Variables that must be considered to test the program:

In this case we have 6 variables that are the age, studying, living with parents, is enrol in university, is working and if started working.

Identify the test values for each one of the variables previously identified, specifying the technique used to obtain each of those values:

We decided to create a test case for each one of the banks account types outcomes of our problem:

-Comfort

-Come on you can

-Save Now While You Can

-Jump out of the Nest

-Become independent it's about time

-Welcome to Adult Life

-Negative age

-Non-existing bank account

Furthermore, exist a special test case specially if the age is negative.

The equivalence class method is used for that we divide the possible inputs in disjoin sets:

Age: int

```
|----------------|-------------------------------|-------------------|-------------------------------|
(                ](                              )[                  )[                              )
- ∞              0                               18                  25                              ∞
```

We can consider the equivalence classes:

1) Equivalence class: 'negative' (-∞, 0)

2) Equivalence class: 'minor age' {0-17}

3) Equivalence class: 'legalbutminus25' {18-24}

4) Equivalence class: 'over25' {25, ∞)

Testing values: {-1,0,17,18,20,25,26}

For the rest of variables that are Boolean we only must check true or false.

Calculate the maximum possible number of test cases that could be generated from the test values:

So, the maximum possible number of test cases is 4*2*2*2*2*2 = 128 possible test cases.

Define some test suites using each use:

Age = {-1,0,17,18,20,25,26}

….booleans {true,false} //following constructor order

/* int age, boolean isStudying, boolean islivingwithparents, boolean isEnrolledinUNI, boolean isWorking, boolean startedworking*/

We will create the following test suite:

Test suite = {(-1,x,x,x,x,x),(0,x,x,x,x,x),(17,true,true,x,x,x),(18,false,flase,true,x,x),(20, false,true,true,x,x),(25,false,true,false,x,true,x),(26, false,flase,false,x,true,x)}

Define test suits to achieve pairwise coverage by using the proposed algorithm in Lectures. You can check the results by means of the software PICT1:

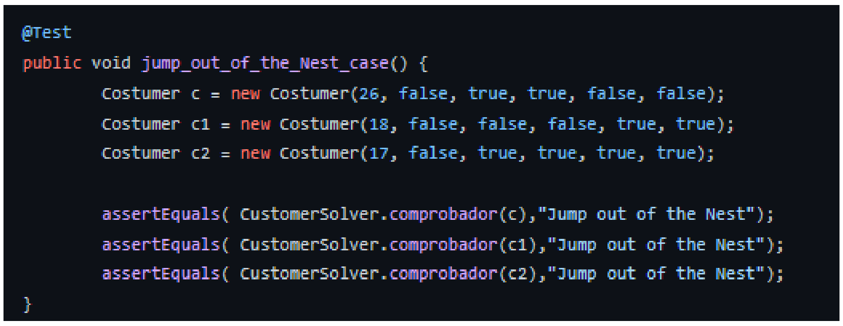
PICT CODE:

```
AGE:            -1,0,17,18,20,25,26

ISSTUDY:        0, 1

LWPARENTS:      0, 1

ENROLUNI :      0, 1

ISWORKING:      0, 1

STARTEDWORKING:     0, 1

METHOD:         Comfort, Comeonyoucan, SaveNowWhileYouCan, JumpoutoftheNest,
Becomeindependentitsabouttime, WelcometoAdultLife,negativeage

if [AGE] < 0 or [AGE] = 0 then [METHOD] = "negativeage";

if [AGE] >= 18 and [AGE] < 25 and [ENROLUNI] = 1 and [LWPARENTS] = 0 then [METHOD] =
"Comeonyoucan";

if [AGE] >= 18 and [AGE] < 25 and [STARTEDWORKING] = 1 and [LWPARENTS] = 0  and
[ISWORKING] = 0 and [ENROLUNI] = 0 then [METHOD] = "JumpoutoftheNest";

if [AGE] < 18 and [ISSTUDY] = 1 and [LWPARENTS] = 1 then [METHOD] = "Comfort";

if [AGE] < 25 and [ENROLUNI] = 1 and [LWPARENTS] = 0 then [METHOD] = "Comeonyoucan";

if [AGE] >= 18 and [AGE] <= 25 and [STARTEDWORKING] = 1 and [LWPARENTS] = 1 then
[METHOD] = "SaveNowWhileYouCan";

if [AGE] >= 18 and [AGE] < 25 and [STARTEDWORKING] = 1 and [LWPARENTS] = 0 then
[METHOD] = "JumpoutoftheNest";

if [AGE] > 25 and [ISWORKING] = 1 and [LWPARENTS] = 1 then [METHOD] =
"Becomeindependentitsabouttime";

if [AGE] > 25 and [ISWORKING] = 1 and [LWPARENTS] = 0 then [METHOD] =
"WelcometoAdultLife"; ALL CASES GENERATED:
```

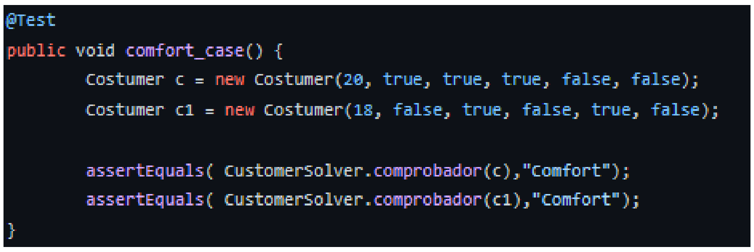
AGE	SDY	LWPS	RLUN	SWG	STRWG	METHOD
26	0	1	1	0	0	JumpoutoftheNest
20	1	0	0	1	0	Comeonyoucan
20	1	1	1	0	0	Comfort
18	0	0	0	1	1	JumpoutoftheNest
26	1	0	0	0	1	WelcometoAdultLife
17	0	1	1	1	1	JumpoutoftheNest
18	0	1	0	1	0	Comfort
17	1	0	0	0	0	Becomeindependentitsabouttime
20	0	1	1	1	0	WelcometoAdultLife
17	0	1	1	1	1	SaveNowWhileYouCan
-1	1	1	1	0	0	negativeage
-1	0	0	0	1	0	negativeage
26	0	0	1	0	1	Comfort
18	0	1	1	1	0	Becomeindependentitsabouttime
17	0	1	1	0	1	Comeonyoucan
-1	0	1	0	1	1	negativeage
17	0	1	0	0	1	WelcometoAdultLife
0	1	1	0	0	1	negativeage
26	0	1	1	1	1	Becomeindependentitsabouttime
0	0	1	0	1	0	negativeage
20	0	1	0	0	0	Becomeindependentitsabouttime
25	1	0	1	1	0	Becomeindependentitsabouttime
17	0	1	1	1	1	Comfort
26	1	0	0	0	0	SaveNowWhileYouCan
25	0	0	0	0	1	WelcometoAdultLife
25	1	1	0	0	0	Comfort
18	0	0	0	0	0	SaveNowWhileYouCan
25	1	1	0	1	0	Comeonyoucan
18	1	1	1	0	0	Comeonyoucan
25	1	1	0	1	0	JumpoutoftheNest

18	1	1	0	1	0	WelcometoAdultLife
20	1	1	1	0	1	SaveNowWhileYouCan
0	0	1	1	1	0	negativeage
25	0	0	1	0	0	SaveNowWhileYouCan
26	1	1	1	0	0	Comeonyoucan
-1	0	1	1	0	0	negativeage
20	1	1	1	0	0	JumpoutoftheNest
-1	1	0	0	1	1	negativeage
0	0	1	1	0	1	negativeage
0	1	0	0	0	0	negativeage
-1	1	0	1	0	0	negativeage

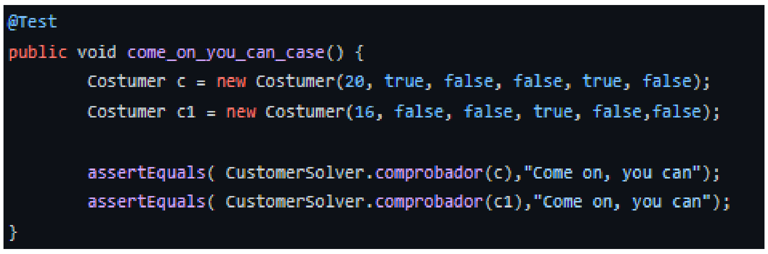
<u>For code snippets that include decisions, propose a set of test cases to achieve coverage of decisions:</u>



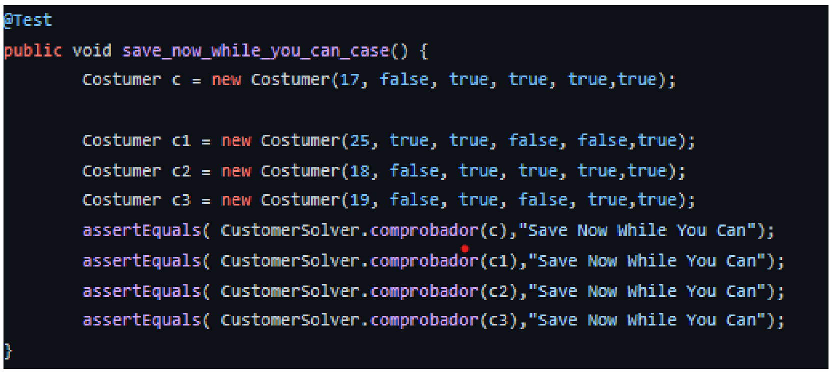
```java
@Test
public void jump_out_of_the_Nest_case() {
        Costumer c = new Costumer(26, false, true, true, false, false);
        Costumer c1 = new Costumer(18, false, false, false, true, true);
        Costumer c2 = new Costumer(17, false, true, true, true, true);

        assertEquals( CustomerSolver.comprobador(c),"Jump out of the Nest");
        assertEquals( CustomerSolver.comprobador(c1),"Jump out of the Nest");
        assertEquals( CustomerSolver.comprobador(c2),"Jump out of the Nest");
}
```



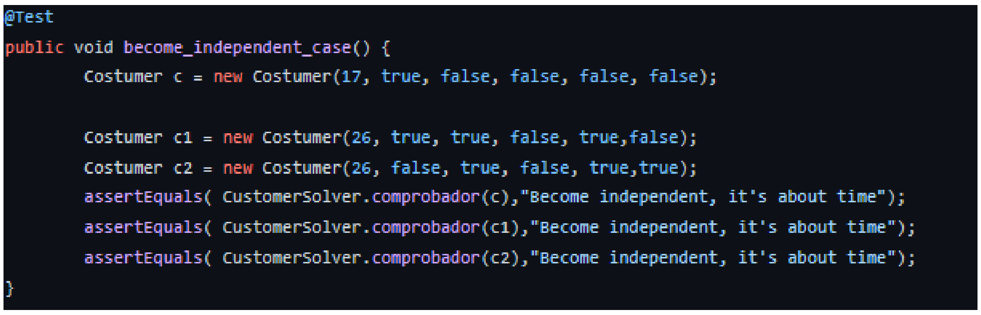
```java
@Test
public void comfort_case() {
        Costumer c = new Costumer(20, true, true, true, false, false);
        Costumer c1 = new Costumer(18, false, true, false, true, false);

        assertEquals( CustomerSolver.comprobador(c),"Comfort");
        assertEquals( CustomerSolver.comprobador(c1),"Comfort");
}
```



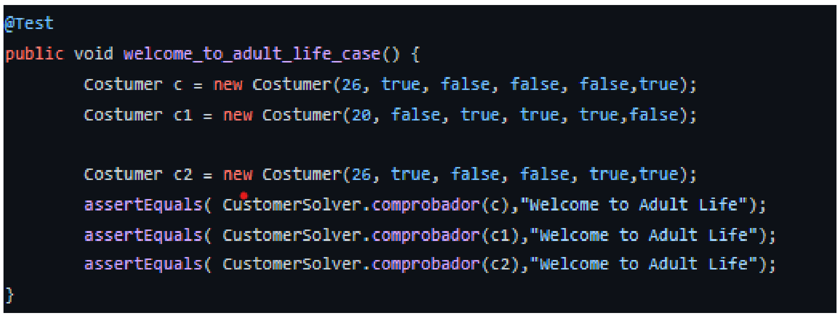
```java
@Test
public void come_on_you_can_case() {
        Costumer c = new Costumer(20, true, false, false, true, false);
        Costumer c1 = new Costumer(16, false, false, true, false,false);

        assertEquals( CustomerSolver.comprobador(c),"Come on, you can");
        assertEquals( CustomerSolver.comprobador(c1),"Come on, you can");
}
```



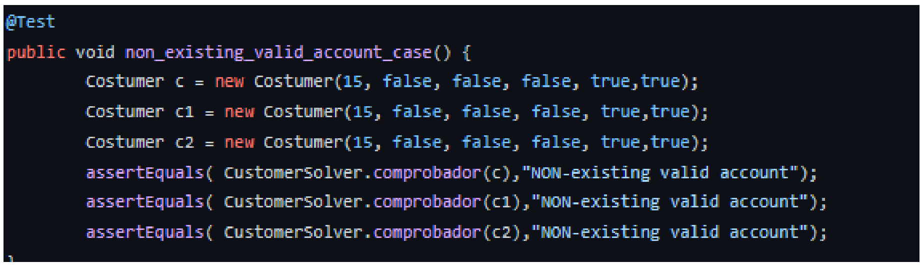
```java
@Test
public void save_now_while_you_can_case() {
        Costumer c = new Costumer(17, false, true, true, true,true);

        Costumer c1 = new Costumer(25, true, true, false, false,true);
        Costumer c2 = new Costumer(18, false, true, true, true,true);
        Costumer c3 = new Costumer(19, false, true, false, true,true);
        assertEquals( CustomerSolver.comprobador(c),"Save Now While You Can");
        assertEquals( CustomerSolver.comprobador(c1),"Save Now While You Can");
        assertEquals( CustomerSolver.comprobador(c2),"Save Now While You Can");
        assertEquals( CustomerSolver.comprobador(c3),"Save Now While You Can");
}
```



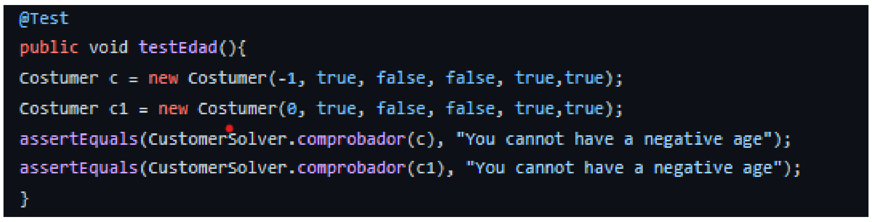
```java
@Test
public void become_independent_case() {
        Costumer c = new Costumer(17, true, false, false, false, false);

        Costumer c1 = new Costumer(26, true, true, false, true,false);
        Costumer c2 = new Costumer(26, false, true, false, true,true);
        assertEquals( CustomerSolver.comprobador(c),"Become independent, it's about time");
        assertEquals( CustomerSolver.comprobador(c1),"Become independent, it's about time");
        assertEquals( CustomerSolver.comprobador(c2),"Become independent, it's about time");
}
```



```java
@Test
public void welcome_to_adult_life_case() {
        Costumer c = new Costumer(26, true, false, false, false,true);
        Costumer c1 = new Costumer(20, false, true, true, true,false);

        Costumer c2 = new Costumer(26, true, false, false, true,true);
        assertEquals( CustomerSolver.comprobador(c),"Welcome to Adult Life");
        assertEquals( CustomerSolver.comprobador(c1),"Welcome to Adult Life");
        assertEquals( CustomerSolver.comprobador(c2),"Welcome to Adult Life");
}
```



```java
@Test
public void non_existing_valid_account_case() {
        Costumer c = new Costumer(15, false, false, false, true,true);
        Costumer c1 = new Costumer(15, false, false, false, true,true);
        Costumer c2 = new Costumer(15, false, false, false, true,true);
        assertEquals( CustomerSolver.comprobador(c),"NON-existing valid account");
        assertEquals( CustomerSolver.comprobador(c1),"NON-existing valid account");
        assertEquals( CustomerSolver.comprobador(c2),"NON-existing valid account");
```



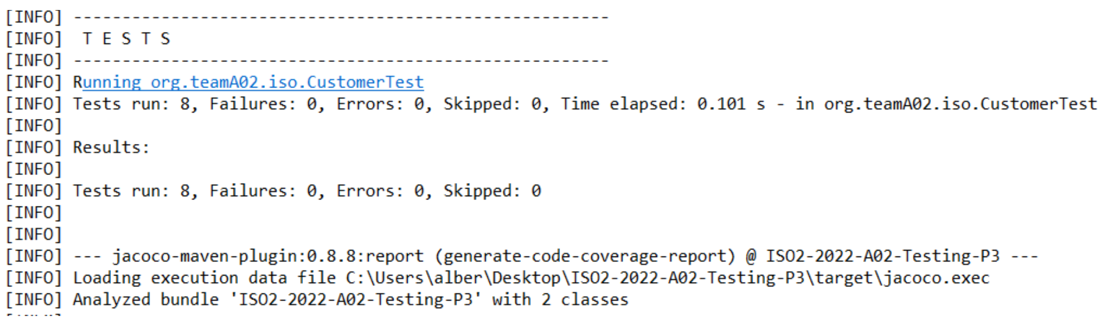
```java
@Test
public void testEdad(){
Costumer c = new Costumer(-1, true, false, false, true,true);
Costumer c1 = new Costumer(0, true, false, false, true,true);
assertEquals(CustomerSolver.comprobador(c), "You cannot have a negative age");
assertEquals(CustomerSolver.comprobador(c1), "You cannot have a negative age");
}
```

<u>Comment on the results of the number of test cases obtained in section 4, 5, and 6, as well as the execution of the oracles: what could be said about the coverage achieved?</u>

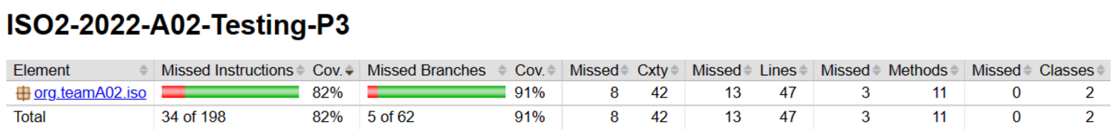
Number of test cases obtained in section 4 (128 test cases) is too big to be implemented for this theoretical exercise. We believe that we can perform decent testing, while using smaller test case set.

In the case of each use although the test cases were less than the section 4 we think its better to use pairwise due to it's a more complete test and smaller than section 4 test cases number.

About the coverage obtained by testing:



```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running org.teamA02.iso.CustomerTest
[INFO] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.101 s - in org.teamA02.iso.CustomerTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jacoco-maven-plugin:0.8.8:report (generate-code-coverage-report) @ ISO2-2022-A02-Testing-P3 ---
[INFO] Loading execution data file C:\Users\alber\Desktop\ISO2-2022-A02-Testing-P3\target\jacoco.exec
[INFO] Analyzed bundle 'ISO2-2022-A02-Testing-P3' with 2 classes
```



ISO2-2022-A02-Testing-P3

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
org.teamA02.iso	🟥🟩	82%	🟥🟩	91%	8	42	13	47	3	11	0	2
Total	34 of 198	82%	5 of 62	91%	8	42	13	47	3	11	0	2