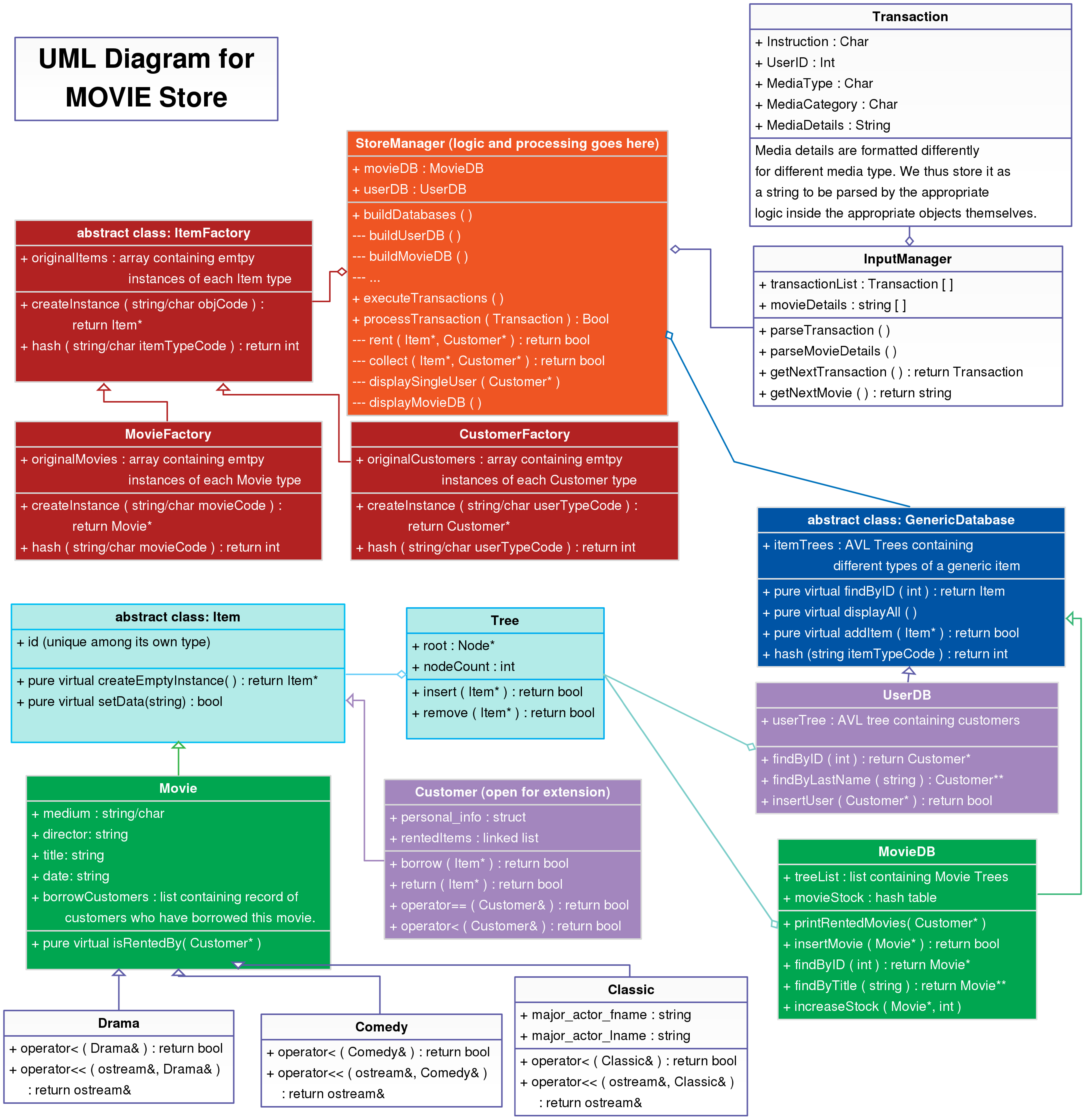
UML Diagram for MOVIE Store


Transaction
+ Instruction : Char
+ UserID : Int
+ MediaType : Char
+ MediaCategory : Char
+ MediaDetails : String
Media details are formatted differently for different media type. We thus store it as a string to be parsed by the appropriate logic inside the appropriate objects themselves.
StoreManager (logic and processing goes here)
+ movieDB : MovieDB
+ userDB : UserDB
+ buildDatabases ( )
--- buildUserDB ( )
--- buildMovieDB ( )
--- ...
+ executeTransactions ( )
+ processTransaction (Transaction) : Bool
--- rent (Item*, Customer*) : return bool
--- collect (Item*, Customer*) : return bool
--- displaySingleUser (Customer*)
--- displayMovieDB ( )
InputManager
+ transactionList : Transaction [ ]
+ movieDetails : string [ ]
+ parseTransaction ( )
+ parseMovieDetails ( )
+ getNextTransaction ( ) : return Transaction
+ getNextMovie ( ) : return string
abstract class: ItemFactory
+ originalItems : array containing emtpy instances of each Item type
+ createInstance (string/char objCode) : return Item*
+ hash (string/char itemTypeCode) : return int
MovieFactory
+ originalMovies : array containing emtpy instances of each Movie type
+ createInstance (string/char movieCode) : return Movie*
+ hash (string/char movieCode) : return int
CustomerFactory
+ originalCustomers : array containing emtpy instances of each Customer type
+ createInstance (string/char userTypeCode) : return Customer*
+ hash (string/char userTypeCode) : return int
abstract class: GenericDatabase
+ itemTrees : AVL Trees containing different types of a generic item
+ pure virtual findByID (int) : return Item
+ pure virtual displayAll ( )
+ pure virtual addItem (Item*) : return bool
+ hash (string itemTypeCode) : return int
abstract class: Item
+ id (unique among its own type)
+ pure virtual createEmptyInstance( ) : return Item*
+ pure virtual setData(string) : bool
Tree
+ root : Node*
+ nodeCount : int
+ insert (Item*) : return bool
+ remove (Item*) : return bool
UserDB
+ userTree : AVL tree containing customers
+ findByID (int) : return Customer*
+ findByLastName (string) : Customer**
+ insertUser (Customer*) : return bool
Movie
+ medium : string/char
+ director: string
+ title: string
+ date: string
+ borrowCustomers : list containing record of customers who have borrowed this movie.
+ pure virtual isRentedBy(Customer*)
Customer (open for extension)
+ personal_info : struct
+ rentedItems : linked list
+ borrow (Item*) : return bool
+ return (Item*) : return bool
+ operator== (Customer&) : return bool
+ operator&lt; (Customer&) : return bool
MovieDB
+ treeList : list containing Movie Trees
+ movieStock : hash table
+ printRentedMovies(Customer*)
+ insertMovie (Movie*) : return bool
+ findByID (int) : return Movie*
+ findByTitle (string) : return Movie**
+ increaseStock (Movie*, int)
Drama
+ operator&lt; (Drama&) : return bool
+ operator&lt;&lt; (ostream&, Drama&) : return ostream&
Comedy
+ operator&lt; (Comedy&) : return bool
+ operator&lt;&lt; (ostream&, Comedy&) : return ostream&
Classic
+ major_actor_fname : string
+ major_actor_lname : string
+ operator&lt; (Classic&) : return bool
+ operator&lt;&lt; (ostream&, Classic&) : return ostream&