Testing

Rubrica telefonica

Componenti Gruppo 17:

Di Marino Domenico
Matricola: 0612707421

Adinolfi Giovanni
Matricola: 0612708352

Di Crescenzo Francesco
Matricola: 0612708640

INDICE DEL DOCUMENTO

1.0 Introduzione agli Unit Test **2**
2.0 Unit Test Cases 2
2.1 UNIT TEST CASE: Rubrica 2
2.2 UNIT TEST CASE: Contatto 8
2.3 UNIT TEST CASE: Email 19
2.4 UNIT TEST CASE: Telefono 23
3.0 Usability Testing **27**
4.0 Test Report **27**

1.0 Introduzione agli Unit Test

Di seguito vengono riportati i test delle singole classi derivate dal class diagram (presente nel Design). Per testarli in maniera automatizzata fare riferimento ai test, presenti nel progetto contenente il codice implementato, eseguiti con JUnit5 su Neatbeans.

2.0 Unit Test Cases

2.1 UNIT TEST CASE: Rubrica

Prima di effettuare ogni test, vengono eseguite le seguenti operazioni usando il tag BeforeEach e alla fine di ogni test viene eseguito l'AfterEach:

```
@BeforeEach
    public void setUp() throws NominativoAssenteException {
        t = new Telefono();
        e = new Email();
        t.aggiungiNumero("3111111111");
        e.aggiungiEmail("mariorossi@example.com");
        c = new Contatto("Mario","Rossi",t,e);
        r = new Rubrica("rubrica");
        r.aggiungiContatto(c);
    }
```

I test riguardanti la Rubrica sono i seguenti:

UTC 1.1 Test Rubrica.constructor
Test items: classe Rubrica, metodo constructor
Test: testConstructor **Case description**: constructor classico **Location**: ContattoTest.testConstructor **Input**: Rubrica result = new Rubrica("result"); List<Contatto> lista = new ArrayList<>(); **Oracle**: lista = result.getContatti();

UTC 1.2 Test Rubrica.aggiungiContatto
Test items: classe Rubrica, metodo aggiungiContatto
Test: testAggiungiContatto **Case description**: aggiungere un contatto alla rubrica **Location**: RubricaTest.testAggiungiContatto **Input**: Rubrica expResult = new Rubrica("expResult"); expResult.aggiungiContatto(new Contatto("Giuseppe","Verdi")); **Oracle**: Rubrica expResult = Giuseppe Verdi;

UTC 1.3 Test Rubrica.modificaContatto
Test items: classe Rubrica, metodo modificaContatto
Test: testModificaContatto1 **Case description**: modificare solo il nome di un contatto **Location**: RubricaTest.testModificaContatto1 **Input**: String nuovoNome = "Francesco"; Rubrica expResult = new Rubrica("expResult"); expResult.modificaContatto(c, nuovoNome, "Rossi")); **Oracle**: Rubrica expResult = Francesco Rossi;
Test: testModificaContatto2 **Case description**: modificare solo il cognome di un contatto **Location**: RubricaTest.testModificaContatto2 **Input**: String nuovoCognome = "Adinolfi"; Rubrica expResult = new Rubrica("expResult"); expResult.modificaContatto(c, "Mario", nuovoCognome); **Oracle**: Rubrica expResult = Mario Adinolfi;

Test: testModificaContatto3
Case description: modificare solo numero/i di un contatto
Location: RubricaTest.testModificaContatto3
 	
Input: Telefono nuovoTelefono = new Telefono();
nuovoTelefono.aggiungiNumero("3123456789");
Rubrica expResult = new Rubrica("expResult");
expResult.modificaContatto(c, "Mario", "Rossi", nuovoTelefono);
 	
Oracle: Rubrica expResult = Mario Rossi, 3123456789;

Test: testModificaContatto4 **Case description**: modificare solo email di un contatto **Location**: RubricaTest.testModificaContatto4 **Input**: Email nuovaEmail = new Email();

nuovaEmail.aggiungiEmail("mariorossi@gmail.com");
Rubrica expResult = new Rubrica("expResult");
expResult.modificaContatto(c, "Mario", "Rossi", nuovaEmail));

Oracle: Rubrica expResult = Mario Rossi, mariorossi@gmail.com;

Test: testModificaContatto5
Case description: modifica con nome e cognome null
Location: RubricaTest.testModificaContatto5

Input: Rubrica expResult = new Rubrica("expResult");
expResult.modificaContatto(c, null, null));

Oracle: NominativoAssenteException generata;

Test: testModificaContatto6
Case description: modifica di un contatto null (cambio prima cognome in null e poi il nome)
Location: RubricaTest.testModificaContatto6

Input: Rubrica expResult = new Rubrica("expResult");
expResult.modificaContatto(c, null, null));

Oracle: Eccezione generata;

Test: testModificaContatto7
Case description: modifica di un contatto null (cambio prima nome in null e poi il cognome)
Location: RubricaTest.testModificaContatto7

Input: Rubrica expResult = new Rubrica("expResult");
expResult.aggiungiContatto(c, null, null));

Oracle: Eccezione generata;

Test: testModificaContatto8
Case description: modifica di un contatto nome e cognome
Location: RubricaTest.testModificaContatto8

Input: String nuovoNome = "Antonio";
String nuovoCognome = "Conte";
Rubrica expResult = new Rubrica("expResult");
expResult.aggiungiContatto(c, nuovoNome, nuovoCognome));

Oracle: Rubrica expResult = Antonio Conte;

UTC 1.4 Test Rubrica.eliminaContatto

Test items: classe Rubrica, metodo eliminaContatto

Test: testEliminaContatto
Case description: elimina contatto
Location: RubricaTest.testEliminaContatto

Input: Rubrica expResult = new Rubrica("expResult");
expResult.aggiungiContatti(c);
expResult.eliminaContatto(c);

Oracle: Rubrica vuota;

UTC 1.5 Test Rubrica.ricercaContatto

Test items: classe Rubrica, metodo ricercaContatto

Test: testRicercaContatto
Case description: ricerca contatto
Location: RubricaTest.testRicercaContatto
Input: String sottostringa = "Ros";
Rubrica expResult = new Rubrica("expResult");
expResult.aggiungiContatto(c, “Giovanni”, “Rosalinda”);
expResult.aggiungiContatto(c, “Mario”, “Verdi”);
expResult.ricercaContatto(sottostringa);

Oracle: Rubrica expResult = Giovanni Rosalinda;

UTC 1.6 Test Rubrica.sort

Test items: classe Rubrica, metodo Sort

Test: testSort
Case description: ordina la rubrica
Location: RubricaTest.testSort

Input: Rubrica expResult = new Rubrica("expResult");
expResult.aggiungiContatto(c, “Giovanni”, “Rosalinda”);
expResult.aggiungiContatto(c, “Mario”, “Verdi”);
expResult.sort();

Oracle: Rubrica expResult = Giovanni Rosalinda, Mario Verdi;

UTC 1.7 Test Rubrica.importaRubrica

Test items: classe Rubrica, metodo importaRubrica

Test: testImportaRubrica1
Case description: importa la rubrica da un file csv
Location: RubricaTest.testImportaRubrica1

Input: String filename = "rubImport1.csv";
Rubrica expResult = new Rubrica("expResult");
expResult.aggiungiContatto(c);
r.importRubrica(filename);

Oracle: Rubrica expResult = Mario Rossi;

Test: testImportaRubrica2
Case description: importa un file csv che non ha i campi della rubrica
Location: RubricaTest.testImportaRubrica2

Input: String filename = "rubImport2.csv";
Rubrica expResult = new Rubrica("expResult");
expResult.importRubrica(filename);

Oracle: Eccezione generata.

Test: testImportaRubrica3
Case description: importa un file non csv
Location: RubricaTest.testImportaRubrica3

Input: String filename = "rubImport3.txt";
Rubrica expResult = new Rubrica("expResult");
expResult.aggiugiContatto(c);
expResult.importRubrica(filename);

Oracle: Eccezione generata.

UTC 1.8 Test Rubrica.esportaRubrica

Test items: classe Rubrica, metodo esportaRubrica

Test: testEsportaRubrica
Case description: esporta la rubrica su un file csv
Location: RubricaTest.testEsportaRubrica

Input: String filename = "rubExport.csv";
Rubrica expResult = new Rubrica("expResult");
expResult.aggiungiContatto(c);
expResult.esportaRubrica(filename);

Oracle: Rubrica expResult = Mario Rossi;

UTC 1.9 Test Rubrica.getContatti
Test items: classe Rubrica, metodo getContatti
Test: testGetContatti **Case description**: restituisce i contatti presenti nella rubrica **Location**: RubricaTest.testGetContatti **Input**: Rubrica expResult = new Rubrica("expResult"); expResult.aggiungiContatto(new Contatto(“Francesco”, “Bianchi”); expResult.aggiungiContatto(new Contatto(“Giuseppe”, “Verdi”); expResult.getContatti(); **Oracle**: expResult = Francesco Bianchi, Giuseppe Verdi;

UTC 1.10 Test Rubrica.toString
Test items: classe Rubrica, metodo toString
Test: testToString **Case description**: overriding del metodo toString **Location**: RubricaTest.testToString **Input**: Rubrica expResult = new Rubrica("expResult"); expResult.aggiungiContatto(new Contatto(“Francesco”, “Bianchi”); expResult.aggiungiContatto(new Contatto(“Lucia”, “Rossi”); expResult.toString(); **Oracle**: Francesco Bianchi, Lucia Rossi;

2.2 UNIT TEST CASE: Contatto

Prima di effettuare ogni test, vengono eseguite le seguenti operazioni usando il tag BeforeEach e alla fine di ogni test viene eseguito l'AfterEach:

```
@BeforeEach
    public void setUp() throws NominativoAssenteException {
        telefoni1 = new Telefono();
        telefoni1.aggiungiNumero("3111111111");
        telefoni1.aggiungiNumero("3222222222");
        email1 = new Email();
        email1.aggiungiEmail("mariorossi@example.com");
        contatto1 = new Contatto("Mario","Rossi",telefoni1,email1);
        contatto2 = new Contatto("Giuseppe","Verdi");
    }
```

UTC 2.1 Test Contatto.Constructor
Test items: classe Contatto, metodo Constructor

Test: testConstructor1
Case description: constructor classico
Location: ContattoTest.testConstructor1
 	
Input: String nome = "Francesco";
String cognome = "Bianchi";
Telefono tx = new Telefono();
tx.aggiungiNumero("3123456789");
Email ex = new Email();
ex.aggiungiEmail("francescobianchi@example.com");
Contatto expResult = new Contatto(nome, cognome, tx, ex);
 	
Oracle: Contatto expResult = Francesco Bianchi, 3123456789,
francescobianchi@example.com;

Test: testConstructor2 **Case description**: constructor solo nome e cognome **Location**: ContattoTest.testConstructor2 **Input**: String nome = "Domenico"; String cognome = "Rossi"; Contatto expResult = new Contatto(nome, cognome); **Oracle**: Contatto expResult = Domenico Rossi;
Test: testConstructor3 **Case description**: constructor solo cognome con nome null **Location**: ContattoTest.testConstructor3 **Input**: String cognome = "Rossi"; Contatto expResult = new Contatto(cognome);

Oracle: Contatto expResult = Rossi;

Test: testConstructor4
Case description: constructor solo nome con cognome null
Location: ContattoTest.testConstructor4

Input: String nome = "Gabriele";
Contatto expResult = new Contatto(nome);

Oracle: Contatto expResult = Gabriele;

Test: testConstructor5
Case description: constructor con i valori null
Location: ContattoTest.testConstructor5

Input: Contatto expResult = new Contatto();

Oracle: NominativoAssenteException generata;

Test: testConstructor6
Case description: constructor completo con nome e cognome null
Location: ContattoTest.testConstructor6

Input: Telefono tx = new Telefono();
tx.aggiungiNumero("3123456789");
Email ex = new Email();
ex.aggiungiEmail("francescobianchi@example.com");
Contatto expResult = new Contatto(tx, ex);

Oracle: NominativoAssenteException generata;

Test: testConstructor7
Case description: constructor completo con nome null
Location: ContattoTest.testConstructor7

Input: String cognome = "Bianchi";
Telefono tx = new Telefono();
tx.aggiungiNumero("3123456789");
Email ex = new Email();
ex.aggiungiEmail("francescobianchi@example.com");
Contatto expResult = new Contatto(nome, cognome, tx, ex);

Oracle: Contatto expResult = Bianchi, 3123456789,
francescobianchi@example.com;

UTC 2.2 Test Contatto.getNome

Test items: classe Contatto, metodo getNome

Test: testGetNome
Case description: restituisce il nome di un contatto
Location: ContattoTest.testGetNome

Input: String nome = “Giuseppe”;
Contatto expResult = new Contatto(nome, null);
expResult.getNome();

Oracle: Contatto expResult = Giuseppe;

UTC 2.3 Test Contatto.getCognome
Test items: classe Contatto, metodo getCognome
Test: testGetCognome **Case description**: restituisce il cognome di un contatto **Location**: ContattoTest.testGetCognome **Input**: String cognome = “Marotta”; Contatto expResult = new Contatto(null, cognome); expResult.getCognome(); **Oracle**: Contatto expResult = Marotta;

UTC 2.4 Test Contatto.getNumeri
Test items: classe Contatto, metodo getNumeri
Test: testGetNumeri **Case description**: restituisce il/i numero/i di un contatto **Location**: ContattoTest.testGetNumeri **Input**: String nome = “Francesco”; String cognome = "Bianchi"; Telefono tx = new Telefono(); tx.aggiungiNumero("3123456789"); Contatto expResult = new Contatto(nome, cognome, tx, null); List<String> numeri = expResult.getNumeri(); **Oracle**: Contatto expResult = 3123456789;

UTC 2.5 Test Contatto.getEmail
Test items: classe Contatto, metodo getEmail
Test: testGetEmail **Case description:** restituisce l’email di un contatto **Location**: ContattoTest.testGetEmail **Input**: String nome = “Francesco”; String cognome = "Bianchi"; Email ex = new Email();

ex.aggiungiEmail("francescobianchi@example.com");
Contatto expResult = new Contatto(nome, cognome, null, ex);
List<String> email = expResult.getEmail();

Oracle: Contatto expResult = francescobianchi@example.com;

UTC 2.6 Test Contatto.setNome
Test items: classe Contatto, metodo setNome
Test: testSetNome1 **Case description**: imposta un nome di un contatto **Location**: ContattoTest.testSetNome1 **Input**: String nome = “Carlo”; String cognome = “Franchetti”; Contatto expResult = new Contatto(nome, cognome); expResult.setNome(“GianCarlo”); **Oracle**: Contatto expResult = GianCarlo Franchetti;
Test: testSetNome2 **Case description**: imposta un nome di un contatto con cognome assente **Location**: ContattoTest.testSetNome2 **Input**: String nome = “Carlo”; Contatto expResult = new Contatto(nome, null); expResult.setNome(“GianCarlo”); **Oracle**: Contatto expResult = GianCarlo;

UTC 2.7 Test Contatto.setCognome
Test items: classe Contatto, metodo setCognome
Test: testSetCognome1 **Case description**: imposta il cognome di un contatto **Location**: ContattoTest.testSetCognome1 **Input**: String cognome = “Di Lieto”; Contatto expResult = new Contatto(null, cognome); expResult.setCognome(“Di Marino”); **Oracle**: Contatto expResult = Di Marino;
Test: testSetCognome2 **Case description**: imposta un nome di un contatto con cognome assente **Location**: ContattoTest.testSetCognome2 **Input**: String nome = “Mario”;

UTC 2.7 Test Contatto.setCognome
Test items: classe Contatto, metodo setCognome
Test: testSetCognome1 **Case description**: imposta il cognome di un contatto **Location**: ContattoTest.testSetCognome1 **Input**: String cognome = “Di Lieto”; Contatto expResult = new Contatto(null, cognome); expResult.setCognome(“Di Marino”); **Oracle**: Contatto expResult = Di Marino;
Contatto expResult = new Contatto(nome, null); expResult.setCognome(); **Oracle**: NominativoAssenteException generata;

UTC 2.8 Test Contatto.aggiungiNumero
Test items: classe Contatto, metodo aggiungiNumero
Test: testAggiungiNumero1 **Case description**: aggiunta numeri di un contatto **Location**: ContattoTest.testAggiungiNumero1 **Input**: String nome = “Luisa”; String tx = “3123456789”; Contatto expResult = new Contatto(nome); expResult.aggiungiNumeri(tx); **Oracle**: Contatto expResult = Luisa, 3123456789;

Test: testAggiungiNumero2
Case description: aggiunta numeri di un contatto con 3 numeri già inseriti
Location: ContattoTest.testAggiungiNumero2
 	
Input: String nome = “Luisa”;
String tx1 = “3123456789”;
String tx2 = “3123456790”;
String tx3 = “3123456791”;
Contatto expResult = new Contatto(nome);
expResult.aggiungiNumero(tx1);
expResult.aggiungiNumero(tx2);
expResult.aggiungiNumero(tx3);
Oracle: Contatto expResult = Luisa, 3123456789, 3123456790, 3123456791;

UTC 2.9 Test Contatto.aggiugiEmail
Test items: classe Contatto, metodo aggiungiEmail
Test: testAggiungiEmail1 **Case description**: assegna ad un contatto l'email **Location**: ContattoTest.testAggiungiEmail1 **Input**: String nome = "Luisa"; String ex = "luisa@example.com"; Contatto expResult = new Contatto(nome); expResult.aggiungiEmail(ex); **Oracle**: Contatto expResult = Luisa, luisa@example.com;
Test: testAggiungiEmail2 **Case description**: tentativo di inserire un'email con lista piena **Location**: ContattoTest.testAggiungiEmail2 **Input**: String nome = "Luisa"; String ex1 = "luisa1@example.com"; String ex2 = "luisa2@example.com"; String ex3 = "luisa3@example.com"; String ex4 = "luisa4@example.com"; Contatto expResult = new Contatto(nome, ex1, ex2, ex3); expResult.aggiungiEmail(ex4); **Oracle**: Contatto expResult = Luisa, luisa1@example.com, luisa2@example.com, luisa3@example.com;

UTC 2.10 Test Contatto.compareTo
Test items: classe Contatto, metodo compareTo
Test: testCompareTo1 **Case description**: due contatti con nome e cognome diversi **Location**: ContattoTest.testCompareTo1 **Input**: Contatto c1 = new Contatto("Mario", "Rossi"); Contatto c2 = new Contatto("Giuseppe", "Verdi"); int result = c1.compareTo(c2); **Oracle**: expResult = -1;
Test: testCompareTo2 **Case description**: due contatti con lo stesso cognome ma diverso nome **Location**: ContattoTest.testCompareTo2 **Input**: Contatto c1 = new Contatto("Mario", "Rossi"); Contatto c2 = new Contatto("Giuseppe", "Rossi"); int result = c1.compareTo(c2);

Oracle: expResult = -1;

Test: testCompareTo3
Case description: due contatti con lo stesso cognome e nome
Location: ContattoTest.testCompareTo3

Input: Contatto c1 = new Contatto("Mario", "Rossi");
Contatto c2 = new Contatto("Mario", "Rossi");
int result = c1.compareTo(c2);

Oracle: expResult = 0;

Test: testCompareTo4
Case description: due contatti con lo stesso nome ma diverso cognome
Location: ContattoTest.testCompareTo4

Input: Contatto c1 = new Contatto("Mario", "Rossi");
Contatto c2 = new Contatto("Mario", "Bianchi");
int result = c1.compareTo(c2);

Oracle: expResult = 1;

Test: testCompareTo5
Case description: due contatti con lo stesso nome, ma cognome assente
Location: ContattoTest.testCompareTo5

Input: Contatto c1 = new Contatto("Mario", null);
Contatto c2 = new Contatto("Mario", "Rossi");
int result = c1.compareTo(c2);

Oracle: expResult = -1;

Test: testCompareTo6
Case description: due contatti con lo stesso cognome, ma nomi assenti
Location: ContattoTest.testCompareTo6

Input: Contatto c1 = new Contatto(null, “Rossi”);
Contatto c2 = new Contatto(null, "Rossi");
int result = c1.compareTo(c2);

Oracle: expResult = 0;

Test: testCompareTo7
Case description: due contatti con lo stesso cognome, il primo con nome null
Location: ContattoTest.testCompareTo7

Input: Contatto c1 = new Contatto(null, “Rossi”);
Contatto c2 = new Contatto(“Mario”, "Rossi");
int result = c1.compareTo(c2);

Oracle: expResult = -1;

Test: testCompareTo8
Case description: due contatti con lo stesso cognome, il secondo con nome null
Location: ContattoTest.testCompareTo8

Input: Contatto c1 = new Contatto(“Mario”, “Rossi”);
Contatto c2 = new Contatto(null, "Rossi");
int result = c1.compareTo(c2);

Oracle: expResult = 1;

Test: testCompareTo9
Case description: due contatti, il secondo con cognome null
Location: ContattoTest.testCompareTo9

Input: Contatto c1 = new Contatto(“Mario”, “Rossi”);
Contatto c2 = new Contatto(“Giuseppe”, null);
int result = c1.compareTo(c2);

Oracle: expResult = 1;

Test: testCompareTo10
Case description: due contatti, il primo con cognome null
Location: ContattoTest.testCompareTo10

Input: Contatto c1 = new Contatto(“Mario”, null);
Contatto c2 = new Contatto(“Giuseppe”, “Rossi”);
int result = c1.compareTo(c2);

Oracle: expResult = -1;

UTC 2.11 Test Contatto.toString

Test items: classe Contatto, metodo toString

Test: testToString
Case description: overriding del metodo toString
Location: ContattoTest.testToString

Input: Telefono telefoni1 = new Telefono();
telefoni1.aggiungiNumero("3123456789");
Email email1 = new Email();
email1.aggiungiEmail("mariorossi@example.com");
Contatto contatto = new Contatto("Mario", "Rossi", telefoni1, email1);
expResult.toString();

Oracle: String expResult = "Contatto{nome=Mario, cognome=Rossi,
telefoni=[3123456789], email=[mariorossi@example.com]}"

UTC 2.12 Test Contatto.equals

Test items: classe Contatto, metodo equals

Test: testEquals1
Case description: confronta se due contatti sono uguali
Location: ContattoTest.testEquals1

Input: Telefono telefono1 = new Telefono();
telefono1.aggiungiNumero("1234567890");
Email email1 = new Email();
email1.aggiungiEmail("mario.rossi@example.com");
Contatto contatto1 = new Contatto("Mario", "Rossi", telefono1, email1);
Telefono telefono2 = new Telefono();
telefono2.aggiungiNumero("1234567890");
Email email2 = new Email();
email2.aggiungiEmail("mario.rossi@example.com");
Contatto contatto2 = new Contatto("Mario", "Rossi", telefono2, email2);
contatto1.equals(contatto2);

Oracle: boolean expResult = true;

Test: testEquals2
Case description: confronta che due contatti siano diversi tra loro
Location: ContattoTest.testEquals2

Input: Telefono telefono1 = new Telefono();
telefono1.aggiungiNumero("1234567890");
Email email1 = new Email();
email1.aggiungiEmail("mario.rossi@example.com");
Contatto contatto1 = new Contatto("Mario", "Rossi", telefono1, email1);
Telefono telefono2 = new Telefono();
telefono2.aggiungiNumero("3987654321");
Email email2 = new Email();
email2.aggiungiEmail("mario.rossi@example.com");
Contatto contatto2 = new Contatto("Mario", "Rossi", telefono2, email2);
contatto1.equals(contatto2);

Oracle: boolean expResult = false;

Test: testEquals3
Case description: verifica se il contatto è null
Location: ContattoTest.testEquals3

Input: Telefono telefono = new Telefono();
telefono.aggiungiNumero("1234567890");
Email email = new Email();
email.aggiungiEmail("mario.rossi@example.com");
Contatto contatto = new Contatto("Mario", "Rossi", telefono, email);
contatto.equals(null);

Oracle: boolean expResult = false;

Test: testEquals4
Case description: confronta due oggetti di classi differenti
Location: ContattoTest.testEquals4

Input: Telefono telefono = new Telefono();
telefono.aggiungiNumero("1222222222");
Email email = new Email();
email.aggiungiEmail("domdima@example.com");
Contatto contatto = new Contatto("Domenico", "Di Marino", telefono, email);
contatto.equals(telefono);

Oracle: boolean expResult = false;

Test: testEquals5
Case description: verifica se due contatti hanno lo stesso nome ma non lo stesso cognome
Location: ContattoTest.testEquals5

Input: Contatto contatto1 = new Contatto("Domenico", "Di Marino");
Contatto contatto2 = new Contatto("Domenico", "Marini");
contatto1.equals(contatto2);

Oracle: boolean expResult = false;

Test: testEquals6
Case description: verifica se due oggetti hanno lo stesso nome e cognome, ma uno dei due oggetti è stato modificato dopo la creazione
Location: ContattoTest.testEquals6

Input: Contatto esempio1 = new Contatto("Domenico", "Di Marino");
Contatto esempio2 = new Contatto("Domenico", "Di Marino");
Telefono telefono = new Telefono();
telefono.aggiungiNumero("1222222222");
esempio2.setNumeri(telefono);
esempio.equals(esempio2);

Oracle boolean expResult =false;

Test: testEquals7
Case description: verifica se due oggetti hanno lo stesso nome, cognome e numeri di telefono, ma differiscono per l'email
Location: ContattoTest.testEquals7
Input: Telefono esempioTel1 = new Telefono();
esempioTel1.aggiungiNumero("1222222222");
Email esempioEma1 = new Email();
esempioEma1.aggiungiEmail("domdima@example.com");
Contatto contatto1 = new Contatto("Domenico", "Di Marino", esempioTel1, esempioEma1);
Contatto contatto2 = new Contatto("Domenico", "Di Marino", esempioTel1, esempioEma1);
contatto2.setNumeri(new Telefono());
contatto2.getNumeri().aggiungiNumero("3333333333");
contatto1.equals(contatto2);
Oracle: boolean expResult = false;

UTC 2.13 Test Contatto.hashCode

Test items: classe Contatto, metodo hashCode

Test: testHashCode
Case description: confronta valore di di due oggetti che hanno gli stessi valori per nome, cognome, numeri di telefono e email
Location: ContattoTest.testHashCode

Input: Contatto contatto1 = new Contatto("Domenico", "Di Marino");
Telefono telefono = new Telefono();
telefono.aggiungiNumero("1222222222");
Email email = new Email();
email.aggiungiEmail("domdima@example.com");
contatto1.setNumeri(telefono);
contatto1.setEmail(email);
Contatto contatto2 = new Contatto("Domenico", "Di Marino");
contatto2.setNumeri(telefono);
contatto2.setEmail(email);
contatto1.hashCode();
contatto2.hashCode();

Oracle: expResult → Restituisce un valore uguale per tutte e due gli oggetti.

2.3 UNIT TEST CASE: Email

Prima di effettuare ogni test, vengono eseguite le seguenti operazioni usando il tag BeforeEach:

```
@BeforeEach
    public void setUp() {
        mail = new Email();
    }
```

UTC 3.1 Test Email.constructor
Test items: classe Email, metodo constructor
Test: testConstructor **Case description**: costructor dove verifico che venga creata una lista vuota di email **Location**: EmailTest.testCostructor **Input**: Email email = new Email(); **Oracle**: Email expResult = "Email{[]}";

UTC 3.2 Test Email.aggiungiEmail
Test items: classe Email, metodo aggiungiEmail
Test: testAggiugiEmail1 **Case description**: verifica il corretto funzionamento del metodo aggiungiEmail **Location**: EmailTest.testAggiungiEmail1 **Input**: Email email = new Email(); email.aggiungiEmail("mariorossi@example.com"); **Oracle**: Email expResult = "Email{[mariorossi@example.com]}";

Test: testAggiugiEmail2
Case description: tentativo di aggiunta con già tre email presenti
Location: EmailTest.testAggiungiEmail2
 	
Input: Email expResult = new Email();
expResult.aggiungiEmail("domenicodimarino@example.com");
expResult.aggiungiEmail("giovanniadinolfi@example.com");
expResult.aggiungiEmail("francescodicrescenzo@example.com");
expResult.aggiungiEmail("mariorossi@example.com");
 	
Oracle: Email expResult = domenicodimarino@example.com,
giovanniadinolfi@example.com,
francescodicrescenzo@example.com;

UTC 3.3 Test Email.getSize
Test items: classe Email, metodo getSize
Test: testGetSize **Case description**: restituisce correttamente il numero di email presenti nella lista dopo l'aggiunta di una email. **Location**: EmailTest.testGetSize **Input**: email.aggiungiEmail(“domenicodimarino@example.com”); email.getSize(); **Oracle**: int expResult = 1;

UTC 3.4 Test Email.getEmail
Test items: classe Email, metodo getEmail
Test: testGetEmail **Case description**: restituisce le email **Location**: EmailTest.testGetEmail **Input**: email.aggiungiEmail(“domenicodimarino@example.com”); email.aggiungiEmail(“gionny@example.com”); email.getEmail(); **Oracle**: Email expResult = domenicodimarino@example.com, gionny@example.com;

UTC 3.5 Test Email.getSingolaEmail
Test items: classe Email, metodo getSingolaEmail
Test: testGetSingolaEmail **Case description**: restituisce una mail specifica **Location**: EmailTest.testGetSingolaEmail **Input**: email.aggiungiEmail(“domenicodimarino@example.com”); email.aggiungiEmail(“gionny@example.com”); email.getSingolaEmail(0); **Oracle**: String expResult = domenicodimarino@example.com;

UTC 3.6 Test Email.setEmail
Test items: classe Email, metodo setEmail

Test: testGetSetEmail
Case description: reimposta le email
Location: EmailTest.testSetEmail
 	
Input: Email expResult = new Email();
expResult.aggiungiEmail("domenicomarini@example.com");
Email email = new Email();
email.aggiungiEmail("giovannimarino@example.com");
expResult.setEmail(telefono.getEmail());
 	
Oracle: List<String> expResult = new ArrayList<>();
expected.add("giovannimarino@example.com");

UTC 3.7 Test Email.toString
Test items: classe Email, metodo toString

Test: testToString
Case description: sovrascrive Object
Location: EmailTest.testToString
 	
Input: email.aggiugiEmail("mariorossi@example.com");
email.aggiugiEmail("giovanniadinolfi@example.com");
email.toString();
 	
Oracle: Email expResult = "Email{[mariorossi@example.com,
giovanniadinolfi@example.com]}"

UTC 3.8 Test Email.equals
Test items: classe Email, metodo equals

Test: testEquals1
Case description: verifica se due email sono uguali
Location: EmailTest.testEquals1
 	
Input: Email expResult = new Email();
expResult.aggiungiEmail("domenicodimarino@example.com");
Email mail = new Email();
mail.aggiungiEmail("domenicodimarino@example.com");
expResult.equals(mail);
 	
Oracle: boolean expResult = true;

Test: testEquals2 **Case description**: confronta una email con un oggetto null

Location: EmailTest.testEquals2

Input: Email expResult = new Email();
expResult.aggiungiEmail("domenicodimarino@example.com");
expResult.equals(null);
Oracle: boolean expResult = false;

Test: testEquals3
Case description: confronta una mail con un oggetto differente
Location: EmailTest.testEquals3

Input: Email expResult = new Email();
expResult.aggiungiEmail("domenicodimarino@example.com");
String mail = "domenicodimarino@example.com";
expResult.equals(mail);

Oracle: boolean expResult = false;

2.4 UNIT TEST CASE: Telefono

Prima di effettuare ogni test, vengono eseguite le seguenti operazioni usando il tag BeforeEach:

```
@BeforeEach
    public void setUp() {
        telefono = new Telefono();
    }
```

UTC 4.1 Test Telefono.constructor
Test items: classe Telefono, metodo constructor
Test: testConstructor **Case description**: costructor dove verifico che venga creata una lista vuota di numeri **Location**: TelefonoTest.testCostructor **Input**: Telefono telefono = new Telefono(); **Oracle**: Telefono expResult = "Telefono{[]}";

UTC 4.2 Test Telefono.aggiungiNumero
Test items: classe Numero, metodo aggiungiNumero
Test: testAggiugiNumero1 **Case description**: verifica il corretto funzionamento del metodo aggiungiNumero. **Location**: TelefonoTest.testAggiungiNumero1 **Input**: Telefono telefono = new Telefono(); telefono.aggiungiNumero("1234567890"); **Oracle**: Telefono expResult = "Telefono{[1234567890]}";

Test: testAggiugiNumero2
Case description: tentativo di aggiunta con già tre numeri presenti
Location: TelefonoTest.testAggiungiNumero2
 	
Input: Telefono expResult = new Telefonol();
expResult.aggiungiNumero("1234567890");
expResult.aggiungiNumero("3987654321");
expResult.aggiungiNumero("2638519672");
expResult.aggiungiNumero("1953728963");
 	
Oracle: Telefono expResult = "Telefono{[1234567890, 3987654321, 2638519672]}";

UTC 4.3 Test Telefono.getSize
Test items: classe Telefono, metodo getSize
Test: testGetSize **Case description**: restituisce correttamente il numero di numeri di telefono presenti nella lista dopo l'aggiunta di una numero. **Location**: TelefonoTest.testGetSize **Input**: telefono.aggiungiNumero(“1234567890”); telefono.getSize(); **Oracle**: int expResult = 1;

UTC 4.4 Test Telefono.getNumeri
Test items: classe Telefono, metodo getNumeri
Test: testGetNumeri **Case description**: restituisce i numeri presenti **Location**: TelefonoTest.testGetNumeri **Input**: telefono.aggiungiNumeri(“1234567890”); telefono.aggiungiNumeri(“2134532345”); telefono.getNumeri(); **Oracle**: List<String> expResult = ["1234567890", "2134532345"];

UTC 4.5 Test Telefono.getSingoloNumero
Test items: classe Telefono, metodo getSingoloNumero
Test: testGetSingoloNumero **Case description**: restituisce un numero di telefono specifico **Location**: TelefonoTest.testGetSingoloNumero **Input**: telefono.aggiungiNumero(“1234567890”); telefono.aggiungiNumero(“2134532345”); telefono.getSingoloNumero(1); **Oracle**: String expResult = “2134532345”;

UTC 4.6 Test Telefono.toString
Test items: classe Telefono, metodo toString

Test: testToString
Case description:
Location: TelefonoTest.testToString
 	
Input: telefono.aggiugiNumero("1234567890");
telefono.aggiugiNumero("2134532345");
telefono.toString();
 	
Oracle: String expResult = “Telefono{[1234567890, 2134532345]}”;

UTC 4.7 Test Telefono.setNumeri
Test items: classe Telefono, metodo setNumeri

Test: testSetNumeri
Case description: reimposta i numeri di telefono
Location: TelefonoTest.testSetNumeri
 	
Input: Telefono expResult = new Telefono();
expResult.aggiungiNumero("4444444444");
Telefono telefono = new Telefono();
telefono.aggiungiNumero("3333333333");
expResult.setNumeri(telefono.getNumeri());
 	
Oracle: String expResult = “3333333333”;

UTC 4.8 Test Telefono.equals
Test items: classe Telefono, metodo equals

Test: testEquals1
Case description: verifica se due numeri di telefono sono uguali
Location: TelefonoTest.testEquals1
 	
Input: Telefono expResult = new Telefono();
expResult.aggiungiNumero("1111111111");
Telefono telefono = new Telefono();
telefono.aggiungiNumero("1111111111");
expResult.equals(telefono);
 	
Oracle: boolean expResult = true;

Test: testEquals2 **Case description**: confronta un numero di telefono con un oggetto null **Location**: TelefonoTest.testEquals2

Input: Telefono expResult = new Telefono();
expResult.aggiungiNumero("1111111111");
expResult.equals(null);

Oracle: boolean expResult = false;

Test: testEquals3
Case description: confronta numeri di telefono di classi differenti
Location: TelefonoTest.testEquals3

Input: Telefono expResult = new Telefono();
expResult.aggiungiNumero("1111111111");
String numero = "1111111111";
expResult.equals(numero);

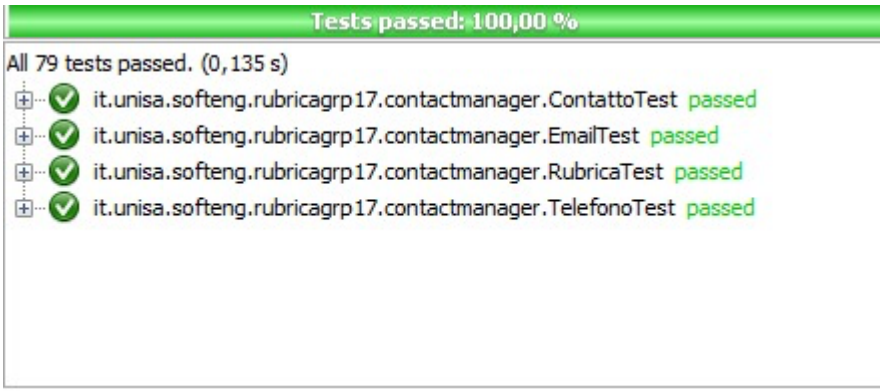
Oracle: boolean expResult = false;

3.0 Usability Testing

Il test effettuato da persone esterne ha dato esito positivo, dimostrando le perfomance attese in termini di semplicità di utilizzo ed efficienza delle varie funzioni che sono state introdotte nella rubrica. Ad esempio, la ricerca di un contatto tramite una sottostringa inserita nella barra di ricerca.
Non sono sorti problemi riguardanti l'utilizzo dell'interfaccia grafica da parte degli utenti che l'hanno testata.

4.0 Test Report

Risultati dei test automatizzati ottenuti da Netbeans.



Tutti i test sono stati superati con successo.