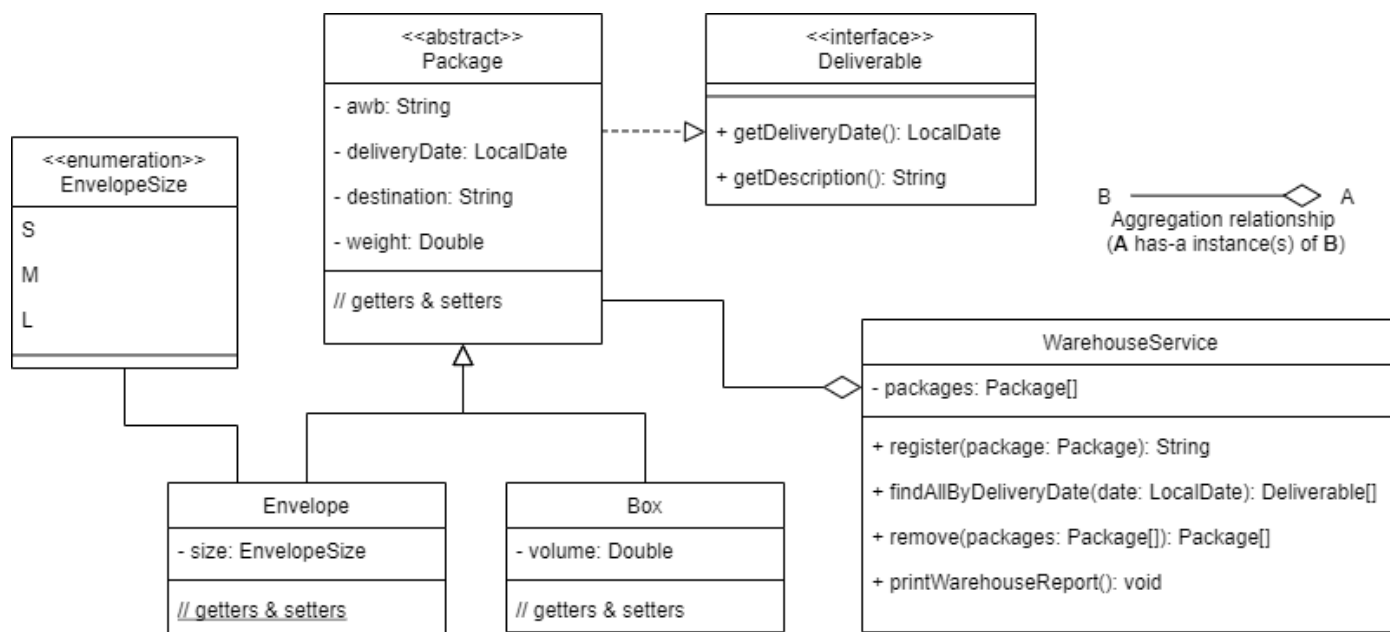


## A. Packages

The application you'll have to build today represents a management application to be used inside a warehouse for packages management. There will be two types of packages to work with: **Envelope** and **Box**. The functionalities to be built includes package registration, package removal, filtering packages by delivery date and printing a warehouse report.

Figure 1: UML diagram



## Requirements

1. Read and understand the UML diagram and then create the classes structure as presented in the diagram.
2. Implement the **WarehouseService** register & remove methods by taking into account the following guidelines:
  - The **register** method must assign a unique **awb** string to each package before saving it to the **packages**. The same string will be returned on successful processing. The **packages** must be always sorted ascending by **deliveryDate** and then by **destination** (if 2 packages have same **deliveryDate**);
  - If the package has an **awb** already set on registering then return the '**AWB already assigned**' message. If the package doesn't have a **deliveryDate** or **destination** set then return the '**Delivery date/Destination not set**' message;
  - If there is no space left to store another package in the **packages** array then resize the array before storing the package successfully;
  - The **remove** method must remove all the packages that are found (search by **awb**) from the **packages** class attribute;
  - The **remove** method must return an array of the packages that were not found to be removed or an empty array otherwise.
3. Create a class called **Main** where you should provide the **main** static method. Create an instance of **WarehouseService** class and then create some instances of the other classes as follows:
  - Create an instance of a **Box** and two instances of **Envelope**;
  - Register them by using the **WarehouseService** **register** method;
  - Use the **WarehouseService** **remove** method on one of the **Envelope** instances.
4. Implement the **findAllByDeliveryDate** method so that it returns an array of **Deliverable** objects that are to be delivered on a specific date. If the method parameter **date** is **null** then return the packages to be delivered today.

5. Implement the method `printWarehouseReport()` inside your `WarehouseService` class so that it displays the contents of the `packages` vector as shown in the next picture (or similar-meaning you might get a different format for the date for example). Order report ascending by delivery date and for every package display the `awb` and type (`BOX` or `ENVELOPE`) plus the extra info for each of the types. (use method overriding for this)

Figure 2: Warehouse report example

```
1 Packages report:
2 1. 12.08.2021 - 2 packages
3 AWB123 - BOX - volume 0.2 m^3
4 AWB234 - ENVELOPE - size S
5 2. 13.08.2021 - 1 package
6 AWB222 - BOX - volume 1 m^3
```