Dipartimento di Informatica Sistemistica e Comunicazione
Corso di Laurea in Informatica

Programmazione 2

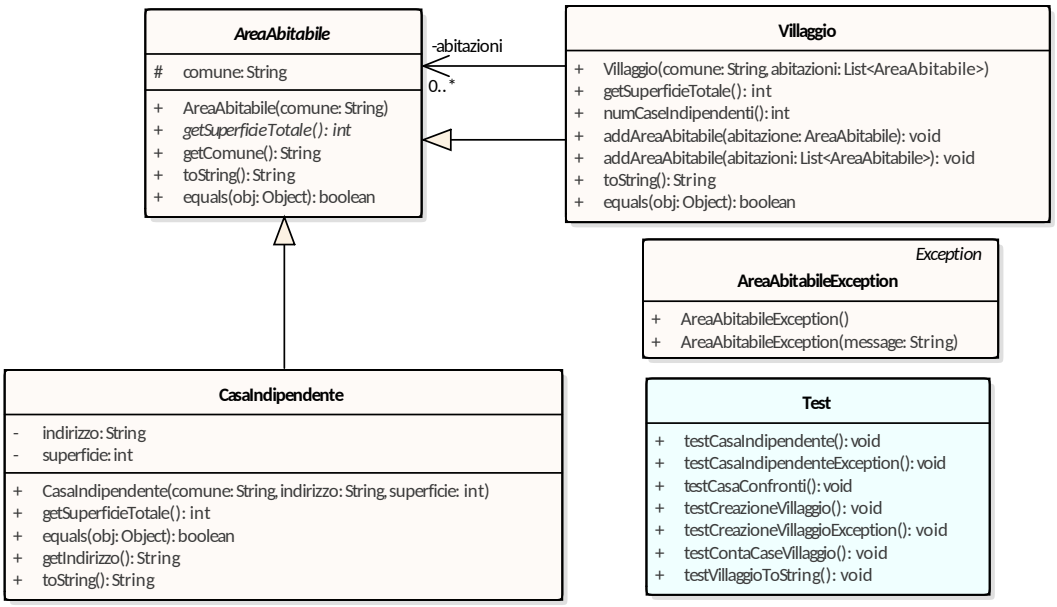
30 Giugno 2017 – Recupero secondo compitino

Testo parte di pratica

Si consideri un sistema di gestione urbanistica in cui un Villaggio è rappresentato come una lista di aree abitabili che a loro volta possono includere Case Indipendenti e altri Villaggi.

Implementare le classi come rappresentate dal seguente diagramma UML.

La classe `Test` (già fornita) contiene un insieme di casi di test che devono essere fatti girare di volta in volta in modo da verificare la corretta realizzazione del software. **Come requisito *minimo* per ottenere una valutazione positiva, lo studente deve garantire che la sua implementazione non presenti errori di compilazione e superi almeno 3 casi di test fra quelli dati.**



Classe `AreaAbitabileException`: rappresenta l'eccezione di tipo *checked* che viene prodotta quando gli oggetti del sistema sono utilizzati erroneamente.

Classe `AreaAbitabile`:

- Definisce il concetto astratto di un'area abitabile caratterizzata dal `comune` in cui l'area è situata. L'attributo si imposta con il costruttore ed è accessibile in sola lettura tramite il metodo `getComune`(). Se il `comune` in input è `null`, il costruttore solleva un'eccezione di tipo `AreaAbitabileException`.
- Il metodo *astratto* `getSuperficieTotale()` restituisce la superficie dell'area abitabile in metri quadri.
- Il metodo `toString` restituisce una stringa che descrive l'area abitabile indicandone il `comune` e la `superficie`.
- Il metodo `equals` restituisce `true` se le due aree hanno stesso `comune` e stessa `superficie`.

Classe `CasaIndipendente`:

- rappresenta una casa indipendente, caratterizzata, oltre che dal `comune` di ubicazione, dal proprio `indirizzo` e dalla propria `superficie` in metri quadri. Gli attributi sono accessibili in sola lettura, tramite i rispettivi metodi getter e si impostano con il costruttore. Se `comune` o `indirizzo` sono `null`, o se `superficie` non è un numero strettamente positivo, il costruttore solleva un'eccezione di tipo `AreaAbitabileException`.
- Due oggetti `CasaIndipendente` sono uguali se sono nel medesimo `comune`, hanno il medesimo `indirizzo` e la medesima `superficie`.

- Il metodo `toString` restituisce una stringa che descrive la casa indipendente indicandone il `comune`, la `superficie` e `l'indirizzo`.

Classe `Villaggio`:

- rappresenta un villaggio caratterizzato da una lista (`ArrayList`) di `AreaAbitabili` che ne fanno parte.
- Il costruttore riceve in input un `comune` e una lista di `AreaAbitabile` e crea un villaggio ubicato nel detto `comune` e che contiene le dette `AreaAbitabili`. Se il `comune` è `null`, o la lista è `null`o vuota, o una delle aree abitabili è ubicata in un `comune` diverso da quello del `Villaggio`, il costruttore solleva un'eccezione di tipo `AreaAbitabileException`.
- il metodo `addAreaAbitabile(abitazione)` aggiunge un'area abitabile al villaggio. Il metodo deve controllare che l'area abitabile sia diversa da `null` e che il suo `comune` sia coerente con il comune del villaggio, altrimenti solleva un'eccezione di tipo `AreaAbitabileException`.
- il metodo `addAreaAbitabile(abitazioni)`, in overloading al precedente, aggiunge al villaggio tutte le aree abitabili contenute nella lista `abitazioni`. Nel caso in cui la lista sia `null`, o se una delle aree abitabili nella lista non appartenesse al comune del villaggio, il metodo si interrompe e lancia un'eccezione di tipo `AreaAbitabileException`.
- il metodo `getSuperficieTotale` ritorna la superficie del villaggio calcolata come la somma della superficie delle aree abitabili che lo compongono.
- Il metodo `numCaseIndipendenti` ritorna il numero di `caseIndipendenti` contenute nel villaggio, contando anche le case indipendenti incluse all'interno dei villaggi che fanno parte del villaggio.
- il metodo `toString` ritorna una stringa riportante le informazioni su tutte le aree abitabili incluse nel villaggio.
- Il metodo `equals` controlla che il villaggio passato in input abbia lo stesso `comune` e che tutte le abitazioni in esso abbiano un corrispettivo uguale nella lista del villaggio (e viceversa). Quindi, le due liste di abitazioni devono avere lo stesso numero di elementi, tutti uguali a coppie (considerando l'ordine).