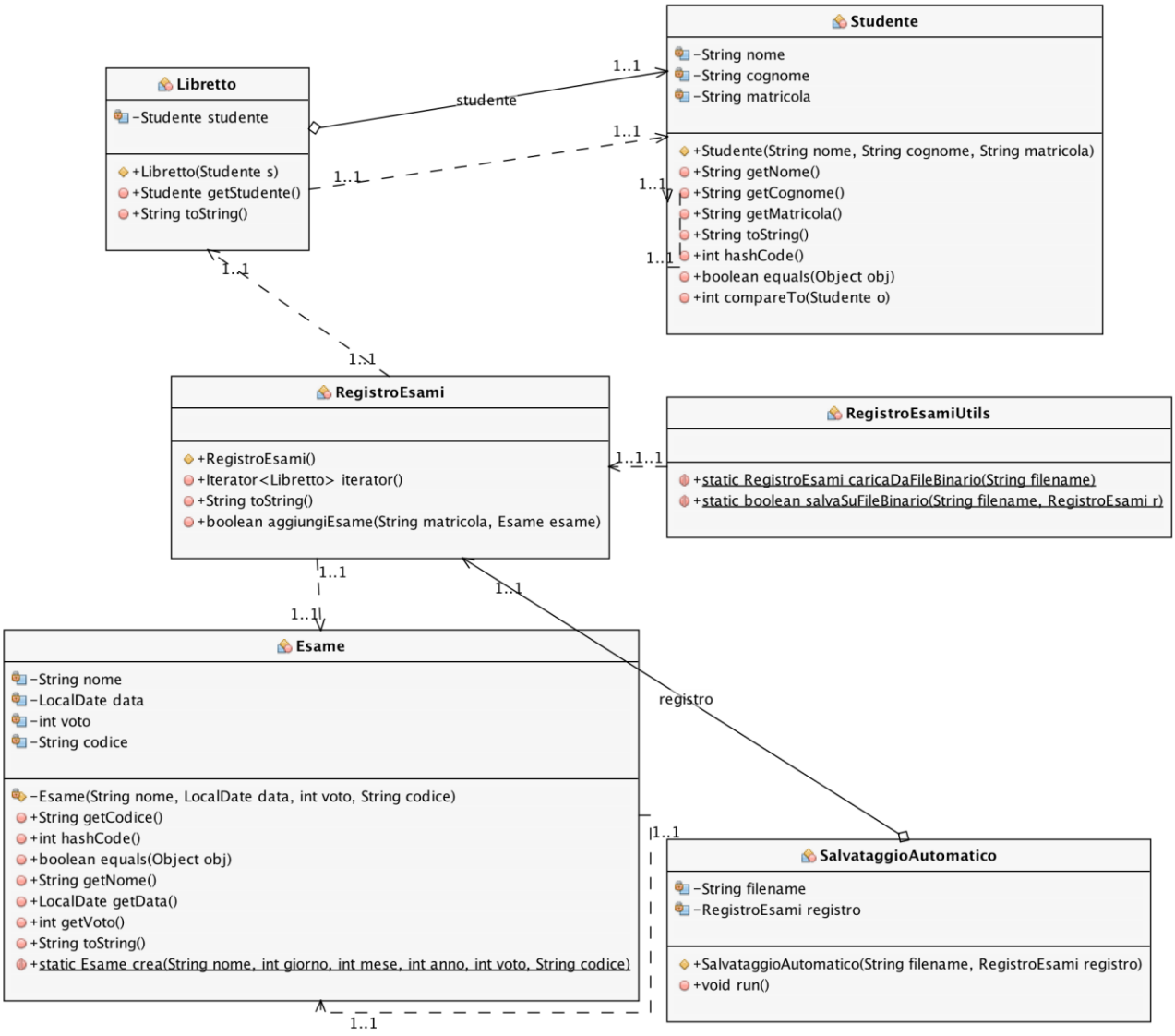
Università degli Studi di Salerno
Corso di Laurea in Ingegneria Informatica
Programmazione ad Oggetti
Esame scritto del 10 gennaio 2017

Si chiede di implementare le classi di seguito descritte necessarie alla realizzazione di un'applicazione per la gestione delle registrazioni degli esami di studenti universitari.



Studente

La classe Studente consente di modellare l'entità omonima del dominio del problema caratterizzata dagli attributi di tipo String nome, cognome e matricola. Tali attributi possono essere impostati solo tramite il costruttore e non sono più modificabili. La classe rende disponibili i metodi di accesso agli attributi in sola lettura. Inoltre la classe sovrascrive i metodi equals ed hashcode in modo che siano coerenti tra di loro e che operino sull'attributo matricola in modalità non case-sensitive, nonché il metodo toString in modo da restituire una stringa nel formato come nell'esempio seguente: Studente{nome=Lucia, cognome=Bianchi, matricola=0612710134}. Infine la classe supporta il meccanismo di serializzazione degli

oggetti di Java; sul tipo è definita una relazione d'ordine naturale basata sull'ordine alfabetico crescente non case sensitive sul campo matricola.

Esame
La classe Esame consente di modellare l'entità omonima del dominio del problema caratterizzata dagli attributi nome e codice di tipo String, data di tipo LocalDate e voto di tipo intero. La classe mette a disposizione un metodo statico di creazione denominato crea che consente di istanziare un oggetto della classe esame (mediante invocazione del costruttore private); il metodo lancia le eccezioni DataNonValidaException nel caso di data non valida, VotoNonValidoException se non incluso nell'intervallo 18-31, e CodiceNonValidoException se il codice passato è uguale a null. La classe rende disponibili i metodi di accesso agli attributi in sola lettura. In particolare il metodo getData restituisce una deep copy dell'attributo (quindi non restituisce il reference dell'attributo, ma istanzia un nuovo oggetto con valore identico all'attributo e restituisce il reference di tale nuovo oggetto). Inoltre la classe sovrascrive i metodi equals ed hashcode in modo che siano coerenti e che operino sull'attributo codice in modalità non case-sensitive, nonché il metodo toString in modo da restituire una stringa nel formato come nell'esempio seguente: Esame{nome=Programmazione ad Oggetti, data=2017-01-05, voto=25, codice=0612700009}. Infine la classe supporta il meccanismo di serializzazione degli oggetti di Java.

Libretto
La classe Libretto consente di modellare l'entità omonima del dominio del problema. Nello specifico la classe Libretto è un HashSet di Esame in grado di supportare il meccanismo di serializzazione degli oggetti di Java ed ha un attributo di tipo Studente. Tale attributo può essere impostato solo tramite il costruttore e non è più modificabile. Il costruttore inizializza l'attributo creando una deep copy del parametro ad esso passato. La classe rende disponibile il metodo di accesso all'attributo in sola lettura restituendone una deep copy. Infine la classe sovrascrive il metodo toString in modo da restituire una stringa come nell'esempio seguente:
Studente{nome=Elena, cognome=Verdi, matricola=0612707654}
Esame{nome=Matematica II, data=2016-12-10, voto=25, codice=0612700007}
Esame{nome=Matematica I, data=2016-10-10, voto=21, codice=0612700005}
Esame{nome=Programmazione ad Oggetti, data=2017-01-10, voto=31, codice=0612700009}

ComparatoreMatricolaDescrescente
La classe implementa l'interfaccia funzionale Comparator<String> fornendo l'implementazione del metodo compare(String s1, String s2) in modo da restituire un valore intero negativo/positivo nel caso in cui s1 sia maggiore/minore di s2 rispetto all'ordinamento lessicografico non case sensitive, e 0 nel caso in cui le due stringhe siano uguali (sempre in modalità non case sensitive). La classe supporta il meccanismo di serializzazione degli oggetti di Java.

RegistroEsami
La classe RegistroEsami è una TreeMap<String, Libretto>. La classe supporta il meccanismo di serializzazione degli oggetti di Java ed implementa l'interfaccia Iterable<Libretto>. La classe sovrascrive il metodo toString in modo da restituire una stringa ottenuta per concatenazione della stringa "RegistroEsami\n" e delle stringhe ottenute invocando il metodo toString sulla collection degli oggetti Libretto presenti nella struttura dati. Gli elementi presenti nella classe RegistroEsami devono essere gestiti in maniera ordinata rispetto all'ordine indotto dalla classe ComparatoreMatricolaDescrescente (descritta in precedenza). La classe rende inoltre disponibile il metodo aggiungiEsame che associa nella struttura dati l'esame passato come

parametro allo studente la cui matricola è passata come parametro restituendo true in caso di successo; nel caso in cui invece non esista nella struttura dati uno studente identificato dal parametro matricola o esista già un'associazione dell'esame con la matricola passati come parametro, il metodo non altera la struttura dati e restituisce false.

RegistroEsamiUtils
La classe mette a disposizione i due metodi statici caricaDaFileBinario e salvaSuFileBinario che consentono rispettivamente di restituire un'istanza di RegistroEsami letta in maniera bufferizzata da un file binario il cui nome è passato come parametro al metodo; il metodo restituisce null nel caso in cui la lettura non vada a buon fine. Viceversa, il metodo salvaSuFileBinario salva su un file binario il cui nome è passato come parametro, gestendolo in maniera bufferizzata, l'istanza di RegistroEsami passata come parametro, restituendo true nel caso di successo e false nel caso di insuccesso. Entrambi i metodi sfruttano il meccanismo di serializzazione degli oggetti di Java.

SalvataggioAutomatico
La classe è un thread che nel proprio metodo run esegue continuamente un ciclo in cui provvede al salvataggio su file binario usando l'omonimo metodo della classe RegistroEsamiUtils dell'istanza di RegistroEsami riferita dall'attributo dello stesso tipo e sul file il cui nome è specificata dall'attributo della classe di tipo String. Entrambi gli attributi non sono modificabili e sono inizializzati dal costruttore. Durante il salvataggio del registro su file il thread deve operare in mutua esclusione sul registro. Il salvataggio deve essere effettuato ogni qualvolta al thread venga notificata una modifica avvenuta al registro. Il thread inoltre segnala su System.out il salvataggio avvenuto correttamente e su System.err un errore durante il salvataggio.

All'allievo sono fornite la classi DataNonValidaException, CodiceNonValidoException, VotoNonValidoException, nonché la classe TestRegistroEsami che consente di effettuare un semplice test delle classi scritte. Tale ultima classe può essere liberamente integrata dall'allievo in fase di test per verificare la correttezza del proprio codice, tuttavia deve essere garantito che il codice scritto dall'allievo funzioni correttamente con il file fornito.
Tutte le classi scritte dallo studente devono essere inserite nel package oop2016.gennaio2017.CognomeNome, dove CognomeNome devono essere sostituti dal cognome e dal nome effettivo dell'allievo.
L'allievo deve consegnare un file zip denominato CognomeNome.zip (CognomeNome devono essere sostituti dal cognome e dal nome effettivo dell'allievo) contenente tutte e sole le seguenti classi: Studente, Esame, Libretto, ComparatoreMatricolaDescrescente, RegistroEsami, RegistroEsamiUtils, SalvataggioAutomatico.