calling cleanTextFields() method to clean all text fields.

2.4 public void arrayTraversing(Person [ ] arr, int index)

This method is used for traversing in the array. It gets the id,name,street,city,gender and zip data by using arr_AB array and Person class' getter methods and sets text fields with the relevant data.

2.5 public void findToUpdateArray(Person [ ] people, int index)

This method finds the relevant record in the array and sets the textfields by getting text fields's entered data and calls writeAddressToFile(long position) method to write the same data to the file. All the text fields do not have to be changed but if you want to change all of the data of the relevant record (except id), you can make this change.

2.6 public void readFileFillArray(Person [ ] people, long position)

This method is used for filling the array which is given to the method. When the program starts to run, method is invoked by the relevant try catch block. This method is also invoked by btAdd button to add a new record into the array's last record's next. This method also uses position (`raf.seek(position);`) to find the right position in the file.

2.7 public void readFileByPos(int counter, long position)

When the program starts to run, this method is invoked by the relevant try catch block to show the first record. This method also uses position (`raf.seek(position);`) to find the right position in the file

2.8 public void writeAddressToFile(int counter, long position)

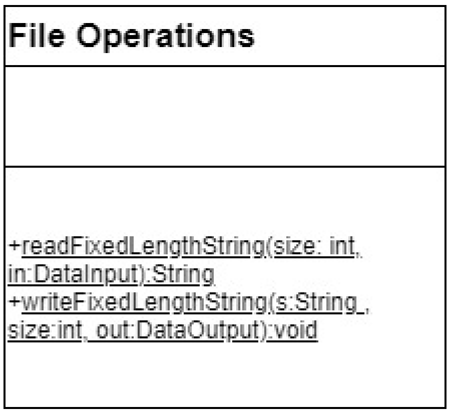
This method invokes the method `writeFixedLengthString(String s,int size,DataOutput out)` of the FileOperations class to write the data recevied from the text fields into the file. This method also uses position (`raf.seek(position);`) for writing relevant data into the right position in the file.

2.9 public static void main(String [ ] args)

This is the main method which is every java application has for running the program.

FileOperation Class

This class helps the Main class to read from the file and to write into the file.


File Operations
+readFixedLengthString(size: int, in:DataInput):String
+writeFixedLengthString(s:String , size:int, out:DataOutput):void

1.Data Fields (Attributes)

There is not any data fields.

2.Methods (Behaviors)

2.1 public static String readFixedLengthString(int size, DataInput in)

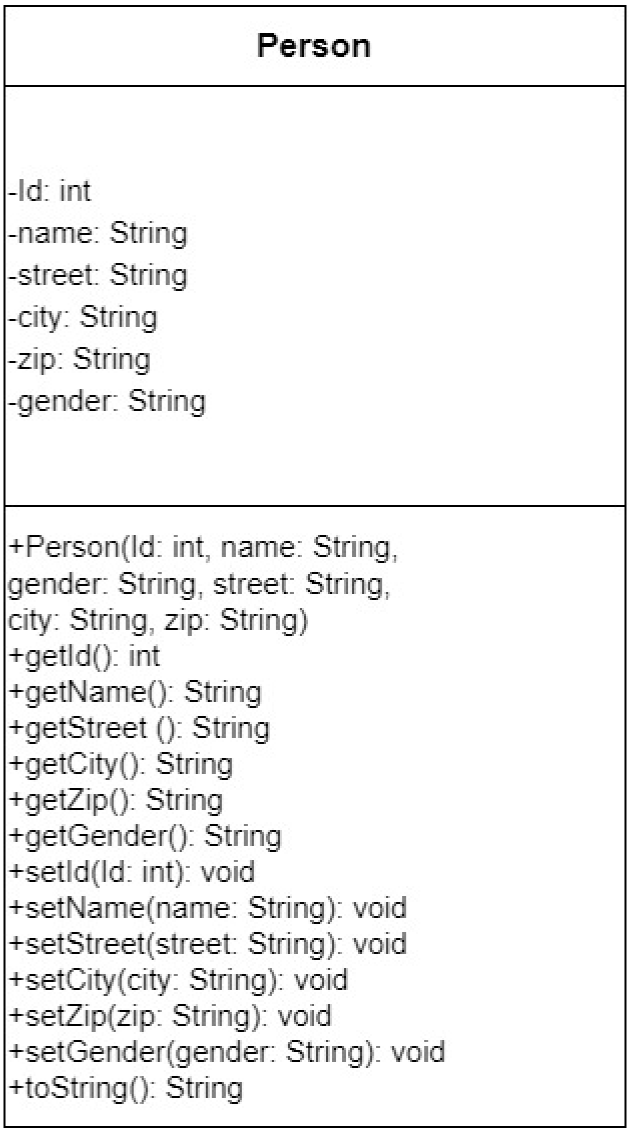
This method takes 2 parameters from the Main class. It creates an array by using the given size to read the file and returns it to the Main class.

2.2 public static void writeFixedLengthString(String s, int size, DataOutput out)

This method is used for writing the given size of the data into the file. It creates an array by using the given size.If the given String s is less than the given size it fills rest of the size with " " (blanks).

Person Class

This class is used for creating Person objects and their properties. It has getter and setter methods for making operations on the object.


Person
-Id: int
-name: String
-street: String
-city: String
-zip: String
-gender: String
+Person(Id: int, name: String, gender: String, street: String, city: String, zip: String)
+getId(): int
+getName(): String
+getStreet (): String
+getCity(): String
+getZip(): String
+getGender(): String
+setId(Id: int): void
+setName(name: String): void
+setStreet(street: String): void
+setCity(city: String): void
+setZip(zip: String): void
+setGender(gender: String): void
+toString(): String

1.Data Fields (Attributes)

1.1 private int Id

This field is used for object's id. It is changeable.

1.2 private String name

This field is used for object's name. It is changeable.

1.3 private String street
This field is used for object's street. It is changeable.

1.4 private String city
This field is used for object's city. It is changeable.

1.5 private String zip
This field is used for object's zip. It is changeable.

1.6 private String gender
This field is used for object's gender. It is changeable.

2.Methods (Behaviors)
2.1 public Person(int id, String name, String gender, String street, String city, String zip)
This is the constuctor method. It creates a Person object. It makes relevant assignments for the created object.

2.2 public int getId()
This method returns the relevant object's id.

2.3 public String getName()
This method returns the relevant object's name.

2.4 public String getStreet()
This method returns the relevant object's street.

2.5 public String getCity()
This method returns the relevant object's city.

2.6 public String getZip()
This method returns the relevant object's zip.

2.7 public String getGender()
This method returns the relevant object's gender.

2.8 public void setId(int Id)
This method sets the given parameter to the relevant object's id. But this method is not available to use because we disabled the id changing by the user.

2.9 public void setName(String name)
This method sets the given parameter to the relevant object's name.

2.10 public void setStreet(String street)
This method sets the given parameter to the relevant object’s street.

2.11 public void setCity(String city)
This method sets the given parameter to the relevant object’s city.

2.12 public void setZip(String zip)
This method sets the given parameter to the relevant object’s zip.

2.13 public void setGender(String gender)
This method sets the given parameter to the relevant object’s gender.

2.14 public String toString()
This method returns id, name, street, city, gender and zip variables of the relevant object by using getter methods.

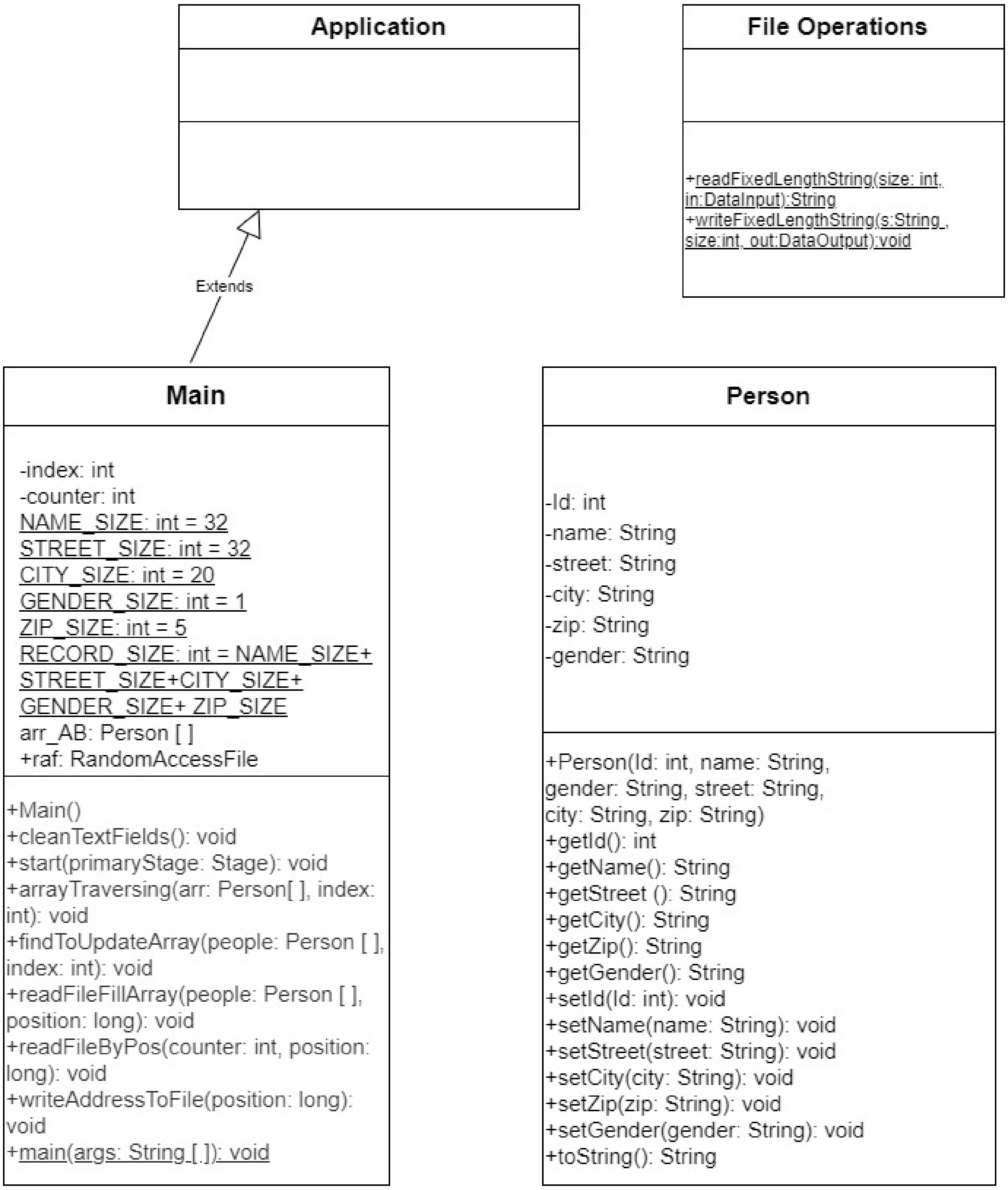
Test

Our program can keep 100 records. We test it by decrasing that number to 10. But this was just for making the test more quick. As we tested it, if we say that this program can keep 10 records, we saw the program can achieve it and operates all methods and buttons without any error. Because all the possible errors are prevented by the alerts.

Aydın Adnan Menderes University
Department of Computer Engineering, 2020-2021/Fall
CSE203 Object Oriented Programming
Student ID: 181805031 / Formal Education
Student Name Surname: Fatma Elvan Sarıaydın
Student ID: 191805070/ Formal Education
Student Name Surname: Seray Karaefe

UML Diagram

Here is our UML diagram.


Application
File Operations
+readFixedLengthString(size: int, in:DataInput):String
+writeFixedLengthString(s:String , size:int, out:DataOutput):void
Extends
Main
-index: int
-counter: int
NAME_SIZE: int = 32
STREET_SIZE: int = 32
CITY_SIZE: int = 20
GENDER_SIZE: int = 1
ZIP_SIZE: int = 5
RECORD_SIZE: int = NAME_SIZE+ STREET_SIZE+CITY_SIZE+ GENDER_SIZE+ ZIP_SIZE
arr_AB: Person [ ]
+raf: RandomAccessFile
+Main()
+cleanTextFields(): void
+start(primaryStage: Stage): void
+arrayTraversing(arr: Person[ ], index: int): void
+findToUpdateArray(people: Person [ ], index: int): void
+readFileFillArray(people: Person [ ], position: long): void
+readFileByPos(counter: int, position: long): void
+writeAddressToFile(position: long): void
+main(args: String [ ]): void
Person
-Id: int
-name: String
-street: String
-city: String
-zip: String
-gender: String
+Person(Id: int, name: String, gender: String, street: String, city: String, zip: String)
+getId(): int
+getName(): String
+getStreet (): String
+getCity(): String
+getZip(): String
+getGender(): String
+setId(Id: int): void
+setName(name: String): void
+setStreet(street: String): void
+setCity(city: String): void
+setZip(zip: String): void
+setGender(gender: String): void
+toString(): String