### **Introduction**

This project demonstrates the use of Java **generics** to create a reusable and type-safe **library catalog system**. The application allows users to add, remove, and view different types of library items using a generic catalog class and a generic item class. It includes CLI interaction, error handling, and demonstrates compile-time type safety. The design is based on the principles outlined in the course readings and resources.

### **Key Concepts Used**

* **Generic Classes & Methods**: Used in AssetShelf<T> and for managing different asset types.
* **Type Safety**: Prevents type casting and runtime errors.
* **CLI Interface**: Built with Scanner for interactive user input.
* **Error Handling**: Proper warnings for removing non-existent items.
* **Code Reusability & Modularity**: Following object-oriented principles.

### **1. Source Code**

#### **LibraryAsset.java**

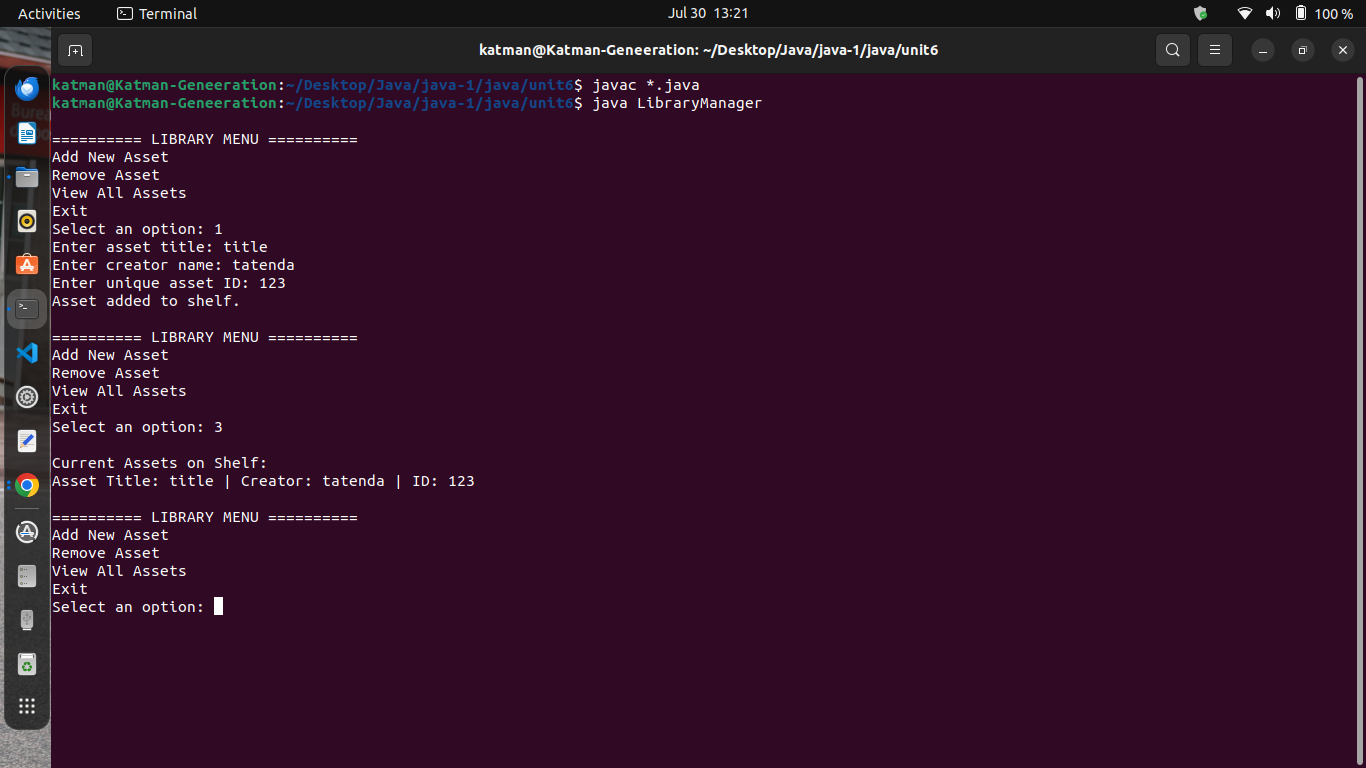
public class LibraryAsset {  
 private String assetTitle;  
 private String assetCreator;  
 private String assetID;  
public LibraryAsset(String assetTitle, String assetCreator, String assetID) {  
 this.assetTitle = assetTitle;  
 this.assetCreator = assetCreator;  
 this.assetID = assetID;  
 }  
  
 public String getAssetID() {  
 return assetID;  
 }  
  
 @Override  
 public String toString() {  
 return "Asset Title: " + assetTitle + " | Creator: " + assetCreator + " | ID: " + assetID;  
 }  
}

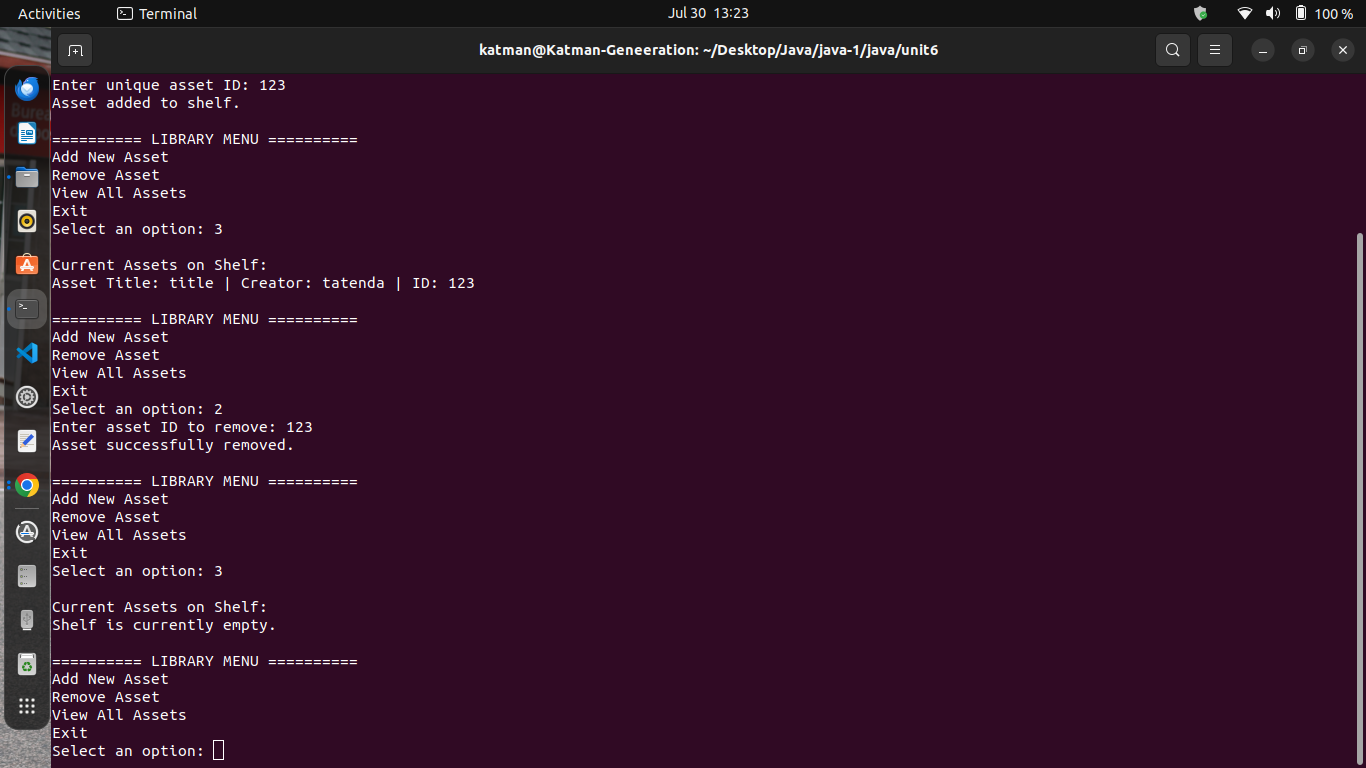
#### **AssetShelf.java**

import java.util.\*;  
  
public class AssetShelf<T extends LibraryAsset> {  
 private final Map<String, T> shelfMap;  
  
 public AssetShelf() {  
 shelfMap = new HashMap<>();  
 }  
  
 public void insertAsset(T newAsset) {  
 shelfMap.put(newAsset.getAssetID(), newAsset);  
 System.out.println("Asset added to shelf.");  
 }  
  
 public void deleteAsset(String assetCode) {  
 if (shelfMap.containsKey(assetCode)) {  
 shelfMap.remove(assetCode);  
 System.out.println(" Asset successfully removed.");  
 } else {  
 System.out.println(" Error: No asset found with ID: " + assetCode);  
 }  
 }  
  
 public void displayShelf() {  
 if (shelfMap.isEmpty()) {  
 System.out.println("Shelf is currently empty.");  
 } else {  
 for (T element : shelfMap.values()) {  
 System.out.println(element);  
 }  
 }  
 }  
}

#### **LibraryManager.java**

import java.util.Scanner;  
public class LibraryManager {  
 public static void main(String[] args) {  
 AssetShelf<LibraryAsset> digitalShelf = new AssetShelf<>();  
 Scanner inputScanner = new Scanner(System.in);  
 int menuChoice;  
  
 do {  
 System.out.println("\n========== LIBRARY MENU ==========");  
 System.out.println("Add New Asset");  
 System.out.println(" Remove Asset");  
 System.out.println("View All Assets");  
 System.out.println(" Exit");;  
 System.out.print("Select an option: ");  
 menuChoice = inputScanner.nextInt();  
 inputScanner.nextLine(); // Clear newline  
  
 switch (menuChoice) {  
 case 1:  
 System.out.print("Enter asset title: ");  
 String titleInput = inputScanner.nextLine();  
  
 System.out.print("Enter creator name: ");  
 String creatorInput = inputScanner.nextLine();  
  
 System.out.print("Enter unique asset ID: ");  
 String idInput = inputScanner.nextLine();  
  
 LibraryAsset asset = new LibraryAsset(titleInput, creatorInput, idInput);  
 digitalShelf.insertAsset(asset);  
 break;  
  
 case 2:  
 System.out.print("Enter asset ID to remove: ");  
 String idToRemove = inputScanner.nextLine();  
 digitalShelf.deleteAsset(idToRemove);  
 break;  
  
 case 3:  
 System.out.println("\nCurrent Assets on Shelf:");  
 digitalShelf.displayShelf();  
 break;  
  
 case 4:  
 System.out.println(" Exiting Library Catalog. Goodbye!");  
 break;  
  
 default:  
 System.out.println(" Invalid option. Please try again.");  
 }  
  
 } while (menuChoice != 4);  
  
 inputScanner.close();  
 }  
}  
 **Output Screenshot**





Example description:

This screenshot shows the interaction with the CLI:  
- Added book titled "Java Mastery"  
- Viewed catalog with the new entry  
- Removed the item successfully  
- Viewed catalog again showing it's empty

### **Conclusion**

This assignment successfully applies the principles of **generic programming in Java** to build a flexible library system. The use of type-safe generics, proper error handling, and user interaction via CLI demonstrates understanding of key concepts from Java SE 5.0 and beyond.