THEORETICAL EXERCISE 2

PERSONA CLASS PROBLEM

GROUP A03

1. Pseudocode of the identified methods

We must create a Persona class that initializes objects with valid states. The data for each Persona object includes name, last name, date of birth, nationality, education, English certification, phone number and email.

```
public Persona(String name, String lastName, Date dateOfBirth, String nationality, String education, String englishCertification, String phoneNumber, String email) {
        this.name = name;
        this.lastName = lastName;
        this.dateOfBirth = dateOfBirth;
        this.nationality = nationality;
        this.education = education;
        this.englishCertification = englishCertification;
        this.phoneNumber = phoneNumber;
        this.email = email;
    }
```

Also, we are interested in including methods that help determine if the person is of legal age and if they are European.

```
public int getAge() {
        Date currentDate = new Date();
        long ageInMillis = currentDate.getTime() - dateOfBirth.getTime();
        long millisInYear = 1000L * 60 * 60 * 24 * 365;
        return (int) (ageInMillis / millisInYear);
    }
public boolean isLegalAge() {
        return getAge() >= 18;
    }
```

```
public boolean isEuropean() {

    List<String> europeanCountries = Arrays.asList("Austria", "Belgium", "Bulgaria",
"Croatia", "Republic of Cyprus", "Czech Republic", "Denmark", "Estonia", "Finland",
"France", "Germany", "Greece", "Hungary", "Ireland", "Italy", "Latvia", "Lithuania",
"Luxembourg", "Malta", "Netherlands", "Poland", "Portugal", "Romania", "Slovakia",
"Slovenia", "Spain", "Sweden");

    return europeanCountries.contains(nationality);

}
```

2. Identifying variables

The variables are GetAge() and isEuropean().

3. Identifying test values

Parameters	**Equivalence classes**	**Equivalence class values**	**Lightweight variant**	**Heavy variant**	**Error-guessing**
GetAge()	$(-\infty, 18)$, $[18, +\infty)$	$15,\ 33$	18	$17,\ 19$	$-2^{31}-1$, $2^{31}+1$, 200
IsEuropean()	true /false	true / false			NULL

4. Maximum number of test cases

The maximum number of test cases is: 3 from the IsEuropean() and 8 from the GetAge(). So in total, we have: 3*8=24 test cases.

5. Set of test cases

Each use coverage goes as follows:

CP1: {15, false}

CP2: {33, true}

CP3: {18, NULL}

CP4: {17, false}

CP5: {19, false}

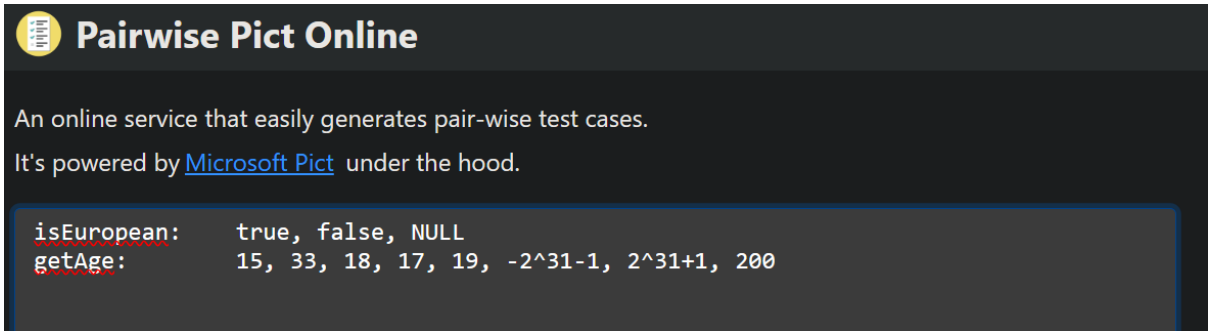
CP6: $\{-2^{31}-1$, true$\}$

CP7: $\{2^{31}+1$, NULL$\}$

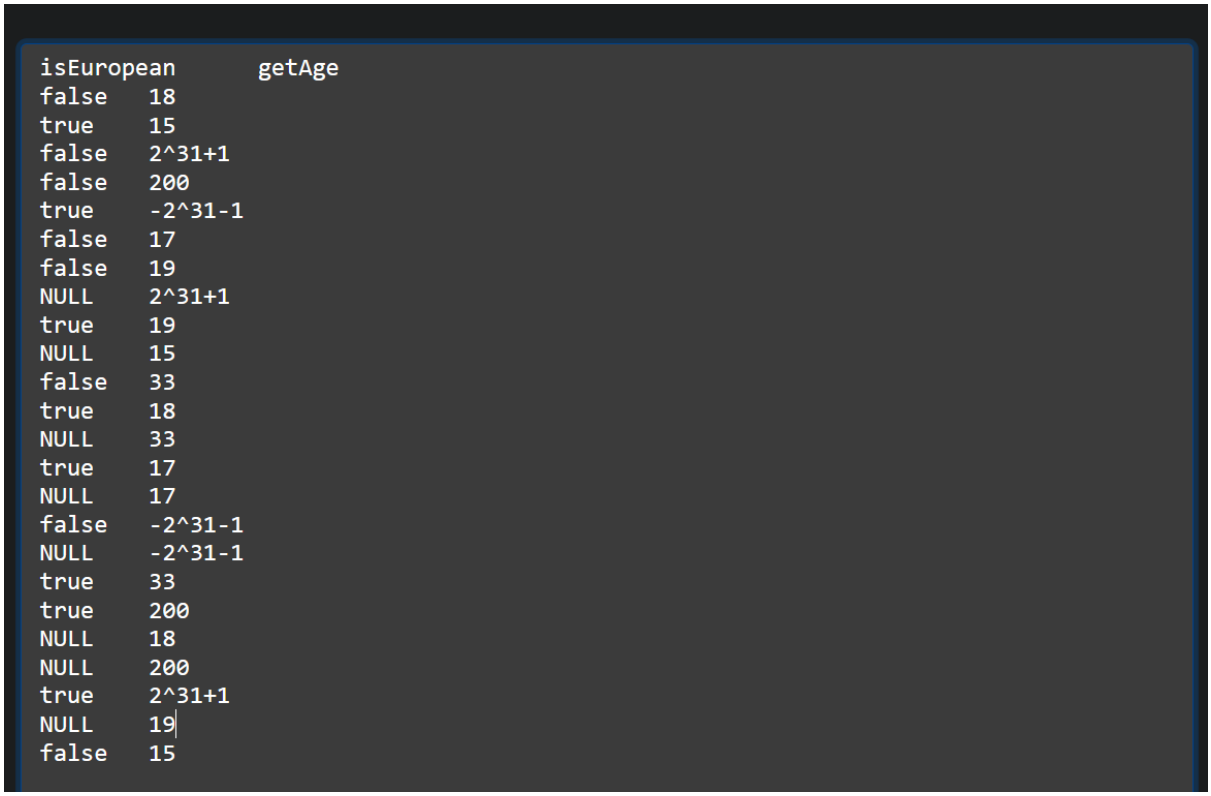
CP8: $\{200$, false$\}$

6. Pairwise Testing

We have used the following website to calculate it: [Pairwise Pict Online (yuuniworks.com)](). After that we have included the following:



The result of the pairwise testing can be observed in this image:



7. Decision Coverage

Parameter **GetAge**:

Since this parameter has been highly simplified, testing it by means of isLegalAge(), assures that GetAge is also properly tested. Hence, the resultant table is also very simple:

A: isLegalAge

Condition	Decision	Dominant
A	A	
true	true	A
false	false	A

Test cases:

isLegalAge()	Result
21	true
12	false

Parameter **isEuropean**:

As it happened with isLegalAge, we have a perfectly simplified table:

A: IsEuropean

Condition	Decision	Dominant
A	A	
true	true	A
false	false	A

Test cases:

IsEuropean()	Result
Austria	true
Canada	false

8. MC/DC coverage

As the code we have is simple, the decision coverage corresponds also to the MC/DC coverage.

9. Final comments

This is the result of the code after being tested with Junit. As we can see, the coverage is 98% for the persona class (notice also that auxiliary methods have been tested as well), this means that the program is well tested as it was expected:

Ejemplo Uso Plugins Informes Testing

Ejemplo Uso Plugins Informes Testing

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
es.uclm.esi.iso2.ga03.persona		98%		100%	1	7	1	34	1	6	0	2
es.uclm.esi.iso2.ga03.persona.utility		91%		n/a	1	5	1	12	1	5	0	1
Total	6 of 280	97%	0 of 2	100%	2	12	2	46	2	11	0	3