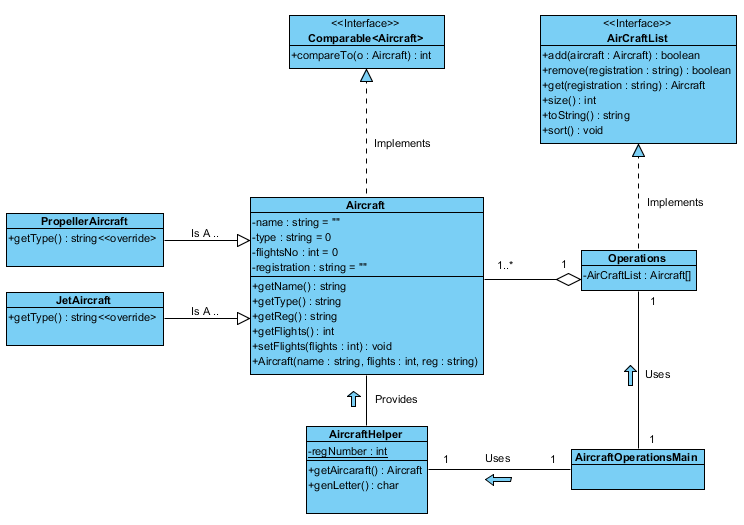
UML Design Class Diagram



Design Notes

**Design Patterns Used**

* **Creator pattern**. By creating the Aircraft helper and giving it the sole responsibility of creating specific concrete implementations of a common base type or interface.
* **Dependency inversion**. By using Aircraft-List as interface we are able to swap out different implementations without effecting the core Application (Aircraft-Operations-Main)
* **Expert**: Aircraft-Helper expert in aircraft details. Operations Expert in maintaining list of Aircraft.
* **Creator**: Aircraft-Helper is the Creator/ factory for serving up different types of aircraft
* **Low Coupling:** through the use of Base Type Aircraft, meaning there is low dependences between the new subtypes. I.e. Operations list depend on the low level details of each aircraft but the high level detail of just being an aircraft. Using interface or abstract class reduces coupling
* **High Cohesion:** with classes being responsible for specific task and only knowing what they need to about other types, you can also see from the diagram that related classes are kept close together. Classes have well defined roles.
* **Polymorphism**. With the use of the Base Type aircraft and the use of interfaces like Comparable<T> you can easily swap out the subtypes without effecting the core of the application i.e. operations does not need to change when using different aircraft

**Suggested improvements**

* Could be improved by using Enum for Aircraft type. Increases readability and helps when refactoring and adding new types
* Add ability to expose Array List stream property. This will allow cleaner code and the use of predicates and should be more efficient than looping through all items in the array.