

## DSA Project

Clique of people

Group: G612277

Brajan Miśkowicz

Adam Szałański

Łucja Pałkus



---

<b>1 User Documentation</b>	<b>1</b>
<b>2 UML Diagram</b>	<b>3</b>
<b>3 Namespace Index</b>	<b>5</b>
3.1 Package List . . . . .	5
<b>4 Class Index</b>	<b>7</b>
4.1 Class List . . . . .	7
<b>5 File Index</b>	<b>9</b>
5.1 File List . . . . .	9
<b>6 Namespace Documentation</b>	<b>11</b>
6.1 Package fileHandling . . . . .	11
6.2 Package people . . . . .	11
6.3 Package relationships . . . . .	11
<b>7 Class Documentation</b>	<b>13</b>
7.1 fileHandling.FileHandling Class Reference . . . . .	13
7.1.1 Detailed Description . . . . .	13
7.1.2 Member Function Documentation . . . . .	13
7.1.2.1 readFile() . . . . .	13
7.1.2.2 writeToFile() . . . . .	14
7.2 Menu Class Reference . . . . .	15
7.2.1 Detailed Description . . . . .	15
7.2.2 Member Function Documentation . . . . .	15
7.2.2.1 main() . . . . .	15
7.2.2.2 menu() . . . . .	16
7.3 people.People Class Reference . . . . .	17
7.3.1 Constructor & Destructor Documentation . . . . .	18
7.3.1.1 People() . . . . .	18
7.3.2 Member Function Documentation . . . . .	18
7.3.2.1 addPersonFromString() . . . . .	18
7.3.2.2 findPersonById() . . . . .	19
7.3.2.3 getPeople() . . . . .	19
7.3.2.4 removePersonById() . . . . .	20
7.3.2.5 setPeople() . . . . .	21
7.3.2.6 writeToFile() . . . . .	21
7.3.3 Member Data Documentation . . . . .	22
7.3.3.1 FIRST_LINE_PEOPLE . . . . .	22
7.4 people.Person Class Reference . . . . .	23
7.4.1 Constructor & Destructor Documentation . . . . .	24
7.4.1.1 Person() . . . . .	24

7.4.2 Member Function Documentation	25
7.4.2.1 equals()	25
7.4.2.2 getBirthdate()	26
7.4.2.3 getBirthplace()	26
7.4.2.4 getFilms()	26
7.4.2.5 getGender()	27
7.4.2.6 getGroupCode()	27
7.4.2.7 getHome()	27
7.4.2.8 getIdPerson()	27
7.4.2.9 getLastName()	28
7.4.2.10 getName()	28
7.4.2.11 getStudiedAt()	28
7.4.2.12 getWorkplaces()	28
7.4.2.13 hashCode()	29
7.4.2.14 setBirthdate()	29
7.4.2.15 setBirthplace()	29
7.4.2.16 setFilms()	30
7.4.2.17 setGender()	30
7.4.2.18 setGroupCode()	30
7.4.2.19 setHome()	30
7.4.2.20 setIdPerson()	31
7.4.2.21 setLastName()	31
7.4.2.22 setName()	31
7.4.2.23 setStudiedAt()	32
7.4.2.24 setWorkplaces()	32
7.4.2.25 toString()	32
7.5 relationships.Relationship Class Reference	33
7.5.1 Detailed Description	33
7.5.2 Constructor & Destructor Documentation	33
7.5.2.1 Relationship()	33
7.5.3 Member Function Documentation	34
7.5.3.1 equals()	34
7.5.3.2 getFriend1()	34
7.5.3.3 getFriend2()	35
7.5.3.4 hashCode()	36
7.5.3.5 setFriend1()	36
7.5.3.6 setFriend2()	36
7.5.3.7 toString()	37
7.6 relationships.Relationships Class Reference	38
7.6.1 Constructor & Destructor Documentation	38
7.6.1.1 Relationships()	39
7.6.2 Member Function Documentation	39

7.6.2.1 addRelationship()	39
7.6.2.2 deleteRelationship()	40
7.6.2.3 findRelationshipsById()	40
7.6.2.4 findRelationshipsByIds()	41
7.6.2.5 getRelations()	42
7.6.2.6 toString()	42
7.6.2.7 writeToFile()	42
7.6.3 Member Data Documentation	43
7.6.3.1 FIRST_LINE_RELATIONSHIPS	43
7.7 UnitTests Class Reference	44
7.7.1 Member Function Documentation	44
7.7.1.1 done()	44
<b>8 File Documentation</b>	<b>45</b>
8.1 D:/Projekty/DSA_ehu_JavaProject/src/fileHandling/FileHandling.java File Reference	45
8.2 D:/Projekty/DSA_ehu_JavaProject/src/Menu.java File Reference	45
8.3 D:/Projekty/DSA_ehu_JavaProject/src/people/People.java File Reference	45
8.4 D:/Projekty/DSA_ehu_JavaProject/src/people/Person.java File Reference	46
8.5 D:/Projekty/DSA_ehu_JavaProject/src/relationships/Relationship.java File Reference	46
8.6 D:/Projekty/DSA_ehu_JavaProject/src/relationships/Relationships.java File Reference	46
8.7 D:/Projekty/DSA_ehu_JavaProject/src/tests/UnitTests.java File Reference	46
<b>Index</b>	<b>47</b>



# Chapter 1

## User Documentation

The aim of this programming project is to manage a social network. This social network is formed by people that may be linked among each other if there is a friendship relationship between them.

If one runs program, menu options are displayed, where one can choose loading data from file with people, file with relationships between people, displaying loaded data, saving current data to new files, deleting people (including all of their relationships) and deleting particular relationship.

One has to type file name without '.txt' extension if decides to load data from file with people or from file with relationships.

If one decides to display data from people or relationships files, data will be listed on screen.

One has to type file name without '.txt' extension when decides to save current data. If file has already existed, it will be overwritten.

One has to type person's ID if decides to delete person from data. All relationships that include this ID will be removed.

Deleting relationship between two people from data requires typing their IDs.

This program is protected from invalid input:

One cannot input string in menu options. Exception with message "Bad input" will appear on screen, but menu will reload.

Unexisting files cannot be loaded. Error "Such file does not exist" will be displayed, but program will not crash.

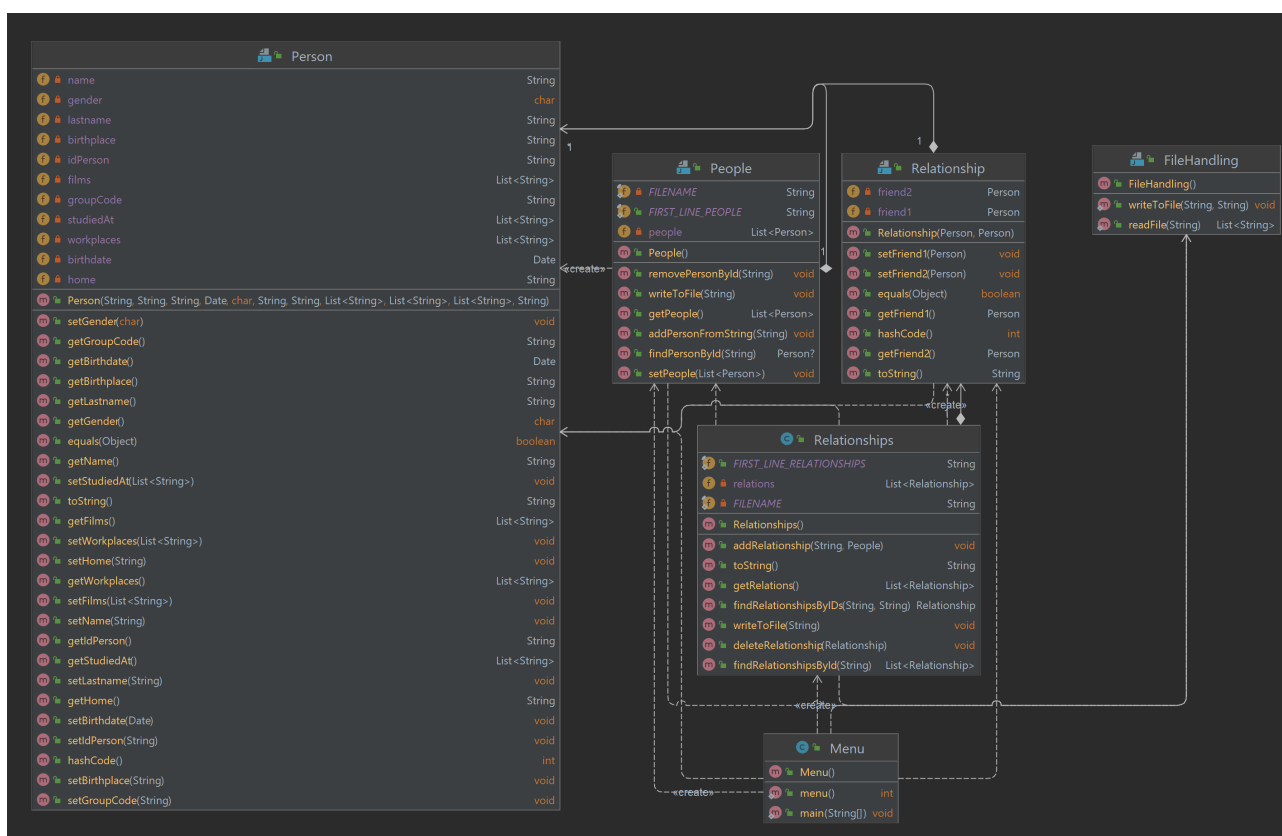
Unexisting person cannot be deleted. Exception "Person with that id does not exist" is thrown, but it doesn't interrupt the program.

Program doesn't crash if one tries to delete unexisting relationship.





## UML Diagram



**Figure 2.1 UML diagram of the classes**



## Chapter 3

# Namespace Index

### 3.1 Package List

Here are the packages with brief descriptions (if available):

<a href="#">fileHandling</a>	.....	<a href="#">11</a>
<a href="#">people</a>	.....	<a href="#">11</a>
<a href="#">relationships</a>	.....	<a href="#">11</a>



## Chapter 4

# Class Index

### 4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">fileHandling.FileHandling</a>	13
<a href="#">Menu</a>	15
<a href="#">people.People</a>	17
<a href="#">people.Person</a>	23
<a href="#">relationships.Relationship</a>	33
<a href="#">relationships.Relationships</a>	38
<a href="#">UnitTests</a>	44



## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

D:/Projekty/Dsa_ehu_JavaProject/src/Menu.java . . . . .	45
D:/Projekty/Dsa_ehu_JavaProject/src/fileHandling/FileHandling.java . . . . .	45
D:/Projekty/Dsa_ehu_JavaProject/src/people/People.java . . . . .	45
D:/Projekty/Dsa_ehu_JavaProject/src/people/Person.java . . . . .	46
D:/Projekty/Dsa_ehu_JavaProject/src/relationships/Relationship.java . . . . .	46
D:/Projekty/Dsa_ehu_JavaProject/src/relationships/Relationships.java . . . . .	46
D:/Projekty/Dsa_ehu_JavaProject/src/tests/UnitTests.java . . . . .	46





## Chapter 6

# Namespace Documentation

### 6.1 Package fileHandling

#### Classes

- class [FileHandling](#)

### 6.2 Package people

#### Classes

- class [People](#)
- class [Person](#)

### 6.3 Package relationships

#### Classes

- class [Relationship](#)
- class [Relationships](#)



## Chapter 7

# Class Documentation

### 7.1 fileHandling.FileHandling Class Reference

Collaboration diagram for fileHandling.FileHandling:

```
{fileHandling.FileHandling
||+ static List< String
> readFile(String fileName)
+ static void writeToFile
(String output, String
fileName)
}
```

#### Static Public Member Functions

- static List< String > [readFile](#) (String fileName) throws FileNotFoundException
- static void [writeToFile](#) (String output, String fileName) throws IOException

#### 7.1.1 Detailed Description

This class responsible for handling operations related to files It has classes which writes and rides text from/to files

#### 7.1.2 Member Function Documentation

##### 7.1.2.1 readFile()

```
static List< String > fileHandling.FileHandling.readFile (
    String fileName ) throws FileNotFoundException [static]
```

This method read lines from the whole file.

**Parameters**

<i>fileName</i>	is String parameter with a file name
-----------------	--------------------------------------

**Returns**

list of Strings

Here is the caller graph for this function:

**7.1.2.2 writeToFile()**

```
static void fileHandling.FileHandling.writeToFile (  
    String output,  
    String fileName ) throws IOException [static]
```

This method write String to the file.

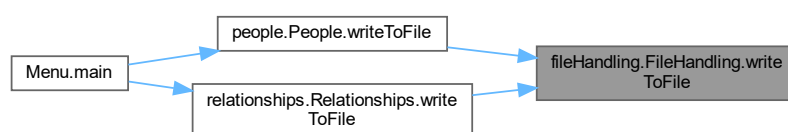
**Parameters**

<i>output</i>	String which will be written to file
<i>fileName</i>	String contains name of file where output will be written

**Exceptions**

<i>IOException</i>	This throws could occur when an I/O exception of some sort has occurred.
--------------------	--

Here is the caller graph for this function:

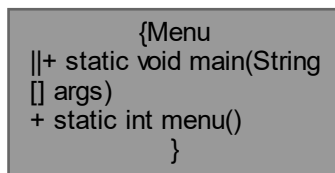


The documentation for this class was generated from the following file:

- D:/Projekty/DSA\_ehu\_JavaProject/src/fileHandling/[FileHandling.java](#)

## 7.2 Menu Class Reference

Collaboration diagram for Menu:



### Static Public Member Functions

- static void [main](#) (String[] args)
- static int [menu](#) ()

### 7.2.1 Detailed Description

This class is currently the main class of the projects. It creates the menu that allows user to insert People and Relationships and view them.

### 7.2.2 Member Function Documentation

#### 7.2.2.1 [main\(\)](#)

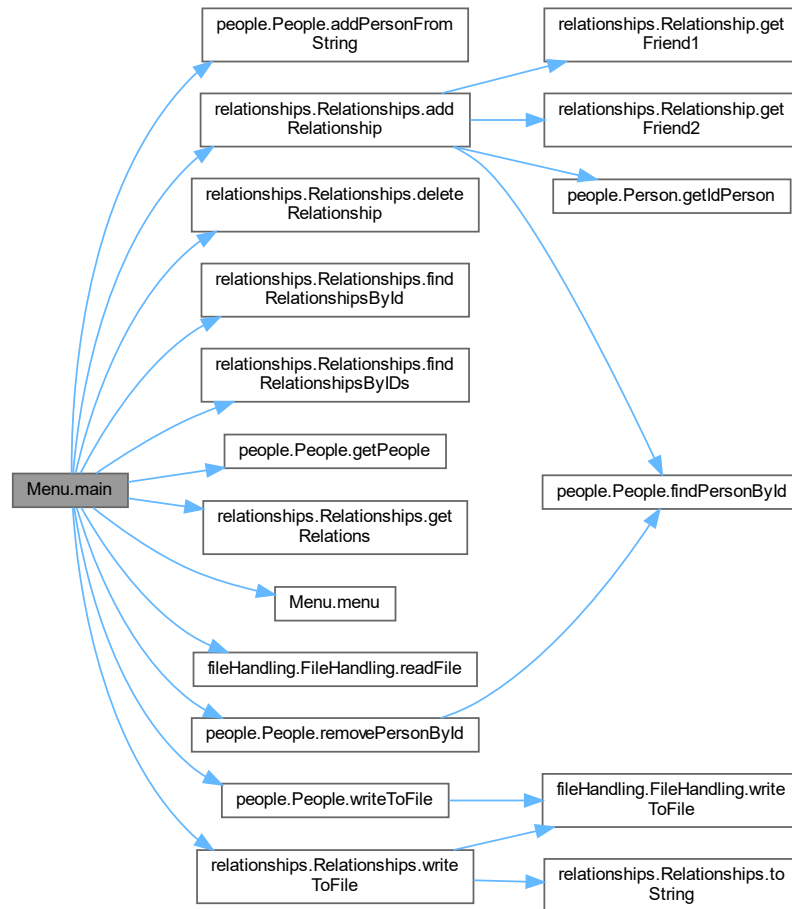
```
static void Menu.main (  
    String[] args ) [static]
```

This method consist of calls to all functions related to the options presented in menu.

#### Parameters

<i>args</i>	
-------------	--

Here is the call graph for this function:



### 7.2.2.2 menu()

```
static int Menu.menu ( ) [static]
```

This method prints menu options of our programming project and reads input from the user.

## Returns

integer

Here is the caller graph for this function:

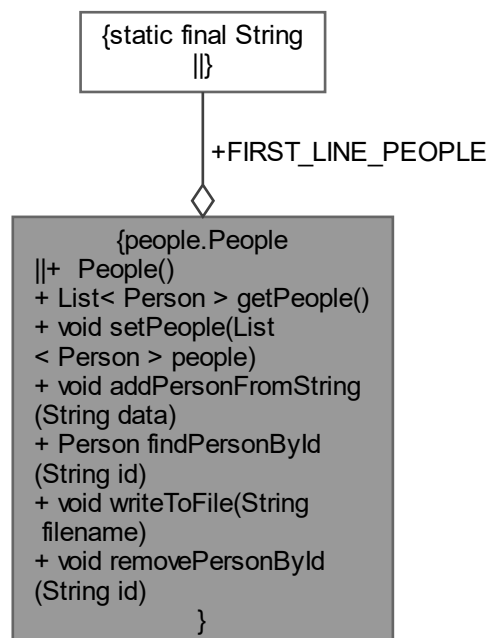


The documentation for this class was generated from the following file:

- [D:/Projekty/DSA\\_ehu\\_JavaProject/src/Menu.java](#)

## 7.3 people.People Class Reference

Collaboration diagram for people.People:



## Public Member Functions

- [People](#) ()
- List< [Person](#) > [getPeople](#) ()
- void [setPeople](#) (List< [Person](#) > people)
- void [addPersonFromString](#) (String data) throws ParseException
- [Person](#) [findPersonById](#) (String id)
- void [writeToFile](#) (String filename) throws IOException
- void [removePersonById](#) (String id)

## Static Public Attributes

- static final String [FIRST\\_LINE\\_PEOPLE](#) = "idperson,name,lastname,birthdate,gender,birthplace,home,studiedat,workplaces,fil

## 7.3.1 Constructor & Destructor Documentation

### 7.3.1.1 People()

```
people.People.People ( )
```

Constructor for a [People](#) class.

## 7.3.2 Member Function Documentation

### 7.3.2.1 addPersonFromString()

```
void people.People.addPersonFromString (
    String data ) throws ParseException
```

This method check if the user ID is already in list of people, if not - add person from string to list of people

#### Parameters

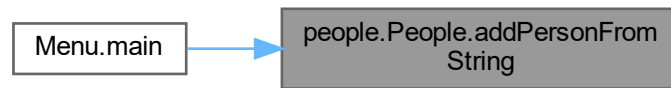
<i>data</i>	is string containing data for creating a new object of <a href="#">Person</a> class.
-------------	--

#### Exceptions

<i>ParseException</i>	Signals that an error has been reached unexpectedly while parsing.
-----------------------	--



Here is the caller graph for this function:



### 7.3.2.2 findPersonById()

```
Person people.People.findPersonById (
    String id )
```

This method finds `Person` object with corresponding ID in people list.

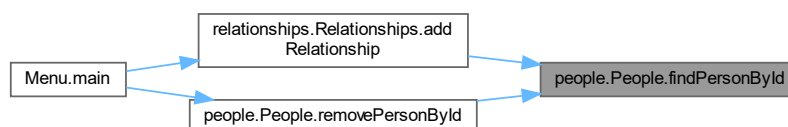
#### Parameters

<i>id</i>	string
-----------	--------

#### Returns

`Person` object

Here is the caller graph for this function:



### 7.3.2.3 getPeople()

```
List< Person > people.People.getPeople ( )
```

Getter for a people field.

**Returns**

list of [Person](#) type objects.

Here is the caller graph for this function:

**7.3.2.4 removePersonById()**

```
void people.People.removePersonById (  
    String id )
```

This method removes person with a certain id from the list of people. If person with that id does not exist, prints out a message.

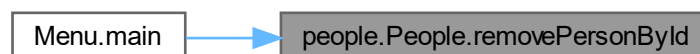
**Parameters**

<i>id</i>	String
-----------	--------

Here is the call graph for this function:



Here is the caller graph for this function:



### 7.3.2.5 setPeople()

```
void people.People.setPeople (
    List< Person > people )
```

Setter for a people field.

#### Parameters

<i>people</i>	is list of <a href="#">Person</a> type objects.
---------------	---

### 7.3.2.6 writeToFile()

```
void people.People.writeToFile (
    String filename ) throws IOException
```

This method writes string to given file name.

#### Parameters

<i>filename</i>	string
-----------------	--------

#### Exceptions

<i>IOException</i>	This throws could occur when writing to a file
--------------------	--

Here is the call graph for this function:



Here is the caller graph for this function:



### 7.3.3 Member Data Documentation

#### 7.3.3.1 FIRST\_LINE\_PEOPLE

```
final String people.People.FIRST_LINE_PEOPLE = "idperson,name,lastname,birthdate,gender,birthplace,home,studie  
[static]
```

The documentation for this class was generated from the following file:

- D:/Projekty/DSA\_ehu\_JavaProject/src/people/[People.java](#)

## 7.4 people.Person Class Reference

Collaboration diagram for people.Person:

```
{people.Person
||+ boolean equals(Object o)
+ int hashCode()
+ Person(String idPerson,
String name, String lastname,
Date birthdate, char gender,
String birthplace, String home,
List< String > studiedAt, List
< String > workplaces, List< String
> films, String groupCode)
+ String getIdPerson()
+ void setIdPerson(String
idPerson)
+ String getName()
+ void setName(String
name)
+ String getLastName()
+ void setLastName(String
lastname)
+ Date getBirthdate()
+ void setBirthdate(Date
birthdate)
+ char getGender()
+ void setGender(char
gender)
+ String getBirthplace()
+ void setBirthplace
(String birthplace)
+ String getHome()
+ void setHome(String
home)
+ List< String > getStudiedAt()
+ void setStudiedAt(List
< String > studiedAt)
+ List< String > getWorkplaces()
+ void setWorkplaces
(List< String > workplaces)
+ List< String > getFilms()
+ void setFilms(List
< String > films)
+ String getGroupCode()
+ void setGroupCode(String
groupCode)
+ String toString()
}
```

### Public Member Functions

- boolean `equals` (Object o)
- int `hashCode` ()

- **Person** (String idPerson, String name, String lastname, Date birthdate, char gender, String birthplace, String home, List< String > studiedAt, List< String > workplaces, List< String > films, String groupCode)
- String **getIdPerson** ()
- void **setIdPerson** (String idPerson)
- String **getName** ()
- void **setName** (String name)
- String **getLastName** ()
- void **setLastName** (String lastname)
- Date **getBirthdate** ()
- void **setBirthdate** (Date birthdate)
- char **getGender** ()
- void **setGender** (char gender)
- String **getBirthplace** ()
- void **setBirthplace** (String birthplace)
- String **getHome** ()
- void **setHome** (String home)
- List< String > **getStudiedAt** ()
- void **setStudiedAt** (List< String > studiedAt)
- List< String > **getWorkplaces** ()
- void **setWorkplaces** (List< String > workplaces)
- List< String > **getFilms** ()
- void **setFilms** (List< String > films)
- String **getGroupCode** ()
- void **setGroupCode** (String groupCode)
- String **toString** ()

## 7.4.1 Constructor & Destructor Documentation

### 7.4.1.1 Person()

```
people.Person.Person (
    String idPerson,
    String name,
    String lastname,
    Date birthdate,
    char gender,
    String birthplace,
    String home,
    List< String > studiedAt,
    List< String > workplaces,
    List< String > films,
    String groupCode )
```

All-argument constructor for a **Person** class.

#### Parameters

<i>idPerson</i>	String contains user ID
<i>name</i>	String contains username
<i>lastname</i>	String contains lastname
<i>birthdate</i>	Date contains birthdate

## Parameters

<i>gender</i>	char contains 'M' as 'male' / 'F' as 'female'
<i>birthplace</i>	String contains birthplace
<i>home</i>	String contains name of hometown
<i>studiedAt</i>	list of Strings contains name of university
<i>workplaces</i>	list of Strings contains workplaces
<i>films</i>	list of Strings contains favourites movie titles
<i>groupCode</i>	String contains group code

## 7.4.2 Member Function Documentation

### 7.4.2.1 equals()

```
boolean people.Person.equals (  
    Object o )
```

This method compares this object with object given in parameter. It overrides default method and allows us to use '==' operator.

## Parameters

<i>o</i>	is an object of undefined type
----------	--------------------------------

## Returns

boolean

Here is the call graph for this function:



Here is the caller graph for this function:



#### 7.4.2.2 getBirthdate()

```
Date people.Person.getBirthdate ( )
```

Getter for a birthdate field.

##### Returns

Date type object.

#### 7.4.2.3 getBirthplace()

```
String people.Person.getBirthplace ( )
```

Getter for a birthplace field.

##### Returns

String type object.

#### 7.4.2.4 getFilms()

```
List< String > people.Person.getFilms ( )
```

Getter for a films field.

##### Returns

List of String type objects.



#### 7.4.2.5 getGender()

```
char people.Person.getGender ( )
```

Getter for a gender field.

##### Returns

char type object.

#### 7.4.2.6 getGroupCode()

```
String people.Person.getGroupCode ( )
```

Getter for a groupCode field.

##### Returns

String type object.

#### 7.4.2.7 getHome()

```
String people.Person.getHome ( )
```

Getter for a getHome field.

##### Returns

String type object.

#### 7.4.2.8 getIdPerson()

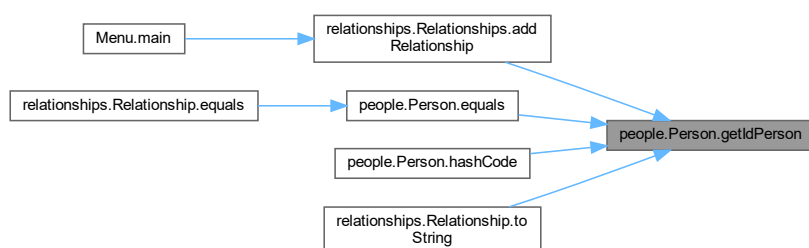
```
String people.Person.getIdPerson ( )
```

Getter for a idPerson field.

##### Returns

String type object.

Here is the caller graph for this function:



#### 7.4.2.9 getLastname()

```
String people.Person.getLastname ( )
```

Getter for a lastname field.

##### Returns

String type object.

#### 7.4.2.10 getName()

```
String people.Person.getName ( )
```

Getter for a name field.

##### Returns

String type object.

#### 7.4.2.11 getStudiedAt()

```
List< String > people.Person.getStudiedAt ( )
```

Getter for a studiedAt field.

##### Returns

List of String type objects.

#### 7.4.2.12 getWorkplaces()

```
List< String > people.Person.getWorkplaces ( )
```

Getter for a workplaces field.

##### Returns

List of String type objects.

#### 7.4.2.13 hashCode()

```
int people.Person.hashCode ( )
```

This method overrides default method and returns the integer hash code value of the object.

##### Returns

integer

Here is the call graph for this function:



#### 7.4.2.14 setBirthdate()

```
void people.Person.setBirthdate (
    Date birthdate )
```

Setter for a birthdate field.

##### Parameters

<i>birthdate</i>	is a Date type object.
------------------	------------------------

#### 7.4.2.15 setBirthplace()

```
void people.Person.setBirthplace (
    String birthplace )
```

Setter for a birthplace field.

##### Parameters

<i>birthplace</i>	is a String type object.
-------------------	--------------------------

#### 7.4.2.16 setFilms()

```
void people.Person.setFilms (
    List< String > films )
```

Setter for a films field.

##### Parameters

<i>films</i>	is a list of String type object.
--------------	----------------------------------

#### 7.4.2.17 setGender()

```
void people.Person.setGender (
    char gender )
```

Setter for a gender field.

##### Parameters

<i>gender</i>	is a char type object.
---------------	------------------------

#### 7.4.2.18 setGroupCode()

```
void people.Person.setGroupCode (
    String groupCode )
```

Setter for a groupCode field.

##### Parameters

<i>groupCode</i>	is a String type object.
------------------	--------------------------

#### 7.4.2.19 setHome()

```
void people.Person.setHome (
    String home )
```

Setter for a home field.

## Parameters

<i>home</i>	is a String type object.
-------------	--------------------------

**7.4.2.20 setIdPerson()**

```
void people.Person.setIdPerson (  
    String idPerson )
```

Setter for a idPerson field.

## Parameters

<i>idPerson</i>	is a String type object.
-----------------	--------------------------

**7.4.2.21 setLastname()**

```
void people.Person.setLastname (  
    String lastname )
```

Setter for a lastname field.

## Parameters

<i>lastname</i>	is a String type object.
-----------------	--------------------------

**7.4.2.22 setName()**

```
void people.Person.setName (  
    String name )
```

Setter for a name field.

## Parameters

<i>name</i>	is a String type object.
-------------	--------------------------

#### 7.4.2.23 setStudiedAt()

```
void people.Person.setStudiedAt (
    List< String > studiedAt )
```

Setter for a studiedAt field.

##### Parameters

<i>studiedAt</i>	is a list of String type object.
------------------	----------------------------------

#### 7.4.2.24 setWorkplaces()

```
void people.Person.setWorkplaces (
    List< String > workplaces )
```

Setter for a workplaces field.

##### Parameters

<i>workplaces</i>	is a list of String type object.
-------------------	----------------------------------

#### 7.4.2.25 toString()

```
String people.Person.toString ( )
```

This method overrides default method and returns a string representation of this object.

##### Returns

string

The documentation for this class was generated from the following file:

- D:/Projekty/DSA\_ehu\_JavaProject/src/people/[Person.java](#)

## 7.5 relationships.Relationship Class Reference

Collaboration diagram for relationships.Relationship:

```
{relationships.Relationship
||+ String toString()
+ boolean equals(Object o)
+ int hashCode()
+ Relationship(Person
  friend1, Person friend2)
+ Person getFriend1()
+ void setFriend1(Person
  friend1)
+ Person getFriend2()
+ void setFriend2(Person
  friend2)
}
```

### Public Member Functions

- String [toString](#) ()
- boolean [equals](#) (Object o)
- int [hashCode](#) ()
- [Relationship](#) (Person friend1, Person friend2)
- [Person](#) [getFriend1](#) ()
- void [setFriend1](#) (Person friend1)
- [Person](#) [getFriend2](#) ()
- void [setFriend2](#) (Person friend2)

### 7.5.1 Detailed Description

This class represents a single relationship between two people. Consists of two objects of type Person. @field friend1 an absolute URL giving the base location of the image @field friend2 the location of the image, relative to the url argument

### 7.5.2 Constructor & Destructor Documentation

#### 7.5.2.1 Relationship()

```
relationships.Relationship.Relationship (
    Person friend1,
    Person friend2 )
```

Constructor for a [Relationships](#) class.

## Parameters

<i>friend1</i>	is parameter of Person type.
<i>friend2</i>	is parameter of Person type.

### 7.5.3 Member Function Documentation

#### 7.5.3.1 equals()

```
boolean relationships.Relationship.equals (
    Object o )
```

This method compares this object with object given in parameter. It overrides default method and allows us to use '==' operator.

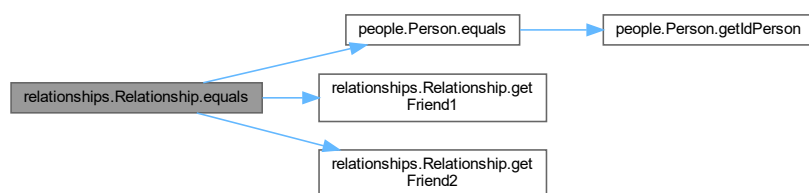
## Parameters

<i>o</i>	is an object of undefined type
----------	--------------------------------

## Returns

boolean

Here is the call graph for this function:



#### 7.5.3.2 getFriend1()

```
Person relationships.Relationship.getFriend1 ( )
```

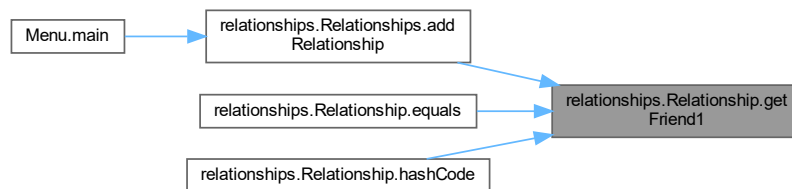
Getter for a friend1 field.



**Returns**

object of Person type.

Here is the caller graph for this function:

**7.5.3.3 getFriend2()**

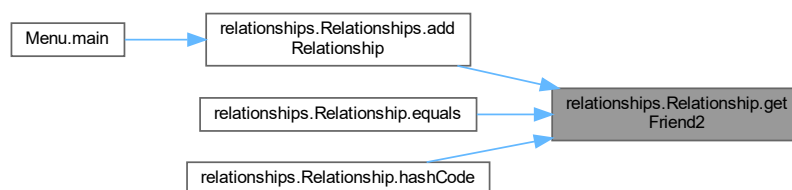
```
Person relationships.Relationship.getFriend2 ( )
```

Getter for a friend2 field.

**Returns**

object of Person type.

Here is the caller graph for this function:



#### 7.5.3.4 hashCode()

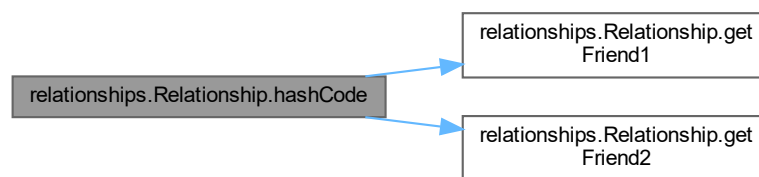
```
int relationships.Relationship.hashCode ( )
```

This method overrides default method and returns the integer hash code value of the object.

##### Returns

integer

Here is the call graph for this function:



#### 7.5.3.5 setFriend1()

```
void relationships.Relationship.setFriend1 (
    Person friend1 )
```

Setter for a friend1 field.

##### Parameters

<i>friend1</i>	is parameter of Person type.
----------------	------------------------------

#### 7.5.3.6 setFriend2()

```
void relationships.Relationship.setFriend2 (
    Person friend2 )
```

Setter for a friend2 field.

##### Parameters

<i>friend2</i>	is parameter of Person type.
----------------	------------------------------

### 7.5.3.7 toString()

```
String relationships.Relationship.toString ( )
```

This method returns a string representation of this object.

#### Returns

string

Here is the call graph for this function:

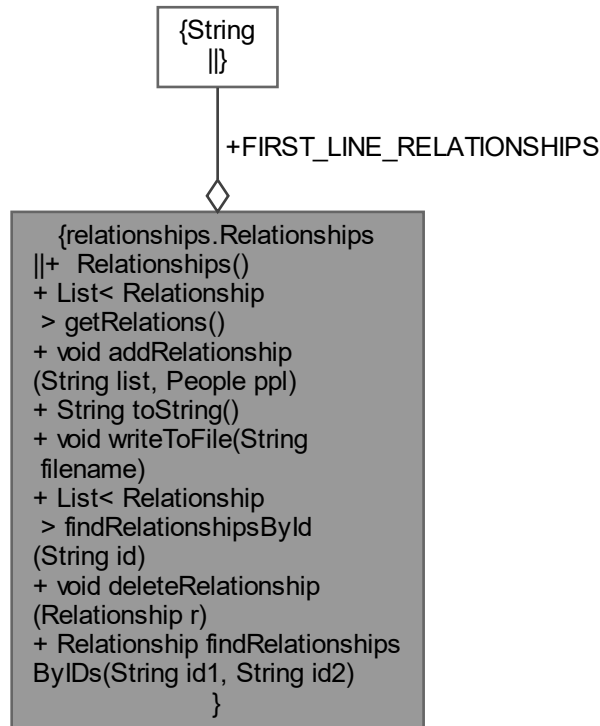


The documentation for this class was generated from the following file:

- [D:/Projekty/DSA\\_ehu\\_JavaProject/src/relationships/Relationship.java](#)

## 7.6 relationships.Relationships Class Reference

Collaboration diagram for relationships.Relationships:



### Public Member Functions

- [Relationships \(\)](#)
- [List< Relationship > getRelations \(\)](#)
- [void addRelationship \(String list, People ppl\)](#)
- [String toString \(\)](#)
- [void writeToFile \(String filename\) throws IOException](#)
- [List< Relationship > findRelationshipsById \(String id\)](#)
- [void deleteRelationship \(Relationship r\)](#)
- [Relationship findRelationshipsByIDs \(String id1, String id2\)](#)

### Static Public Attributes

- [static final String FIRST\\_LINE\\_RELATIONSHIPS = "friend1,friend2\n"](#)

#### 7.6.1 Constructor & Destructor Documentation

### 7.6.1.1 Relationships()

```
relationships.Relationships.Relationships ( )
```

Constructor for a [Relationships](#) class.

## 7.6.2 Member Function Documentation

### 7.6.2.1 addRelationship()

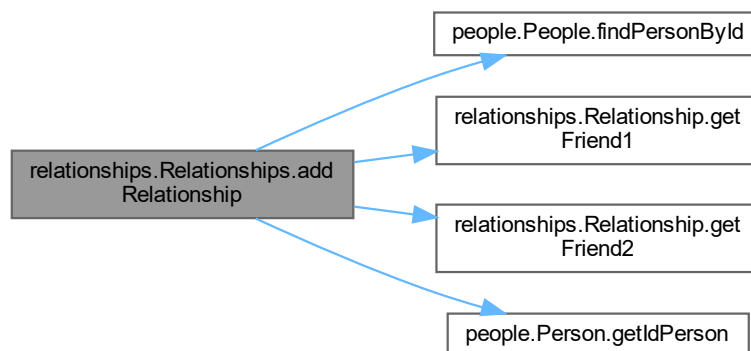
```
void relationships.Relationships.addRelationship (
    String list,
    People ppl )
```

This method check if relationship for two people exists. If not - create relationship between two people and add this relationship to list of relationships.

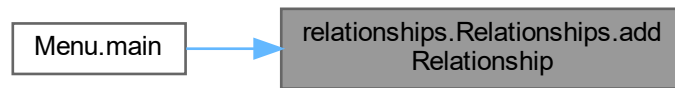
#### Parameters

<i>list</i>	This parameter is string list which consists two user IDs.
<i>ppl</i>	This parameter is People type object which have list of all users

Here is the call graph for this function:



Here is the caller graph for this function:



#### 7.6.2.2 deleteRelationship()

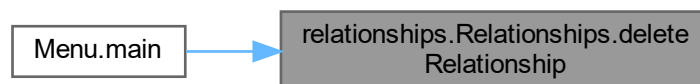
```
void relationships.Relationships.deleteRelationship (
    Relationship r )
```

Removes passed relationship from the list of relationships

##### Parameters

<i>r</i>	Relation
----------	----------

Here is the caller graph for this function:



#### 7.6.2.3 findRelationshipsById()

```
List< Relationship > relationships.Relationships.findRelationshipsById (
    String id )
```

This method finds every relationship containing passed id

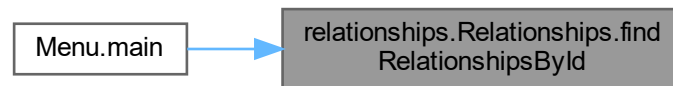
##### Parameters

<i>id</i>	String
-----------	--------

**Returns**

List<Relationship> List of relationships containing that id

Here is the caller graph for this function:

**7.6.2.4 findRelationshipsByIds()**

```
Relationship relationships.Relationships.findRelationshipsByIds (
    String id1,
    String id2 )
```

This method finds relationship containing passed IDs

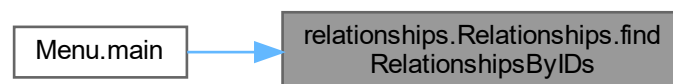
**Parameters**

<i>id1</i>	String
<i>id2</i>	String

**Returns**

Relationship containing passed IDs

Here is the caller graph for this function:



### 7.6.2.5 getRelations()

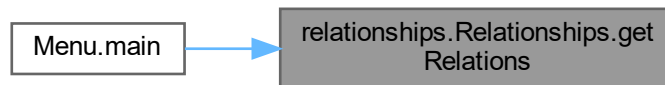
```
List< Relationship > relationships.Relationships.getRelations ( )
```

Getter for a relations field.

#### Returns

list of [Relationship](#) objects

Here is the caller graph for this function:



### 7.6.2.6 toString()

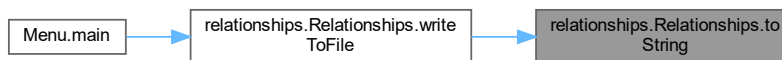
```
String relationships.Relationships.toString ( )
```

This method overrides default method and returns a string representation of this object.

#### Returns

string

Here is the caller graph for this function:



### 7.6.2.7 writeToFile()

```
void relationships.Relationships.writeToFile (
    String filename ) throws IOException
```

This method writes string to given file name.



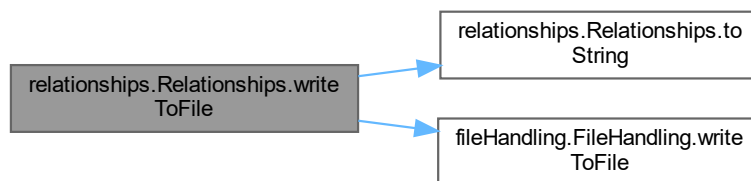
## Parameters

<i>filename</i>	string
-----------------	--------

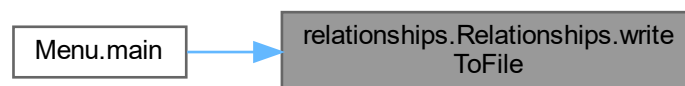
## Exceptions

<i>IOException</i>	This throws could occur when writing to a file
--------------------	--

Here is the call graph for this function:



Here is the caller graph for this function:



## 7.6.3 Member Data Documentation

### 7.6.3.1 FIRST\_LINE\_RELATIONSHIPS

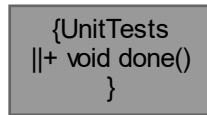
```
final String relationships.Relationships.FIRST_LINE_RELATIONSHIPS = "friend1,friend2\n" [static]
```

The documentation for this class was generated from the following file:

- `D:/Projekty/DSA_ehu_JavaProject/src/relationships/Relationships.java`

## 7.7 UnitTests Class Reference

Collaboration diagram for UnitTests:



### Classes

- class **FileHandlingTest**
- class **MenuTests**
- class **PeopleTest**
- class **PersonTest**
- class **RelationshipsTest**

### Public Member Functions

- void [done](#) ()

#### 7.7.1 Member Function Documentation

##### 7.7.1.1 done()

```
void UnitTests.done ( )
```

The documentation for this class was generated from the following file:

- D:/Projekty/DSA\_ehu\_JavaProject/src/tests/[UnitTests.java](#)

## Chapter 8

# File Documentation

### 8.1 D:/Projekty/DSA\_ehu\_JavaProject/src/fileHandling/FileHandling.java File Reference

#### Classes

- class [fileHandling.FileHandling](#)

#### Packages

- package [fileHandling](#)

### 8.2 D:/Projekty/DSA\_ehu\_JavaProject/src/Menu.java File Reference

#### Classes

- class [Menu](#)

### 8.3 D:/Projekty/DSA\_ehu\_JavaProject/src/people/People.java File Reference

#### Classes

- class [people.People](#)

#### Packages

- package [people](#)

## 8.4 D:/Projekty/DSA\_ehu\_JavaProject/src/people/Person.java File Reference

### Classes

- class [people.Person](#)

### Packages

- package [people](#)

## 8.5 D:/Projekty/DSA\_ehu\_JavaProject/src/relationships/↵ Relationship.java File Reference

### Classes

- class [relationships.Relationship](#)

### Packages

- package [relationships](#)

## 8.6 D:/Projekty/DSA\_ehu\_JavaProject/src/relationships/↵ Relationships.java File Reference

### Classes

- class [relationships.Relationships](#)

### Packages

- package [relationships](#)

## 8.7 D:/Projekty/DSA\_ehu\_JavaProject/src/tests/UnitTests.java File Reference

### Classes

- class [UnitTests](#)
- class [UnitTests.MenuTests](#)
- class [UnitTests.FileHandlingTest](#)
- class [UnitTests.PeopleTest](#)
- class [UnitTests.PersonTest](#)
- class [UnitTests.RelationshipsTest](#)
- class [UnitTests.RelationshipsTest.RelationTest](#)

# Index

addPersonFromString  
    people.Person, 18  
addRelationship  
    relationships.Relationships, 39  
D:/Projekty/DSA\_ehu\_JavaProject/src/fileHandling/FileHandling.java, 45  
D:/Projekty/DSA\_ehu\_JavaProject/src/Menu.java, 45  
D:/Projekty/DSA\_ehu\_JavaProject/src/people/People.java, 45  
D:/Projekty/DSA\_ehu\_JavaProject/src/people/Person.java, 46  
D:/Projekty/DSA\_ehu\_JavaProject/src/relationships/Relationships.java, 46  
D:/Projekty/DSA\_ehu\_JavaProject/src/relationships/Relationships.java, 46  
D:/Projekty/DSA\_ehu\_JavaProject/src/tests/UnitTests.java, 46  
deleteRelationship  
    relationships.Relationships, 40  
done  
    UnitTests, 44  
  
equals  
    people.Person, 25  
    relationships.Relationship, 34  
  
fileHandling, 11  
fileHandling.FileHandling, 13  
    readFile, 13  
    writeToFile, 14  
findPersonById  
    people.Person, 19  
findRelationshipsById  
    relationships.Relationships, 40  
findRelationshipsByIds  
    relationships.Relationships, 41  
FIRST\_LINE\_PEOPLE  
    people.Person, 22  
FIRST\_LINE\_RELATIONSHIPS  
    relationships.Relationships, 43  
  
getBirthdate  
    people.Person, 26  
getBirthplace  
    people.Person, 26  
getFilms  
    people.Person, 26  
getFriend1  
    relationships.Relationship, 34  
getFriend2  
    relationships.Relationship, 35  
getGender  
    people.Person, 26  
getGroupCode  
    people.Person, 27  
getHome  
    people.Person, 27  
getIdPerson  
    people.Person, 27  
getLastname  
    people.Person, 27  
getName  
    people.Person, 28  
getPeople  
    people.People, 19  
getRelations  
    relationships.Relationships, 41  
getStudiedAt  
    people.Person, 28  
getWorkplaces  
    people.Person, 28  
  
hashCode  
    people.Person, 28  
    relationships.Relationship, 35  
  
main  
    Menu, 15  
Menu, 15  
    main, 15  
    menu, 16  
menu  
    Menu, 16  
  
People  
    people.People, 18  
people, 11  
people.People, 17  
    addPersonFromString, 18  
    findPersonById, 19  
    FIRST\_LINE\_PEOPLE, 22  
    getPeople, 19  
    People, 18  
    removePersonById, 20  
    setPeople, 20  
    writeToFile, 21  
people.Person, 23  
    equals, 25  
    getBirthdate, 26

- getBirthplace, 26
- getFilms, 26
- getGender, 26
- getGroupCode, 27
- getHome, 27
- getIdPerson, 27
- getLastname, 27
- getName, 28
- getStudiedAt, 28
- getWorkplaces, 28
- hashCode, 28
- Person, 24
- setBirthdate, 29
- setBirthplace, 29
- setFilms, 29
- setGender, 30
- setGroupCode, 30
- setHome, 30
- setIdPerson, 31
- setLastname, 31
- setName, 31
- setStudiedAt, 31
- setWorkplaces, 32
- toString, 32
- Person
  - people.Person, 24
- readFile
  - fileHandling.FileHandling, 13
- Relationship
  - relationships.Relationship, 33
- Relationships
  - relationships.Relationships, 38
- relationships, 11
- relationships.Relationship, 33
  - equals, 34
  - getFriend1, 34
  - getFriend2, 35
  - hashCode, 35
  - Relationship, 33
  - setFriend1, 36
  - setFriend2, 36
  - toString, 37
- relationships.Relationships, 38
  - addRelationship, 39
  - deleteRelationship, 40
  - findRelationshipsById, 40
  - findRelationshipsByIds, 41
  - FIRST\_LINE\_RELATIONSHIPS, 43
  - getRelations, 41
  - Relationships, 38
  - toString, 42
  - writeToFile, 42
- removePersonById
  - people.People, 20
- setBirthdate
  - people.Person, 29
- setBirthplace
  - people.Person, 29
- setFilms
  - people.Person, 29
- setFriend1
  - relationships.Relationship, 36
- setFriend2
  - relationships.Relationship, 36
- setGender
  - people.Person, 30
- setGroupCode
  - people.Person, 30
- setHome
  - people.Person, 30
- setIdPerson
  - people.Person, 31
- setLastname
  - people.Person, 31
- setName
  - people.Person, 31
- setPeople
  - people.People, 20
- setStudiedAt
  - people.Person, 31
- setWorkplaces
  - people.Person, 32
- toString
  - people.Person, 32
  - relationships.Relationship, 37
  - relationships.Relationships, 42
- UnitTests, 44
  - done, 44
- writeToFile
  - fileHandling.FileHandling, 14
  - people.People, 21
  - relationships.Relationships, 42