Assignment 6

- Java Generics
- Inheritance
- Class associations

1.Create a simple inventory management system using generics and inheritance. You need to implement a generic class `Inventory<T>` where T is a subtype of a base class Product. The Inventory class should manage a collection of products and provide the following functionalities:

1. `addProduct(T product)`: Adds a product to the inventory.
2. `removeProduct(T product)`: Removes a product from the inventory.
3. `listProducts()`: Lists all products in the inventory.

Create a base class Product with a name property, and two subclasses Electronics and Clothing, where Electronics has an additional warrantyPeriod property, and Clothing has an additional size property.

2. A farm simulator has an Animal class and classes specific to each animal; cow, goat, hen, turkey. Animal class should have functions to eat, makeSound. There is a Shelter class which supports the admission and release of specific animal instances. It should not support any other class other than the ones inherited from Animal. Making use of Generics write a Java program which can make the following main function work.

```
public class Main {
    public static void main(String[] args) {
        // This works because Cow extends Animal
        Shelter<Cow> cowShelter = new Shelter<>();
        cowShelter.admit(new Cow());

        // This works because Duck extends Animal
        Shelter<Duck> duckShelter = new Shelter<>();
        duckShelter.admit(new Duck());

        // Following code should fail
        // Shelter<String> stringShelter = new Shelter<>();
    }
}
```

3. Design and implement a generic class HierarchicalManager<T> where T is a subtype of a base class Entity. This class should manage a hierarchical collection of entities. The HierarchicalManager should support the following operations:
addEntity(T entity): Adds an entity to the collection.

removeEntity(T entity): Removes an entity from the collection.

getAllEntities(): Returns a list of all entities in the collection.

getEntitiesByType(Class<? extends T> type): Returns a list of entities that are instances of the specified subclass type.

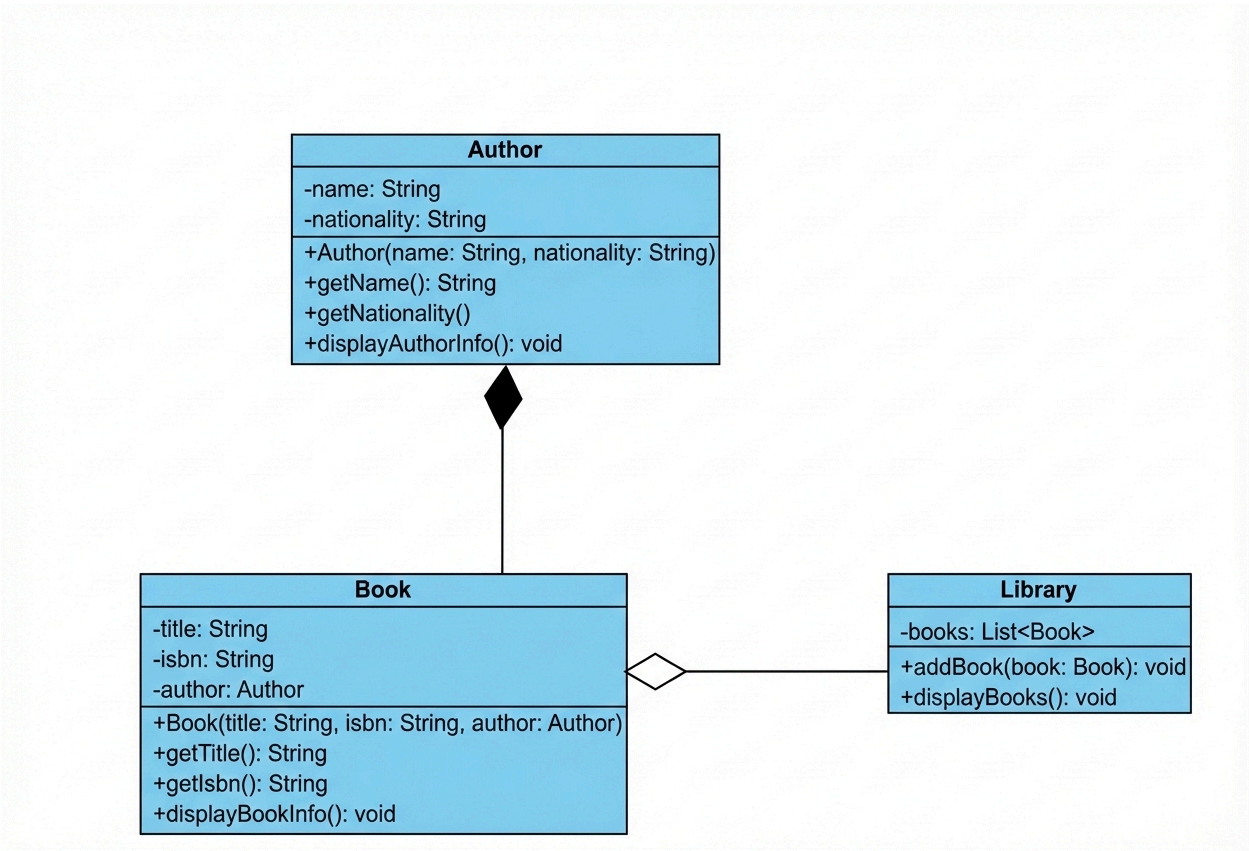
printHierarchy(): Prints out all entities and their type information.

Create a base class Entity and two subclasses Employee and Manager where Manager extends Employee. Implement the HierarchicalManager class to manage entities of these types.

4. Design a library management system that models the relationship between Library, Book, and Author.The system should demonstrate the following relationships:
An aggregation (has-a) relationship between the Library and Book classes, where the library contains a collection of books, but the books can exist independently of the library.

A composition (has-a) relationship between the Book and Author classes, where each book is associated with a specific author, and if the book is deleted, the associated author should also be deleted.

Class Diagram



5. Given the following multi-level class hierarchy in Java:

- Level 1: Vehicle (Base class)
- Level 2: Car (Inherits from Vehicle)
- Level 3: ElectricCar (Inherits from Car)
- Level 4: Tesla (Inherits from ElectricCar)

Each class has a method start() that is overridden in the derived classes, with the derived classes adding additional functionality. The super keyword is used to invoke the start() method of the parent class within each overridden method. The this keyword is used within constructors to initialize class-specific attributes.

1. Task 1: Implement this class hierarchy in Java. The Vehicle class should have a general start() method, which is then overridden by each subclass to add more specific functionality (e.g., starting a Car, an ElectricCar, and a Tesla).
2. Task 2: In the main() method, create an instance of the Tesla class and call the start() method.

6. Create a class to manage ordered pair of instances of any two data types. The first instance of the pair is called as key and the second as a value. It should getKey() and getValue() methods. Write a Java program incorporating the following main function.

```
  public static void main(String[] args) {

        // Your provided code:
        OrderedPair<String, Integer> p1 = new OrderedPair<String,
Integer>("Even", 8);
        OrderedPair<String, String>  p2 = new OrderedPair<String,
String>("hello", "world");
      }
```