In this task it is your job to implement a `Taxi` class that represents taxis, which have a driver and seats for four passengers. The goal of this task is to learn how classes and methods in Java work and how to use them.

Follow the steps below to solve this task.

Step 1: The basic `Human` class

Every passenger in the taxi will be represented by an object of the `Human` class.

Create a basic `Human` class according to the following description: A `Human` has the `String` attributes `forename` and `name` (the surname). The constructor of `Human` has two `String` parameters, `forename` and `name`, which will be used to initialize the attributes of the same name. Furthermore, every `Human` can be asked for his `name` using `getName()` or his `forename` using `getForename()`.

Step 2: Implement `Human.toString()`

Every class in Java is derived from the basic class `java.lang.Object` (means: class `Object` in the package `java.lang`), which implements a method called `toString()`. This method provides the `String` representation of a specific object of the class. It is often used implicitly in Java like here:

```
Human juergen = new Human("Juergen", "Staub");
System.out.println("This is " + juergen + ".");
```

With the default `toString()` method inherited from `java.lang.Object`, that code snippet outputs something like:

```
This is Human@15fea60.
```

This means that `toString()` returned the string `Human@15fea60`. Of course, this isn't really useful. It would be better if it returned a concatenation of `forename` and `name`, separated by a space character.

It is your task to override the method `toString()` in your `Human` class such that the above code snippet results in the following output:

```
This is Juergen Staub.
```

Step 3: The basic `Taxi` class

Create a class called `Taxi` which represents a taxi. Because every taxi has a driver, every `Taxi` object should have a `driver` attribute of type `Human`. Additionally, the `Taxi` can have up to four passengers which should be stored in an attribute `passengers` of type `Human[]` (i.e., an array of `Human`s).

When a `Taxi` is created using its constructor, a driver must be assigned to it. Hence, the class `Taxi` needs a constructor with one parameter of type `Human`. It must be possible to get the full name of the taxi driver by calling `getDriverName()` on a `Taxi` object. It should return a `String` such as `Juergen Staub`.

Step 4: Implement `Taxi.add(…)`

Each taxi can transport up to four passengers plus its driver. Humans get into the taxi separately. The `Taxi` class has the method `add(Human passenger)` to add a new passenger. If the taxi is full, the method doesn't add the passenger. Instead, the following message must be printed to the console:

```
We are sorry, Lucky Fuke. The taxi is full.
```

If otherwise the taxi wasn't full, the passenger is added and another message is printed:

```
Lucky Fuke gets in.
```

(Of course, the name of the respective person should be used.)

Your task is to implement all of this.

Step 5: Implement `Taxi.toString()`

When calling the `toString()` method on a `Taxi` object, the status of that `Taxi` should be returned. Assuming the taxi is driven by Juergen Staub and he has the guests Andrea Bora, Franzi Ada, Leah Posh and Susi Fresh, one of the following messages should be returned:

If the taxi is empty:

```
This is the taxi of Juergen Staub. He takes nobody along.
```

If Andrea Bora is the only passenger:

```
This is the taxi of Juergen Staub. He takes Andrea Bora along.
```

If all four guests are in the taxi:

```
This is the taxi of Juergen Staub. He takes Andrea Bora, Franzi Ada, Leah Posh and
Susi Fresh along.
```

Take care of the correct construction of the passenger list.

Step 6: Implement `Taxi.allGetOut()`

Although passengers get in separately, they can only get out all together. This happens when the method `allGetOut()` is called on a `Taxi` object. After that, the taxi is empty again except for the driver. Furthermore, `Taxi.allGetOut()` returns an array (type `Human[]`) with the former passengers or an empty array if there were no passengers in the taxi. The returned array must not contain `null` values!

Step 7: Write a `main(…)` method (optional)

This task is optional but you can use the following code to try out your implementation of the classes `Human` and `Taxi`. Create a new class, for example `MyTaxiTest`, with only one method:

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


}
```

Insert the following code into the method body:

```
Human juergen = new Human("Juergen", "Staub");


Human andrea = new Human("Andrea", "Bora");
Human franzi = new Human("Franzi", "Ada");
Human leah = new Human("Leah", "Posh");
Human susi = new Human("Susi", "Fresh");
Human lucky = new Human("Lucky", "Fuke");


Taxi taxi = new Taxi(juergen);


taxi.add(andrea);
taxi.add(franzi);
taxi.add(leah);
taxi.add(susi);
taxi.add(lucky);


System.out.println(taxi);
```

If you have done everything right, there should be no compile errors and you should get the following output:

```
Andrea Bora gets in.
Franzi Ada gets in.
Leah Posh gets in.
Susi Fresh gets in.
We are sorry, Lucky Fuke. The taxi is full.
This is the taxi of Juergen Staub. He takes Andrea Bora, Franzi Ada, Leah Posh and
Susi Fresh along.
```