**Assignment: Implement an Airport Baggage Handling System in Java**

**Objective: Create a baggage handling system for an airport to manage bags being checked in, loaded onto flights, and retrieved by passengers. This system should simulate real-time operations and handle concurrency with multithreading.**

**System Requirements:**

1. **Bag Class**
   * **Attributes:**
     + **bagId (int): Unique identifier for each bag.**
     + **ownerName (String): Name of the passenger who owns the bag.**
     + **weight (double): Weight of the bag in kilograms.**
     + **destination (String): Destination airport code (e.g., "JFK," "LHR").**
     + **status (String): Current status of the bag (e.g., "Checked In," "Loaded," "Retrieved").**
   * **Override Methods:**
     + **Override toString() to display bag details.**
     + **Override equals() and hashCode() for identifying bags by bagId.**
2. **Flight Class**
   * **Attributes:**
     + **flightId (int): Unique identifier for each flight.**
     + **destination (String): Destination airport code.**
     + **departureTime (String): Scheduled departure time.**
     + **loadedBags (List<Bag>): List of bags currently loaded onto the flight.**
   * **Methods:**
     + **addBag(Bag bag): Adds a bag to the flight's loaded bags. If the bag exceeds the weight limit or is already loaded, throw a custom exception.**
     + **removeBag(Bag bag): Allows removing a bag if needed (e.g., for retrieval).**
3. **AirportManager Class**
   * **Manages a collection of bags and flights, using HashMap<Integer, Bag> and HashMap<Integer, Flight>.**
   * **Core Operations:**
     + **Check-in Bag: Check in a new bag. If the bag bagId already exists, throw a BagAlreadyCheckedInException.**
     + **Load Bag onto Flight: Load a bag onto the appropriate flight by flightId. If the flight is full or the bag is not checked in, throw an UnableToLoadBagException.**
     + **Retrieve Bag: Allow a passenger to retrieve a checked-in bag. If the bag is not checked in or is already loaded onto a flight, throw a BagNotRetrievableException.**
     + **View Loaded Bags: Display all bags loaded on a particular flight.**
4. **Custom Exception Handling**
   * **Define custom exceptions:**
     + **BagAlreadyCheckedInException: Thrown when trying to check in a bag with an existing bagId.**
     + **UnableToLoadBagException: Thrown when a bag cannot be loaded onto a flight (e.g., if weight limits are exceeded).**
     + **BagNotRetrievableException: Thrown when attempting to retrieve a bag that’s already loaded or not checked in.**
5. **Multithreading**
   * **Implement separate Runnable classes for each core operation (CheckInBag, LoadBag, RetrieveBag, ViewLoadedBags).**
   * **Run each operation in its own thread. Use join() in the main thread to ensure completion of each operation before proceeding.**
6. **Synchronization**
   * **Synchronize access to shared resources, especially the loading of bags onto flights and retrieval operations, to avoid conflicts.**

**Bonus Tasks:**

1. **Update Bag Weight: Allow updating the weight of a bag before it’s loaded onto a flight.**
2. **Search by Destination: Enable viewing all bags destined for a specific airport.**
3. **Baggage Statistics: Maintain statistics on total weight per flight, number of bags handled per day, and most common destinations.**