PACKAGE Model

CASI DI TEST PER LA CLASSE Libro

UTC 1.1	Test Libro - costruttore
Test items	Class Libro, costruttore
Inputs	Libro libro = new Libro(“L’idiota”, “Dostoevskij”, 2023, “9788854175037”, 1)
Oracle	libro.getTitolo() == “L’idiota” libro.getAutori() == “Dostoevskij” libro.getAnnoDiPubblicazione() == 2023 libro.getIsbn() == “9788854175037” libro.getNumeroCopieDisponibili() == 1

UTC 1.2.1	Test Libro – controllo di validità – validità accertata
Test items	Class Libro, isValid
Inputs	Libro libro = new Libro(“L’idiota”, “Dostoevskij”, 2023, “9788854175037”, 1)
Oracle	libro.isValid() == true

UTC 1.2.2	Test Libro – controllo di validità – controllo fallito a causa di isbn non standard
Test items	Class Libro, isValid
Inputs	Libro libro1 = new Libro(“L’idiota”, “Dostoevskij”, 2023, “9268854175037”, 1) Libro libro2 = new Libro(“L’idiota”, “Dostoevskij”, 2023, “9788854”, 1)
Oracle	libro1.isValid() == false libro2.isValid() == false

UTC 1.2.3	Test Libro – controllo validità – controllo fallito a causa di numero di copie minore di 1
Test items	Class Libro, isValid
Inputs	Libro libro1 = new Libro(“L’idiota”, “Dostoevskij”, 2023, “9268854175037”, 0)
Oracle	libro.autoriProperty() != null

UTC 1.3	Test Libro – confronto tra due libri uguali
Test items	Class Libro, equals
Inputs	Libro libro = new Libro(“L’idiota”, “Dostoevskij”, 2023, “9788854175037”, 1) Libro libro1 = new Libro(“”, “”, 0, “9788854175037”, 0)
Oracle	libro.equals(libro1) == true

UTC 1.4.1	Test Libro – ordinamento lessicografico tra i titoli di due libri – libro “maggiore” di libro1
Test items	Class Libro, compareTo

Inputs	Libro libro = new Libro(“L’idiota”, “Dostoevskij”, 2023, “9788854175037”, 1) Libro libro1 = new Libro(“Il libro rosso”, “”, 0, “”, 0)
Oracle	libro.compareTo(libro1) > 0

UTC 1.4.2	Test Libro – ordinamento lessicografico tra i titoli di due libri – libro “uguale” a libro1
Test items	Class Libro, compareTo
Inputs	Libro libro = new Libro(“L’idiota”, “Dostoevskij”, 2023, “9788854175037”, 1) Libro libro1 = new Libro(“L’idiota”, “”, 0, “”, 0)
Oracle	libro.compareTo(libro1) == 0

UTC 1.4.3	Test Libro – ordinamento lessicografico tra i titoli di due libri – libro “minore” a libro1
Test items	Class Libro, compareTo
Inputs	Libro libro = new Libro(“L’idiota”, “Dostoevskij”, 2023, “9788854175037”, 1) Libro libro1 = new Libro(“L’uomo invisibile”, “”, 0, “”, 0)
Oracle	libro.compareTo(libro1) < 0

UTC 1.5	Test Libro – stampa della classe
Test items	Class Libro, toString
Inputs	Libro libro = new Libro(“L’idiota”, “Dostoevskij”, 2023, “9788854175037”, 1)
Oracle	libro.toString() != null

CASI DI TEST PER LA CLASSE Utente

UTC 2.1	Test Utente – costruttore
Test items	Class Utente, costruttore
Inputs	Utente utente = new Utente(“Francesco”, “Vecchione”, “0612709314”, “f.vecchione17@studenti.unisa.it”)
Oracle	utente.getNome() == “Francesco” utente.getCognome() == “Vecchione” utente.getMatricola() == “0612709314” utente.getEmail() == “f.vecchione17@studenti.unisa.it”

UTC 2.2.1	Test Utente - controllo di validità – validità accertata
Test items	Class Utente, isValid
Inputs	Utente utente = new Utente(“Francesco”, “Vecchione”, “0612709314”, “f.vecchione17@studenti.unisa.it”) Utente utente1 = new Utente(“Francesco”, “Vecchione”, “0612709314”, “f.vecchione@studenti.unisa.it”)
Oracle	Utente.isValid() == true Utente1.isValid() == true

UTC 2.2.2	Test Utente - controllo di validità – controllo fallito a causa di matricola errata
Test items	Class Utente, isValid()
Inputs	Utente utente1 = new Utente(“Francesco”, “Vecchione”, “06127AA314”, “f.vecchione17@studenti.unisa.it”) Utente utente2 = new Utente(“Francesco”, “Vecchione”, “921826”, “f.vecchione17@studenti.unisa.it”)
Oracle	Utente1.isValid() == false Utente2.isValid() == false

UTC 2.2.3	Test Utente – controllo di validità – controllo fallito a causa di email errata
Test items	Class Utente, isValid()
Inputs	Utente utente1 = new Utente(“Francesco”, “Vecchione”, “0612709314”, “fran.vecchione17@studenti.unisa.it”) Utente utente2 = new Utente(“Francesco”, “Vecchione”, “0612709314”, “f.vecc17@studenti.unisa.it”) Utente utente3 = new Utente(“Francesco”, “Vecchione”, “0612709314”, “fran.vecchione17studenti.unisa.it”)
Oracle	Utente1.isValid() == false

	Utente2.isValid() == false Utente3.isValid() == false

UTC 2.3	Test Utente – confronto tra due utenti uguali
Test items	Class Utente, equals
Inputs	Utente utente = new Utente(“Francesco”, “Vecchione”, “0612709314”, “f.vecchione17@studenti.unisa.it”) Utente utente1 = new Utente(“”, “”, “0612709314”, “”)
Oracle	Utente.equals(utente1) == true

UTC 2.4.1	Test Utente – ordinamento lessicografico su nome e cognome – utente “maggiore” di utente1
Test items	Class Utente, compareTo
Inputs	Utente utente = new Utente(“Francesco”, “Vecchione”, “0612709314”, “f.vecchione17@studenti.unisa.it”) Utente utente1 = new Utente(“Daniel”, “Vita”, “”, “”)
Oracle	Utente.compareTo(utente1) > 0

UTC 2.4.2	Test Utente - ordinamento lessicografico su nome e cognome – utente “uguale” a utente1
Test items	Class Utente, compareTo
Inputs	Utente utente = new Utente(“Francesco”, “Vecchione”, “0612709314”, “f.vecchione17@studenti.unisa.it”) Utente utente1 = new Utente(“Francesco”, “Vecchione”, “”, “”)
Oracle	Utente.compareTo(utente1) == 0

UTC 2.4.3	Test Utente - ordinamento lessicografico su nome e cognome – utente “minore” di utente1
Test items	Class Utente, compareTo
Inputs	Utente utente = new Utente(“Francesco”, “Vecchione”, “0612709314”, “f.vecchione17@studenti.unisa.it”) Utente utente1 = new Utente(“Marco”, “Del Gaudio”, “”, “”)
Oracle	Utente.compareTo(utente1) < 0

UTC 2.5	Test Utente – stampa della classe
Test items	Class Utente, toString
Inputs	Utente utente = new Utente(“Francesco”, “Vecchione”, “0612709314”, “f.vecchione17@studenti.unisa.it”)
Oracle	Utente.toString() != null

CASI DI TEST PER LA CLASSE Prestito

UTC 3.1	Test Prestito – costruttore
Test items	Class Prestito, costruttore
Inputs	Prestito prestito = new Prestito(“0612709314”, “9788854175037”, LocalDate.of(2025, 12, 12), LocalDate.of(2026, 1, 13))
Oracle	Prestito.getMatricolaUtente() == “0612709314” prestito.getIsbnPrestito() == “9788854175037” prestito.getDataPrestito().equals(LocalDate.of(2025, 12, 12)) == true prestito.getDataRestituzione().equals(LocalDate.of(2026, 1, 13)) == true prestito.getStatoPrestito() == StatoPrestito.ATTIVO

UTC 3.2.1	Test Prestito – controllo validità – validità riscontrata
Test items	Class Prestito, isValid
Inputs	Prestito prestito = new Prestito(“0612709314”, “9788854175037”, LocalDate.of(2025, 12, 12), LocalDate.of(2026, 1, 13))
Oracle	Prestito.isValid() == true

UTC 3.2.2	Test Prestito – controllo validità – controllo fallito a causa di data restituzione precedente a data di prestito
Test items	Class Prestito, isValid
Inputs	Prestito prestito1 = new Prestito(“”, “”, LocalDate.of(2025, 12, 12), LocalDate.of(2025, 12, 10)
Oracle	Prestito1.isValid() == false

UTC 3.3.1	Test Prestito – confronto tra due oggetti uguali – confronto positivo
Test items	Class Prestito, equals
Inputs	Prestito prestito = new Prestito(“0612709314”, “9788854175037”, LocalDate.of(2025, 12, 12), LocalDate.of(2026, 1, 13)) Prestito prestito1 = new Prestito(“0612709314”, “9788854175037”, null, null)
Oracle	Prestito.equals(prestito1) == true

UTC 3.3.2	Test Prestito – confronto tra due oggetti uguali – confronto negativo data matricola diversa
Test items	Class Prestito, equals
Inputs	Prestito prestito = new Prestito(“0612709314”, “9788854175037”, LocalDate.of(2025, 12, 12), LocalDate.of(2026, 1, 13))

	Prestito prestito1 = new Prestito(“0612709323”, “9788854175037”, null, null)
Oracle	Prestito.equals(prestito1) == false

UTC 3.3.3	Test Prestito – confronto tra due oggetti uguali – confronto negativo dato isbn diverso
Test items	Class Prestito, equals
Inputs	Prestito prestito = new Prestito(“0612709314”, “9788854175037”, LocalDate.of(2025, 12, 12), LocalDate.of(2026, 1, 13)) Prestito prestito1 = new Prestito(“0612709314”, “9788823145037”, null, null)
Oracle	Prestito.equals(prestito1) == false

UTC 3.4.1	Test Prestito – ordinamento cronologico sulla data di restituzione – prestito “maggiore” di prestito1
Test items	Class Prestito, compareTo
Inputs	Prestito prestito = new Prestito(“0612709314”, “9788854175037”, LocalDate.of(2025, 12, 12), LocalDate.of(2026, 1, 13)) Prestito prestito1 = new Prestito(“”, “”, null, LocalDate.of(2026, 1, 10))
Oracle	Prestito.compareTo(prestito1) > 0

UTC 3.4.2	Test Prestito – ordinamento cronologico sulla data di restituzione – prestito “uguale” a prestito1
Test items	Class Prestito, compareTo
Inputs	Prestito prestito = new Prestito(“0612709314”, “9788854175037”, LocalDate.of(2025, 12, 12), LocalDate.of(2026, 1, 13)) Prestito prestito1 = new Prestito(“”, “”, null, LocalDate.of(2026, 1, 13))
Oracle	Prestito.compareTo(prestito1) == 0

UTC 3.4.3	Test Prestito – ordinamento cronologico sulla data di restituzione – prestito “minore” di prestito1
Test items	Class Prestito, compareTo
Inputs	Prestito prestito = new Prestito(“0612709314”, “9788854175037”, LocalDate.of(2025, 12, 12), LocalDate.of(2026, 1, 13)) Prestito prestito1 = new Prestito(“”, “”, null, LocalDate.of(2026, 1, 16))
Oracle	Prestito.compareTo(prestito1) < 0

UTC 3.5	Test Prestito – stampa della classe
Test items	Class Prestito, toString

Inputs	Prestito prestito = new Prestito(“0612709314”, “9788854175037”, LocalDate.of(2025, 12, 12), LocalDate.of(2026, 1, 13))
Oracle	Prestito.toString() != null

CASI DI TEST PER LA CLASSE Password

UTC 4.1.1	Test Password – controllo della password – controllo passato con successo
Test items	Class Password, verificaPassword
Inputs	Password password = new Password(“cassaforte”) Password.impostaPassword(“ciaoMondo”);
Oracle	Password.verificaPassword(“ciaoMondo”) == true

UTC 4.1.2	Test Password – controllo della password – controllo fallito a causa di password differenti
Test items	Class Password, verificaPassword
Inputs	Password password = new Password(“cassaforte”) Password.impostaPassword(“ciaoMondo”);
Oracle	Password.verificaPassword(“biblioteca”) == falso

UTC 4.2	Test Password – reimpostazione della password
Test items	Class Password, impostaPassword
Inputs	Password password = new Password(“cassaforte”) Password.impostaPassword(“ciaoMondo”);
Oracle	Password.impostaPassword(“biblioteca”) == true Password.verificaPassword(“biblioteca”) == true

UTC 4.3.1	Test Password – controllare se sia presente una password – caso vero
Test items	Class Password, esistePassword
Inputs	Password password = new Password(“cassaforte”) impostaPassword(“ciaoMondo”);
Oracle	Password.esistePassword() == true

UTC 4.3.2	Test Password – controllare se sia presente una password – caso false
Test items	Class Password, esistePassword
Inputs	Password password = new Password(“cassaforte”)
Oracle	Password.esistePassword() == false

Alla fine di ogni test viene eliminato il file di cassaforte

CASI DI TEST PER LA CLASSE ModelArchivio

UTC 5.1	Test ModelArchivio – apertura archivio
Test items	Class ModelArchivio, apriArchivio
Inputs	ModelArchivio<Libro> mal = new ModelArchivio<>(“archivioLibri”)
Oracle	Mal.apriArchivio() Mal.getArchivioFiltrato() != null

UTC 5.2	Test ModelArchivio – chiusura archivio
Test items	Class ModelArchivio chiudiArchivio
Inputs	ModelArchivio<Libro> mal = new ModelArchivio<>(“archivioLibri”) mal.apriArchivio()
Oracle	mal.chiudiArchivio() == true

UTC 5.3	Test ModelArchivio – getter archivio filtrato
Test items	Class ModelArchivio getArchivioFiltrato
Inputs	ModelArchivio<Libro> mal = new ModelArchivio<>(“ archivioLibri”) mal.apriArchivio()
Oracle	mal.getArchivioFiltrato() != null

UTC 5.4	Test ModelArchivio – aggiunta elemento
Test items	Class ModelArchivio aggiungiElemento
Inputs	ModelArchivio<Libro> mal = new ModelArchivio<>(“ archivioLibri”), mal.apriArchivio(), Libro l1 = new Libro(“Test”, ...),
Oracle	mal.aggiungiElemento(l1) == true, mal.getArchivioFiltrato().contains(l1) == true

UTC 5.5	Test ModelArchivio – rimozione elemento
Test items	Class ModelArchivio rimuoviElemento
Inputs	ModelArchivio<Libro> mal = new ModelArchivio<>(“ archivioLibri”), mal.apriArchivio(), Libro l1 = new Libro(“Test”, ...), mal.aggiungiElemento(l1)
Oracle	mal.rimuoviElemento(l1) == true, mal.getArchivioFiltrato().contains(l1) == false

UTC 5.6	Test ModelArchivio – modifica elemento
Test items	Class ModelArchivio modificaElemento
Inputs	ModelArchivio<Libro> mal = new ModelArchivio<>(“ archivioLibri”), mal.apriArchivio(), Libro oldL = new Libro(“Old Book”, ...),

	Libro newL = new Libro(“New Book”, ...), mal.aggiungiElemento(oldL)
Oracle	mal.modificaElemento(oldL,newL) == true int index = mal.getArchivioFiltrato().indexOf(newL) mal.getArchivioFiltrato().get(index).getTitolo() == “New Book”

UTC 5.7.1	Test ModelArchivio – stress test archivio libri con 150000 entry – chiusura archivio sotto 500 millisecondi
Test items	Class ModelArchivio, chiudiArchivio
Inputs	String STRESS_FILE_TEST = “testFiles/stressArchivioLibri” stressModel = new ModelArchivio<>(STRESS_FILE_TEST); stressModel.apriArchivio();
Oracle	assertTimeout(Duration.ofMillis(500), () -> { stressModel.chiudiArchivio() == true });

UTC 5.7.2	Test ModelArchivio – stress test archivio libri con 150000 entry – apertura archivio sotto 500 millisecondi
Test items	Class ModelArchivio, apriArchivio
Inputs	String STRESS_FILE_TEST = “testFiles/stressArchivioLibri” stressModel = new ModelArchivio<>(STRESS_FILE_TEST);

Oracle	assertTimeout(Duration.ofMillis(500), () -> {
	stressModel.apriArchivio();
	});

UTC 5.7.3	Test ModelArchivio – stress test archivio libri con 150000 entry – aggiunta elemento sotto 100 millisecondi
Test items	Class ModelArchivio, aggiungiElemento
Inputs	String STRESS_FILE_TEST = “testFiles/stressArchivioLibri” stressModel = new ModelArchivio<>(STRESS_FILE_TEST); stressModel.apriArchivio(); Libro l1 = new Libro("Il principe", "Machiavelli", 2023, "9781237475633", 2);
Oracle	assertTimeout(Duration.ofMillis(100), () -> { stressModel.aggiungiElemento(l1) == true });

UTC 5.7.4	Test ModelArchivio – stress test archivio libri con 150000 entry – modifica elemento sotto 100 millisecondi
Test items	Class ModelArchivio, modificaElemento

Inputs	String STRESS_FILE_TEST = “testFiles/stressArchivioLibri”
	stressModel = new ModelArchivio<>(STRESS_FILE_TEST);
	stressModel.apriArchivio();
	Libro l1 = new Libro("L'idiota", "Dostoevskij", 2023, "9781237475664", 3);
	Libro l2 = new Libro("Francesco", "Vecchione", 2023, "9781237475664", 5);
	stressModel.aggiungiElemento(l1);

Oracle	assertTimeout(Duration.ofMillis(100), () -> { stressModel.modificaElemento(l1, l2) == true });

UTC 5.7.5	Test ModelArchivio – stress test archivio libri con 150000 entry – elimina elemento sotto 100 millisecondi
Test items	Class ModelArchivio, eliminaElemento

Inputs	String STRESS_FILE_TEST = “testFiles/stressArchivioLibri”
	stressModel = new ModelArchivio<>(STRESS_FILE_TEST);
	stressModel.apriArchivio();
	Libro l1 = new Libro("Francesco", "Vecchione", 2023, "9781237475664", 5);
	stressModel.aggiungiElemento(l1);

Oracle	assertTimeout(Duration.ofMillis(100), () -> { stressModel.rimuoviElemento(l1) == true });

Alla fine di ogni operazione vengono eliminati sia il file di cache che di archivio, meno i file per lo stress test

PACKAGE io

CASI DI TEST PER LA CLASSE CacheRecord

UTC 6.1.1	Test CacheRecord – costruttore Libro – operazione di aggiunta
Test items	Class CacheRecord<Libro>, costruttore
Inputs	CacheRecord<Libro> cr = new CacheRecord<>(TipoOperazione.AGGIUNTA, null, new Libro("L'idiota", "Dostoevskij", 2023, "9788854175037", 1))
Oracle	Cr.getTipoOperazione() == TipoOperazione.AGGIUNTA Cr.getTarget() == null Cr.getElem().getTitolo() == "L'idiota" Cr.getElem().getAutori() == "Dostoevskij" Cr.getElem().getAnnoPubblicazione() == 2023 Cr.getElem().getIsbn() == "9788854175037" Cr.getElem().getNumeroCopieDisponibili() == 1

UTC 6.1.2	Test CacheRecord – costruttore Libro – operazione di modifica
Test items	Class CacheRecord<Libro>, costruttore
Inputs	CacheRecord<Libro> cr = new CacheRecord<>(TipoOperazione.MODIFICA, new Libro("L'idiota", "Dostoevskij", 2023, "9788854175037", 1), new Libro("L'idiota", "Francesco Vecchione", 2004, "9788854175037", 5))
Oracle	Cr.getTipoOperazione() == TipoOperazione.MODIFICA Cr.getTarget().getTitolo() == "L'idiota" Cr.getTarget().getAutori() == "Dostoevskij" Cr.getTarget().getAnnoPubblicazione() == 2023 Cr.getTarget().getIsbn() == "9788854175037" Cr.getTarget().getNumeroCopieDisponibili() == 1 Cr.getElem().getTitolo() == "L'idiota" Cr.getElem().getAutori() == "Francesco Vecchione" Cr.getElem().getAnnoPubblicazione() == 2004 Cr.getElem().getIsbn() == "9788854175037" Cr.getElem().getNumeroCopieDisponibili() == 5

UTC 6.1.3	Test CacheRecord – costruttore Libro – operazione di cancellazione
Test items	Class CacheRecord<Libro>, costruttore
Inputs	CacheRecord<Libro> cr = new CacheRecord<>(TipoOperazione.CANCELLAZIONE, new Libro("L'idiota", "Dostoevskij", 2023, "9788854175037", 1), null)
Oracle	Cr.getTipoOperazione() == TipoOperazione.CANCELLAZIONE Cr.getTarget().getTitolo() == "L'idiota"

	Cr.getTarget ().getAutori() == “Dostoevskij” Cr.getTarget ().getAnnoPubblicazione() == 2023 Cr.getTarget ().getIsbn() == “9788854175037” Cr.getTarget().getNumeroCopieDisponibili() == 1 Cr.getElem() == null

UTC 6.2.1	Test CacheRecord – costruttore Utente – operazione di aggiunta
Test items	Class CacheRecord<Utente>, costruttore
Inputs	CacheRecord<Utente> cr = new CacheRecord<>(TipoOperazione.AGGIUNTA, null, new Utente(“Francesco”, “Vecchione”, “0612709314”, “f.vecchione17@studenti.unisa.it”))
Oracle	Cr.getTipoOperazione() == TipoOperazione.AGGIUNTA Cr.getTarget() == null Cr.getElem().getNome() == “Francesco” Cr.getElem().getCognome() == “Vecchione” Cr.getElem().getMatricola() == “0612709314” Cr.getElem().getEmail() == “f.vecchione17@studenti.unisa.it”

UTC 6.2.2	Test CacheRecord – costruttore Utente – operazione di modifica
Test items	Class CacheRecord<Utente>, costruttore
Inputs	CacheRecord<Utente> cr = new CacheRecord<>(TipoOperazione.MODIFICA, neww Utente(“Francesco”, “Vecchione”, “0612709314”, “f.vecchione17@studenti.unisa.it”), new Utente(“Maciu”, “Piciu”, “0612709314”, “m.piciu@studenti.unisa.it”))
Oracle	Cr.getTipoOperazione() == TipoOperazione.MODIFICA Cr.getTarget().getNome() == “Francesco” Cr.getTarget().getCognome() == “Vecchione” Cr.getTarget().getMatricola() == “0612709314” Cr.getTarget().getEmail() == “f.vecchione17@studenti.unisa.it” Cr.getElem().getNome() == “Maciu” Cr.getElem().getCognome() == “Piciu” Cr.getElem().getMatricola() == “0612709314” Cr.getElem().getEmail() == “m.piciu@studenti.unisa.it”

UTC 6.2.3	Test CacheRecord – costruttore Utente – operazione di cancellazione
Test items	Class CacheRecord<Utente>, costruttore
Inputs	CacheRecord<Utente> cr = new CacheRecord<>(TipoOperazione.CANCELLAZIONE,

	new Utente(“Francesco”, “Vecchione”, “0612709314”, “f.vecchione17@studenti.unisa.it”), null)
Oracle	Cr.getTipoOperazione() == TipoOperazione.CANCELLAZIONE Cr.getTarget().getNome() == “Francesco” Cr.getTarget().getCognome() == “Vecchione” Cr.getTarget().getMatricola() == “0612709314” Cr.getTarget().getEmail() == “f.vecchione17@studenti.unisa.it”

UTC 6.3.1	Test CacheRecord – costruttore Prestito – operazione di aggiunta
Test items	Class CacheRecord<Prestito>, costruttore
Inputs	CacheRecord<Prestito> cr = new CacheRecord<>(TipoOperazione.AGGIUNTA, null, new Prestito(“0612709314”, “9788854175037”, LocalDate.of(2025, 12, 12), LocalDate.of(2026, 1, 13)))
Oracle	Cr.getTipoOperazione() == TipoOperazione. Cr.getTarget() == null Cr.getElem().getMatricolaUtente() == “0612709314” Cr.getElem().getIsbnPrestito() == “9788854175037” Cr.getElem().getDataPrestito() == LocalDate.of(2025, 12, 12) Cr.getElem().getDataRestituzione() == LocalDate.of(2026, 1, 13)

UTC 6.3.2	Test CacheRecord – costruttore Prestito – operazione di modifica
Test items	Class CacheRecord<Prestito>, costruttore
Inputs	CacheRecord<Prestito> cr = new CacheRecord<>(TipoOperazione.MODIFICA, new Prestito(“0612701283”, “9788854175222”, LocalDate.of(2025, 12, 30), LocalDate.of(2026, 2, 13), new Prestito(“0612709314”, “9788854175037”, LocalDate.of(2025, 12, 12), LocalDate.of(2026, 1, 13))
Oracle	Cr.getTipoOperazione() == TipoOperazione.

UTC 6.3.3	Test CacheRecord – costruttore Prestito – operazione di cancellazione
Test items	Class CacheRecord<Prestito>, costruttore
Inputs	CacheRecord<Prestitoe> cr = new CacheRecord<>(TipoOperazione.CANCELLAZIONE, new Prestito(“0612701283”, “9788854175222”,

	LocalDate.of(2025, 12, 30), LocalDate.of(2026, 2, 13), null)
Oracle	Cr.getTipoOperazione() == TipoOperazione. Cr.getTarget().getMatricolaUtente() == “0612701283” Cr.getTarget().getIsbnPrestito() == “9788854175222” Cr.getTarget().getDataPrestito() == LocalDate.of(2025, 12, 30) Cr.getTarget().getDataRestituzione() == LocalDate.of(2026, 2, 13) Cr.getElem() == null

CASI DI TEST PER LA CLASSE Cassaforte

UTC 7.1	Test Cassaforte - funzione di codifica password
Test items	Class Cassaforte, criptaPassword
Inputs	Cassaforte cf = new Cassaforte(“cassaforte”) String passwordInChiaro1 = “@ksfutlSTTn2&3s” String passwordInChiaro2 = “\’\’\”\”\’\’\’””
Oracle	Cf.criptaPassword(passwordInChiaro1) == passwordInChiaro1.hashCode() Cf.criptaPassword(passwordInChiaro2) == passwordInChiaro2.hashCode()

UTC 7.2	Test Cassaforte – metodo per salvare la password in un file – reimpostazione della password
Test items	Class Cassaforte, salvaPasswordCriptata
Inputs	Cassaforte cf = new Cassaforte(“cassaforte”) String passwordInChiaro1 = “@ksfutlSTTn2&3s” String passwordInChiaro2 = “\’\’\”\”\’\’\’””
Oracle	Cf.salvaPasswordCriptata(passwordInChiaro1) == true cf.leggiPasswordCriptata(passwordInChiaro1) == Cf.criptaPassword(passwordInChiaro1) Cf.salvaPasswordCriptata(passwordInChiaro2) == true cf.leggiPasswordCriptata(passwordInChiaro2) == Cf.criptaPassword(passwordInChiaro2)

UTC 7.3.1	Test Cassaforte – metodo per leggere la password salvata – caso 1
Test items	Class Cassaforte, leggiPasswordCriptata
Inputs	Cassaforte cf = new Cassaforte(“cassaforte”) String passwordInChiaro1 = “@ksfutlSTTn2&3s” Cf.salvaPasswordCriptata(passwordInChiaro1)
Oracle	cf.leggiPasswordCriptata(passwordInChiaro1) == Cf.criptaPassword(passwordInChiaro1)

UTC 7.3.2	Test Cassaforte - metodo per leggere la password salvata – caso 2
Test items	Class Cassaforte, leggiPasswordCriptata
Inputs	Cassaforte cf = new Cassaforte(“cassaforte”) String passwordInChiaro2 = “\’\’\”\”\’\’\’””” Cf.salvaPasswordCriptata(passwordInChiaro2)
Oracle	cf.leggiPasswordCriptata(passwordInChiaro2) == Cf.criptaPassword(passwordInChiaro2)

Alla fine di ogni test viene eliminato il file “cassaforte”.

CASI DI TEST PER LA CLASSE GestoreCache

UTC 8.1.1	Test GestoreCache – costruttore per cache Libri
Test items	Class GestoreCache<Libro>, costruttore
Inputs	String pathLibri = “archivioLibriCache” GestoreCache<Libro> gcl = new GestoreCache<>(pathLibri)
Oracle	New File(pathLibri).exists() == true

UTC 8.1.2	Test GestoreCache – costruttore per cache Utenti
Test items	Class GestoreCache<Utente>, costruttore
Inputs	String pathUtenti = “archivioUtentiCache” GestoreCache<Utente> gcu = new GestoreCache<>(pathUtenti)
Oracle	New File(pathUtenti).exists() == true

UTC 8.1.3	Test GestoreCache – costruttore per cache Prestiti
Test items	Class GestoreCache<Prestito>, costruttore
Inputs	String pathPrestiti = “archivioPrestitiCache” GestoreCache<Prestito> gcp = new GestoreCache<>(pathPrestiti)
Oracle	New File(pathPrestiti).exists() == true

UTC 8.2.1	Test GestoreCache – salva un cache record per archivio Libri
Test items	Class GestoreCache<Libro>, salvaSuCache
Inputs	String pathLibri = “archivioLibriCache” GestoreCache<Libro> gcl = new GestoreCache<>(pathLibri) CacheRecord<Libro> crl1 = new CacheRecord<>(TipoOperazione.AGGIUNTA, null, new Libro(“L’idiota”, “Dostoevskij”, 2023, “9788854175037”, 1)) CacheRecord<Libro> crl2 = new CacheRecord<>(TipoOperazione.MODIFICA, new Libro(“L’idiota”, “Dostoevskij”, 2023, “9788854175037”, 1), new Libro(“L’idiota”, “Francesco Vecchione”, 2004, “9788854175037”, 5)) CacheRecord<Libro> crl3 = new CacheRecord<>(TipoOperazione.CANCELLAZIONE, new Libro(“L’idiota”, “Dostoevskij”, 2023, “9788854175037”, 1), null)
Oracle	Gcl.salvaSuCache(crl1) == true Gcl.salvaSuCache(crl2) == true

	Gcl.salvaSuCache(crl3) == true

UTC 8.2.2	Test GestoreCache – salva un cache record per archivio Utenti
Test items	Class GestoreCache<Utente>, salvaSuCache
Inputs	String pathUtenti = “archivioUtentiCache” GestoreCache<Utente> gcu = new GestoreCache<>(pathUtente) CacheRecord<Utente> cru1 = new CacheRecord<>(TipoOperazione.MODIFICA, neww Utente(“Francesco”, “Vecchione”, “0612709314”, “f.vecchione17@studenti.unisa.it”), new Utente(“Maciu”, “Piciu”, “0612709314”, “m.piciu@studenti.unisa.it”)) CacheRecord<Utente> cru2 = new CacheRecord<>(TipoOperazione.AGGIUNTA, null, new Utente(“Francesco”, “Vecchione”, “0612709314”, “f.vecchione17@studenti.unisa.it”)) CacheRecord<Utente> cru3 = new CacheRecord<>(TipoOperazione.CANCELLAZIONE, new Utente(“Francesco”, “Vecchione”, “0612709314”, “f.vecchione17@studenti.unisa.it”), null)
Oracle	Gcl.salvaSuCache(cru1) == true Gcl.salvaSuCache(cru2) == true Gcl.salvaSuCache(cru3) == true

UTC 8.2.3	Test GestoreCache – salva un cache record per archivio Prestiti
Test items	Class GestoreCache<Prestito>, salvaSuCache
Inputs	String pathPrestiti = “archivioPrestitiCache” GestoreCache<Prestito> gcp = new GestoreCache<>(pathPrestito) CacheRecord<Prestito> crp1 = new CacheRecord<>(TipoOperazione.MODIFICA, new Prestito(“0612701283”, “9788854175222”, LocalDate.of(2025, 12, 30), LocalDate.of(2026, 2, 13), new Prestito(“0612709314”, “9788854175037”, LocalDate.of(2025, 12, 12), LocalDate.of(2026, 1, 13)) CacheRecord<Prestito> crp2 = new CacheRecord<>(TipoOperazione.CANCELLAZIONE, new Prestito(“0612701283”, “9788854175222”, LocalDate.of(2025, 12, 30), LocalDate.of(2026, 2, 13), null)

	CacheRecord<Prestito> crp3 = new CacheRecord<>(TipoOperazione.AGGIUNTA, null, new Prestito(“0612709314”, “9788854175037”, LocalDate.of(2025, 12, 12), LocalDate.of(2026, 1, 13)))
Oracle	Gcl.salvaSuCache(crp1) == true Gcl.salvaSuCache(crp2) == true Gcl.salvaSuCache(crp3) == true

UTC 8.3.1	Test GestoreCache – carica da cache una lista di record per Libri
Test items	Class GestoreCache<Libro>, caricaDaCache
Inputs	String pathLibri = “archivioLibriCache” GestoreCache<Libro> gcl = new GestoreCache<>(pathLibri) CacheRecord<Libro> crl1 = new CacheRecord<>(TipoOperazione.AGGIUNTA, null, new Libro(“L’idiota”, “Dostoevskij”, 2023, “9788854175037”, 1)) CacheRecord<Libro> crl2 = new CacheRecord<>(TipoOperazione.MODIFICA, new Libro(“L’idiota”, “Dostoevskij”, 2023, “9788854175037”, 1), new Libro(“L’idiota”, “Francesco Vecchione”, 2004, “9788854175037”, 5)) CacheRecord<Libro> crl3 = new CacheRecord<>(TipoOperazione.CANCELLAZIONE, new Libro(“L’idiota”, “Dostoevskij”, 2023, “9788854175037”, 1), null) Gcl.salvaSuCache(crl1) Gcl.salvaSuCache(crl2) Gcl.salvaSuCache(crl3) List<CacheRecord<Libro>> test = new LinkedList<>() Test.add(crl1) Test.add(crl2) Test.add(crl3)

Oracle	List<CacheRecord<Libro>> loaded = gcl.caricaDaCache()
	
	For i in [0, 2] {
	  Loaded.get(i).getTipoOperazione() == test.get(i).getTipoOperazione()
	  Loaded.get(i).getTarget() == test.get(i).getTarget()
	  Loaded.get(i).getElem() == test.get(i).getElem()
	}

UTC 8.3.2	Test GestoreCache – carica da cache una lista di record per Utenti
Test items	Class GestoreCache<Utente>, caricaDaCache
Inputs	String pathUtenti = “archivioUtentiCache” GestoreCache<Utente> gcu = new GestoreCache<>(pathUtente) CacheRecord<Utente> cru1 = new CacheRecord<>(TipoOperazione.MODIFICA, neww Utente(“Francesco”, “Vecchione”, “0612709314”, “f.vecchione17@studenti.unisa.it”), new Utente(“Maciu”, “Piciu”, “0612709314”, “m.piciu@studenti.unisa.it”)) CacheRecord<Utente> cru2 = new CacheRecord<>(TipoOperazione.AGGIUNTA, null, new Utente(“Francesco”, “Vecchione”, “0612709314”, “f.vecchione17@studenti.unisa.it”)) CacheRecord<Utente> cru3 = new CacheRecord<>(TipoOperazione.CANCELLAZIONE, new Utente(“Francesco”, “Vecchione”, “0612709314”, “f.vecchione17@studenti.unisa.it”), null) Gcu.salvaSuCache(cru1) Gcu.salvaSuCache(cru2) Gcu.salvaSuCache(cru3) List<CacheRecord<Utente>> test = new LinkedList<>() Test.add(cru1) Test.add(cru2) Test.add(cru3)

Oracle	List<CacheRecord<Utente>> loaded = gcu.caricaDaCache()
	
	For i in [0, 2] {
	  Loaded.get(i).getTipoOperazione() == test.get(i).getTipoOperazione()
	  Loaded.get(i).getTarget() == test.get(i).getTarget()
	  Loaded.get(i).getElem() == test.get(i).getElem()
	}

UTC 8.3.3	Test GestoreCache – carica da cache una lista di record per Prestiti
Test items	Class GestoreCache<Prestito>, caricaDaCache
Inputs	String pathPrestiti = “archivioPrestitiCache” GestoreCache<Prestito> gcp = new GestoreCache<>(pathPrestito)

	CacheRecord<Prestito> crp1 = new CacheRecord<>(TipoOperazione.MODIFICA, new Prestito("0612701283", "9788854175222", LocalDate.of(2025, 12, 30), LocalDate.of(2026, 2, 13), new Prestito("0612709314", "9788854175037", LocalDate.of(2025, 12, 12), LocalDate.of(2026, 1, 13)) CacheRecord<Prestito> crp2 = new CacheRecord<>(TipoOperazione.CANCELLAZIONE, new Prestito("0612701283", "9788854175222", LocalDate.of(2025, 12, 30), LocalDate.of(2026, 2, 13), null) CacheRecord<Prestito> crp3 = new CacheRecord<>(TipoOperazione.AGGIUNTA, null, new Prestito("0612709314", "9788854175037", LocalDate.of(2025, 12, 12), LocalDate.of(2026, 1, 13))) Gcp.salvaSuCache(crp1) Gcp.salvaSuCache(crp2) Gcp.salvaSuCache(crp3) List<CacheRecord<Prestito>> test = new LinkedList<>() Test.add(crp1) Test.add(crp2) Test.add(crp3)

Oracle	List<CacheRecord<Prestito>> loaded = gcu.caricaDaCache()
	
	For i in [0, 2] {
	Loaded.get(i).getTipoOperazione() == test.get(i).getTipoOperazione()
	Loaded.get(i).getTarget() == test.get(i).getTarget()
	Loaded.get(i).getElem() == test.get(i).getElem()
	}

UTC 8.4.1	Test GestoreCache – eliminazione file di cache Libri
Test items	Class GestoreCache<Libro>, eliminaCache
Inputs	String pathLibri = "archivioLibriCache" GestoreCache<Libro> gcl = new GestoreCache<>(pathLibri)
Oracle	Gcl.eliminaCache() == true New File(pathLibri).exists() == false

UTC 8.4.2	Test GestoreCache – eliminazione file di cache Utenti
Test items	ì Class GestoreCache<Utente>, eliminaCache

Inputs	String pathUtenti = “archivioUtentiCache” GestoreCache<Utente> gcu = new GestoreCache<>(pathUtenti)
Oracle	Gcu.eliminaCache() == true New File(pathUtenti).exists() == false

UTC 8.4.3	Test GestoreCache – eliminazione file di cache Prestiti
Test items	Class GestoreCache<Prestito>, eliminaCache
Inputs	String pathPrestiti = “archivioPrestitiCache” GestoreCache<Prestito> gcp = new GestoreCache<>(pathPrestiti)
Oracle	Gcp.eliminaCache() == true newFile(pathPrestiti).exists() == false

Alla fine di ogni test viene eliminato il file di cache.

CASI DI TEST PER LA CLASSE GestoreIO

UTC 9.1.1	Test GestoreIO – costruttore io Libri
Test items	Class GestoreIO<Libro>, costruttore
Inputs	String pathLibri = “archivioLibri” GestoreIO<Libro> giol = new GestoreIO<>(pathLibri)
Oracle	New File(pathLibri).exists() == true

UTC 9.1.2	Test GestoreIO – costruttore io Utenti
Test items	Class GestoreIO<Utente>, costruttore
Inputs	String pathUtenti = “archivioUtenti” GestoreIO<Utente> giou = new GestoreIO<>(pathUtenti)
Oracle	New File(pathUtenti).exists() == true

UTC 9.1.3	Test GestoreIO – costruttore io Prestiti
Test items	Class GestoreIO<Prestito>, costruttore
Inputs	String pathPrestiti = “archivioPrestiti” GestoreIO<Prestiti> giop = new GestoreIO<>(pathPrestiti)
Oracle	New File(pathPrestiti).exists() == true

UTC 9.2.1	Test GestoreIO – salvataggio dell’archivio per Libri
Test items	Class GestoreIO<Libro>, salvaArchivio
Inputs	String pathLibri = “archivioLibri” GestoreIO<Libro> giol = new GestoreIO<>(pathLibri) ObservableList<Libro> archivio = FXCollections.observableArrayList() Archivio.add(new Libro(“Lolita”, “Nobokov”, 1955, “9788845912542”, 2) Archivio.add(new Libro(“L’idiota”, “Dostoevskij”, 1868, “9788854175037”, 3) Archivio.add(new Libro(“Padri e figli”, “Turgenev”, 1862, “9788883373893”, 1)
Oracle	Giol.salvaArchivio(archivio) == true New File(pathLibri + “Cache”).exists == false

UTC 9.2.2	Test GestoreIO – salvataggio dell’archivio per Utenti
Test items	Class GestoreIO<Utente>, salvaArchivio
Inputs	String pathUtenti = “archivioUtenti”

	GestoreIO<Utente> giou = new GestoreIO<>(pathUtenti) ObservableList<Utente> archivio = FXCollections.observableArrayList() Archivio.add(new Utente("Francesco", "Vecchione", "0612709314", "f.vecchione17@studenti.unisa.it")) Archivio.add(new Utente("Francesco", "Pisaturo", "0612709177", "f.pisaturo@studenti.unisa.it")) Archivio.add(new Utente("Matteo", "Sirignano", "0612709812", "m.sirignano2@studenti.unisa.it"))
Oracle	Giou.salvaArchivio(archivio) == true New File(pathUtenti + "Cache").exists == false

UTC 9.2.3	Test GestoreIO – salvataggio dell'archivio per Prestiti
Test items	Class GestoreIO<Prestito>, costruttore
Inputs	String pathPrestiti = "archivioPrestiti" GestoreIO<Prestiti> giop = new GestoreIO<>(pathPrestiti) ObservableList<Prestito> archivio = FXCollections.observableArrayList() Archivio.add(new Prestito("0612709177", "9788883373893", LocalDate.of(2025, 12, 12), LocalDate.of(2026, 1, 10))) Archivio.add(new Prestito("0612709314", "9788854175037", LocalDate.of(2025, 12, 12), LocalDate.of(2026, 1, 8))) Archivio.add(new Prestito("0612709812", "9788854175037", LocalDate.of(2025, 12, 13), LocalDate.of(2026, 1, 10)))
Oracle	Giop.salvaArchivio(archivio) == true New File(pathPrestiti + "Cache").exists == false

UTC 9.3.1.1	Test GestoreIO – caricamento archivio per libri quando l'archivio è vuoto e la cache è vuota
Test items	Class GestoreIO<Libro>, caricaArchivio
Inputs	String pathPrestiti = "archivioPrestiti" GestoreIO<Prestiti> giop = new GestoreIO<>(pathPrestiti)
Oracle	Giop.caricaArchivio().isEmpty() == true

UTC 9.3.1.2	Test GestoreIO – caricamento archivio per libri quando l'archivio è non vuoto e la cache è vuota
Test items	Class GestoreIO<Libro>, caricaArchivio
Inputs	String pathLibri = "archivioLibri" GestoreIO<Libro> giol = new GestoreIO<>(pathLibri) ObservableList<Libro> archivio = FXCollections.observableArrayList() Archivio.add(new Libro("Lolita", "Nobokov", 1955, "9788845912542", 2) Archivio.add(new Libro("L'idiota", "Dostoevskij", 1868, "9788854175037", 3) Archivio.add(new Libro("Padri e figli", "Turgenev", 1862, "9788883373893", 1) Giol.salvaArchivio(archivio)

Oracle	ObservableList<Libro> retrieved = Giop.caricaArchivio()
	
	For i in [0, 2] {
	  Retrieved.get(i).getTitolo() == archivio.get(i).getTitolo()
	  Retrieved.get(i).getAutori() == archivio.get(i).getAutori()
	  Retrieved.get(i).getAnnoDiPubblicazione() == archivio.get(i).getAnnoDiPubblicazione()
	  Retrieved.get(i).getIsbn() == archivio.get(i).getIsbn()
	  Retrieved.get(i).getNumeroCopieDisponibili() = archivio.get(i).getNumeroCopieDisponibili()
	}

UTC 9.3.1.3	Test GestoreIO – caricamento archivio per libri quando l'archivio è non vuoto e la cache è non vuota
Test items	Class GestoreIO<Libro>, caricaArchivio
Inputs	String pathLibri = "archivioLibri" GestoreIO<Libro> giol = new GestoreIO<>(pathLibri) Libro l1 = new Libro("Lolita", "Nobokov", 1955, "9788845912542", 2) Libro l2 = new Libro("L'idiota", "Dostoevskij", 1868, "9788854175037", 3) Libro l3 = new Libro("Guerra e pace", "Tolstoj", 1869, "9788883375873", 3) Libro l4 = new Libro("L'uomo invisibile", "Wells", 1895, "9788845912542", 1)

	ObservableList<Libro> archivio = FXCollections.observableArrayList() Archivio.add(l1) Archivio.add(l2) Giol.salvaArchivio(archivio) Giol.salvaModificaArchivio(new CacheRecord<>(TipoOperazione.CANCELLAZIONE, l2, null) Giol.salvaModificaArchivio(new CacheRecord<>(TipoOperazione.AGGIUNTA, null, l3) Giol.salvaModificaArchivio(new CacheRecord<>(TipoOperazione.MODIFICA, l1, l4)
Oracle	ObservableList<Libro> retrieved = Giop.caricaArchivio() Int index = Retrieved.indexOf(l4) Retrieved.contains(l2) == false Retrieved.contains(l3) == true Retrieved.get(index).getTitolo() == l4.getTitolo() Retrieved.get(index).getAutori() == l4.getAutori() Retrieved.get(index).getAnnoDiPubblicazione() == l4.getAnnoDiPubblicazione() Retrieved.get(index).getIsbn() == l4.getIsbn() Retrieved.get(index).getNumeroCopieDisponibibili() == l4.getNumeroCopieDisponibili()

UTC 9.3.2.1	Test GestoreIO – caricamento archivio per utenti quando l'archivio è vuoto e la cache è vuota
Test items	Class GestoreIO<Utente>, caricaArchivio
Inputs	String pathUtenti = "archivioUtenti" GestoreIO<Utente> giou = new GestoreIO<>(pathUtenti)
Oracle	Giou.caricaArchivio().isEmpty() == true

UTC 9.3.2.2	Test GestoreIO – caricamento archivio per utenti quando l'archivio è non vuoto e la cache è vuota
Test items	Class GestoreIO<Utente>, caricaArchivio
Inputs	String pathUtenti = "archivioUtenti" GestoreIO<Utente> giou = new GestoreIO<>(pathUtenti) ObservableList<Utenti> archivio = FXCollections.observableArrayList() Archivio.add(new Utente("Francesco", "Pisaturo", "0612709311", "f.pisaturo1@studenti.unisa.it"))

	Archivio.add(new Utente(“Matteo”, “Sirignano”, “0612709365”, m.sirignano@studenti.unisa.it)) Archivio.add(new Utente(“Giovanni”, “Scelzo”, “0612709314”, “g.scelzo44@studenti.unisa.it”)) Giou.salvaArchivio(archivio)

Oracle	ObservableList<Utente> retrieved = Giop.caricaArchivio()
	
	For i in [0, 2] {
	Retrieved.get(i).getNome() == archivio.get(i).getNome()
	Retrieved.get(i).getCognome() == archivio.get(i).getCognome()
	Retrieved.get(i).getMatricola() == archivio.get(i).getMatricola()
	Retrieved.get(i).getEmail() == archivio.get(i).getEmail()
	}

UTC 9.3.2.3	Test GestoreIO – caricamento archivio per utenti quando l’archivio è non vuoto e la cache è non vuota
Test items	Class GestoreIO<Utente>, caricaArchivio
Inputs	String pathUtenti = “archivioUtenti” GestoreIO<Utente> giou = new GestoreIO<>(pathUtenti) Utente u1 = new Utente(“Francesco”, “Pisaturo”, “0612709311”, “f.pisaturo1@studenti.unisa.it”)) Utente u2 = new Utente(“Matteo”, “Sirignano”, “0612709365”, m.sirignano@studenti.unisa.it)) Utente u3 = new Utente(“Giovanni”, “Scelzo”, “0612709314”, “g.scelzo44@studenti.unisa.it”)) Utente u4 = new Utente(“Francesco”, “Vecchione”, “0612709314”, f.vecchione17@studenti.unisa.it) ObservableList<Utenti> archivio = FXCollections.observableArrayList() Archivio.add(u1) Archivio.add(u3) Giol.salvaArchivio(archivio) Giol.salvaModificaArchivio(new CacheRecord<>(TipoOperazione.CANCELLAZIONE, u1, null) Giol.salvaModificaArchivio(new CacheRecord<>(TipoOperazione.AGGIUNTA, null, u2) Giol.salvaModificaArchivio(new CacheRecord<>(TipoOperazione.MODIFICA, u3, u4)

Oracle	ObservableList<Utente> retrieved = Giop.caricaArchivio() Int index = retrieved.indexOf(u4) Retrieved.contains(u1) == false Retrieved.contains(u2) == true Retrieved.get(index).getNome() == u4.getNome() Retrieved.get(index).getCognome() == u4.getCognome() Retrieved.get(index).getMatricola() == u4.getMatricola() Retrieved.get(index).getEmail() == u4.getEmail()

UTC 9.3.3.1	Test GestoreIO – caricamento archivio per prestiti quando l'archivio è vuoto e la cache è vuota
Test items	Class GestoreIO<Prestito>, caricaArchivio
Inputs	String pathPrestiti = "archivioPrestiti" GestoreIO<Prestiti> giop = new GestoreIO<>(pathPrestiti)
Oracle	Giop.caricaArchivio().isEmpty() == true

UTC 9.3.3.2	Test GestoreIO – caricamento archivio per prestiti quando l'archivio è non vuoto e la cache è vuota
Test items	Class GestoreIO<Prestito>, caricaArchivio
Inputs	String pathPrestiti = "archivioPrestiti" GestoreIO<Prestiti> giop = new GestoreIO<>(pathPrestiti) ObservableList<Prestiti> archivio = FXCollections.observableArrayList() Archivio.add(new Prestito("0612709177", "9788883373893", LocalDate.of(2025, 12, 12), LocalDate.of(2026, 1, 10))) Archivio.add(new Prestito("0612709314", "9788854175037", LocalDate.of(2025, 12, 12), LocalDate.of(2026, 1, 8))) Archivio.add(new Prestito("0612709812", "9788854175037", LocalDate.of(2025, 12, 13), LocalDate.of(2026, 1, 10))) Giop.salvaArchivio(archivio)

Oracle	ObservableList<Prestito> retrieved = Giop.caricaArchivio()
	
	For i in [0, 2] {
	  Retrieved.get(i).getMatricolaUtente() == archivio.get(i).getMatricolaUtente()
	  Retrieved.get(i).getIsbnPrestito() == archivio.get(i).getIsbnPrestito()

	Retrieved.get(i).getDataPrestito() == archivio.get(i).getDataPrestito() Retrieved.get(i).getDataRestituzione() == archivio.get(i).getDataRestituzione() }

UTC 9.3.3.3	Test GestoreIO – caricamento archivio per prestiti quando l'archivio è non vuoto e la cache è non vuota
Test items	Class GestoreIO<Prestito>, caricaArchivio
Inputs	String pathPrestiti = "archivioPrestiti" GestoreIO<Prestiti> giop = new GestoreIO<>(pathPrestiti) Prestito p1 = new Prestito("0612709177", "9788883373893", LocalDate.of(2025, 12, 12), LocalDate.of(2026, 1, 10)) Prestito p2 = new Prestito("0612709314", "9788854175037", LocalDate.of(2025, 12, 12), LocalDate.of(2026, 1, 8)) Prestito p3 = new Prestito("0612709812", "9788854175037", LocalDate.of(2025, 12, 13), LocalDate.of(2026, 1, 10)) Prestito p4 = new Prestito("0612709812", "9788854175037", LocalDate.of(2025, 12, 13), LocalDate.of(2026, 1, 31)) ObservableList<Prestiti> archivio = FXCollections.observableArrayList() Archivio.add(p3) Archivio.add(p2) Giop.salvaArchivio(archivio) Giop.salvaModificaArchivio(new CacheRecord<>(TipoOperazione.CANCELLAZIONE, p2, null) Giol.salvaModificaArchivio(new CacheRecord<>(TipoOperazione.AGGIUNTA, null, p1) Giol.salvaModificaArchivio(new CacheRecord<>(TipoOperazione.MODIFICA, p3, p4)
Oracle	ObservableList<Prestito> retrieved = Giop.caricaArchivio() Int index = retrieved.indexOf(p4) Retrieved.contains(p2) == false Retrieved.contains(p1) == true Retrieved.get(index).getMatricolaUtente() == u4.getMatricolaUtente() Retrieved.get(index).getIsbnPrestito() == u4.getIsbnPrestito()

	Retrieved.get(index).getDataPrestito() == u4.getDataPrestito() Retrieved.get(index). getDataRestituzione() == u4. getDataRestituzione()

Alla fine di ogni test viene eliminato il file di cache e di archivio

PACKAGE View

CASI DI TEST PER LA CLASSE CreazionePasswordDialog

UTC 10	Test CreazionePasswordDialog – costruttore
Test items	Class CreazionePasswordDialog, costruttore
Inputs	CreazionePasswordDialog d = new CreazionePasswordDialog();
Oracle	d.getTxfPassword() != null d.getTxfConfermaPassword() != null

CASI DI TEST PER LA CLASSE LibriDialog

UTC 11.1	Test LibriDialog – costruttore
Test items	Class LibriDialog, costruttore
Inputs	LibriDialog d1 = new LibriDialog();
Oracle	D1.getTxfTitolo() != null D1.getTxfAutori() != null D1.getTxfAnnoDiPubblicazione() != null D1.getTxfIsbn() != null D1.getTxfNumCopie() != null D1.getDialog() != null

UTC 11.2	Test LibriDialog – costruttore sovraccarico
Test items	Class LibriDialog, costruttore sovraccarico
Inputs	LibriDialog d2 = new LibriDialog(new Libro(“L’idiota”, “Dostoevskij”, 2023, “9788854175037”, 3))
Oracle	D2.getTxfTitolo().getText() == “L’idiota” D2.getTxfAutori().getText() == “Dostoevskij” Integer.parseInt(d.getTxfAnnoDiPubblicazione().getText()) == 2023 D2.getTxfIsbn().getText() == “9788854175037” D2.getTxfNumCopie().getText() == 3

CASI DI TEST PER LA CLASSE LoginDialog

UTC 12	Test LoginDialog – costruttore
Test items	Class LoginDialog, costruttore
Inputs	LoginDialog d = new LoginDialog();
Oracle	d.getTxfPassword() != null d.getLinkPassowordDimenticata () != null

CASI DI TEST PER LA CLASSE UtentiDialog

UTC 13.1	Test UtentiDialog – costruttore
Test items	Class UtentiDialog, costruttore
Inputs	UtentiDialog d1 = new UtentiDialog();
Oracle	D1.getTxfNome()!= null D1.getTxfCognome()!= null D1.getTxfMatricola()!= null D1.getTxfEmail()!= null

	d.getDialog() != null

UTC 13.2	Test UtentiDialog – costruttore sovraccarico
Test items	Class UtentiDialog, costruttore sovraccarico
Inputs	UtentiDialog d2 = new UtentiDialog(new Utente(“Francesco”, “Pisaturo”, “0612709311”, “f.pisaturo1@studenti.unisa.it”));
Oracle	D2.getTxfNome().getText() == “Francesco” D2.getTxfCognome().getText() == “Pisaturo” D2.getTxfMatricola().getText() == “0612709311” D2.getTxfEmail().getText() == “f.pisaturo1@studenti.unisa.it”

CASI DI TEST PER LA CLASSE TabArchivioLibri

UTC 14	Test TabArchivioLibri – costruttore
Test items	Class TabArchivioLibri, costruttore
Inputs	TabArchivioLibri t = new TabArchivioLibri();
Oracle	t.getBtnAggiungi() != null t.getBtnModifica() != null t.getBtnCerca() != null t.getBtnEliminaFiltri() != null t.getTab() != null t.getSelectedItem() == null t.getTxfFiltroRicerca() != null t.getTabella() != null t.getBoxCentro() != null t.getBtnCancella() != null

CASI DI TEST PER LA CLASSE TabArchivioPrestiti

UTC 15	Test TabArchivioPrestiti – costruttore
Test items	Class TabArchivioPrestiti, costruttore
Inputs	TabArchivioPrestiti t = new TabArchivioPrestiti();
Oracle	t.getBtnAggiungi() != null t.getBtnModifica() != null t.getBtnCerca() != null t.getBtnEliminaFiltri() != null t.getTab() != null t.getSelectedItem() == null t.getTxfFiltroRicerca() != null t.getTabella() != null t.getBoxCentro() != null

CASI DI TEST PER LA CLASSE TabArchivioUtenti

UTC 16	Test TabArchivioUtenti – costruttore
Test items	Class TabArchivioUtenti, costruttore
Inputs	TabArchivioUtenti t = new TabArchivioUtenti();
Oracle	t.getBtnAggiungi() != null

	t.getBtnModifica() != null t.getBtnCerca() != null t.getBtnEliminaFiltri() != null t.getTab() != null t.getSelectedItem() == null t.getTxfFiltroRicerca() != null t.getTabella() != null t.getBoxCentro() != null t.getBtnCancella() != null

CASI DI TEST PER LA CLASSE ViewBiblioteca

UTC 17	Test ViewBiblioteca – costruttore
Test items	Class ViewBiblioteca, costruttore
Inputs	ViewBiblioteca v = new ViewBiblioteca (new Stage(), null, null, null);
Oracle	v.getStage() != null v.getTabLibri() != null v.getTabUtenti() != null v.getTabPrestiti() != null