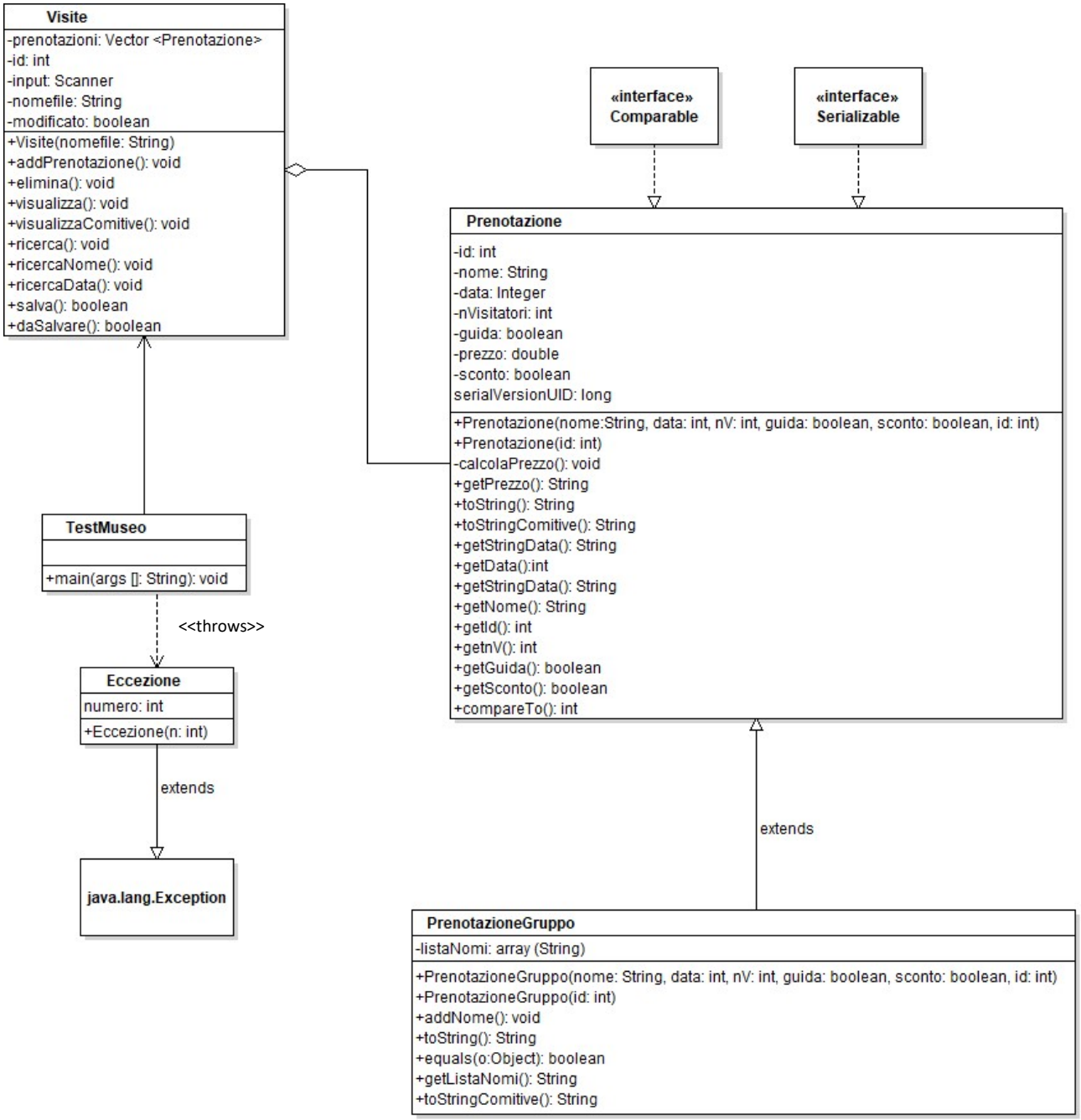
Bertoldo Sara 588361
Cuozzo Silvia 587958

Documentazione del progetto

Gestione prenotazione visite a un museo

Prima di entrare nel dettaglio, il seguente diagramma UML delle classi rappresenta il funzionamento del nostro programma.



Il *main* è contenuto nella classe *TestMuseo* e contiene un menu per far selezionare all'utente le azioni da eseguire. Il *main* contiene un oggetto di tipo *Visite* e salva in una variabile il nome del file da sovrascrivere in caso di salvataggio o creare in caso di primo salvataggio.

Visite è un vettore di oggetti di tipo *Prenotazione* o *PrenotazioneGruppo;* la scelta tra queste due classi è fatta in base al numero di visitatori e alla scelta di applicare o meno lo sconto comitiva. *PrenotazioneGruppo* estende *Prenotazione*, quindi eredita tutti gli attributi e i metodi della superclasse, a cui si aggiunge un array in cui sono salvati i nomi dei visitatori (i componenti della comitiva) ed i relativi metodi di gestione. Il motivo

dell'ereditarietà è per l'estrema similitudine dei due oggetti che si differenziano solo per quest'ultimo array di nomi, rendendo quindi vano il ripetersi di tutto il resto.

Nell'oggetto *Prenotazione*, oltre ai valori richiestoci di salvare, abbiamo creato un ID (numero identificativo) per ogni prenotazione. Questo valore di tipo *int* non è *static* perché nel momento in cui recuperiamo le prenotazioni da un file precedentemente creato, vogliamo che parta dal numero successivo all'ultima prenotazione creata e non da 0.
L'ID viene utilizzato per la rimozione della prenotazione, e per evitare situazioni di omonimia e/o stessa data di prenotazione.

Abbiamo creato una nostra eccezione (*Eccezione*) per gestire eventuali errori nella digitazione che non sarebbero stati intercettati dalle eccezioni di tipo come *InputMismatchException*.

Abbiamo usato varie volte *indexOf* invece di creare noi stesse una funzione di ricerca in quanto crediamo che sia più efficiente, questo però ci ha portate a implementare i metodi *Equals* e *CompareTo* e l'interfaccia *Comparable*, ma ci ha permesso di usare metodi della classe *Vector* come *Collections*.*sort* .

Infine, nella gestione della data avremmo preferito usare *int* al posto di altri tipi per poter agevolmente fare un confronto di tipo maggiore/minore. Salvando infatti la data come un numero di formato AAAAMMGG risulta matematicamente corretto dire che 20190803 è maggiore (successiva temporalmente) a 20180501. Per poter fare ciò i numeri sono stati presi in formato *int* e moltiplicati per potenze di 10 in base alla posizione da mantenere. Per questioni di conflitto di tipi però abbiamo dovuto salvare il formato in *Integer*.