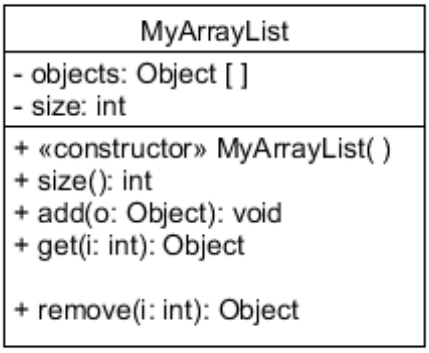
MyArrayList

In this exercise we take a closer look at **inheritance** with focus on **type casting**, the **Object class**, and **algorithms**. For this purpose, we implement a simple version of ArrayList, which helps us understand how Java works under the hood.

1.) Implement the class diagram.



- The **size** attribute stores the current size of the MyArrayList object.
- The **objects** attribute is an array of Object objects.
- The **size()** method returns the current size of the MyArrayList object.
- The **add(o: Object): void** method adds an object to the MyArrayList. Remember to expand the **objects** attribute if there is no space left. Following the STREAM development method it means that you need to add a private method that handles this. This method is not shown in the class diagram.
- The **get(i: int): Object** method returns the object that is on the given index **i**. You can assume that the passed argument is within range (0 and size – 1) – if it is not, the array of Objects will throw the appropriate exception.

2.) **Advanced exercise**: Implement **remove(i: int): Object**. This is a somewhat complex task, as the objects to the “right” (on indices > **i**) should be moved one index to the left (towards index 0). This exercise is for the experienced and/or skilled & ambitious students.

3.) To try the class, implement **JUnit test cases**! Test adding, removing, iterating over the MyArrayList object.

4.) Try this code (assuming, you have made a Vehicle and a car class as sepcified in the Vehicle exrercise)

```
Vehicle v = new Car(“Yellow”, “gas”);
MyArrayList mal = new MyArrayList();
mal.add(v);
```

What types may I receive or cast to from **mal** when I execute

```
mal.get(0);
```