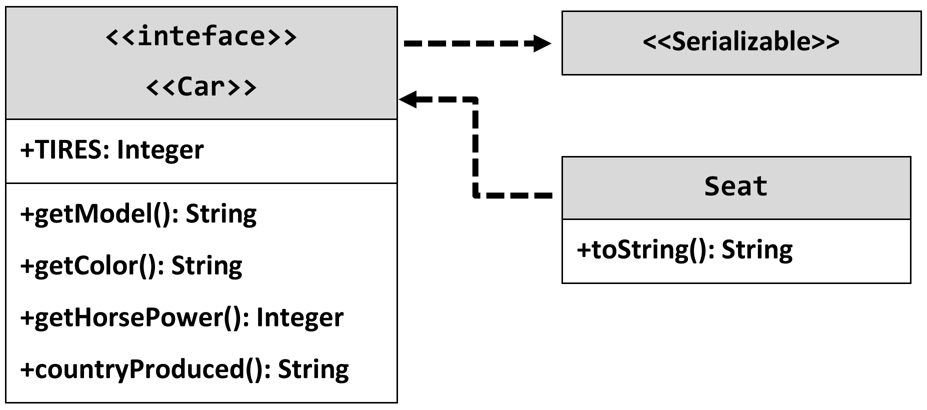
Lab: Interfaces and Abstraction

This document defines the lab for the ["Java Advanced" course @ Software University](). Please submit your solutions (source code) to all below-described problems in [Judge]().

1. Car Shop

Build hierarchy from **classes** and **interfaces** for this UML diagram:



Your hierarchy has to be used with this code.

Main.java

```
public static void main(String[] args) {
    Car seat = new Seat("Leon", "gray", 110, "Spain");

    System.out.println(String.format(
            "%s is %s color and have %s horse power",
            seat.getModel(),
            seat.getColor(),
            seat.getHorsePower()));
    System.out.println(seat.toString());
}
```

Examples

Input	Output
	Leon is gray color and have 110 horse power This is Leon produced in Spain and have 4 tires

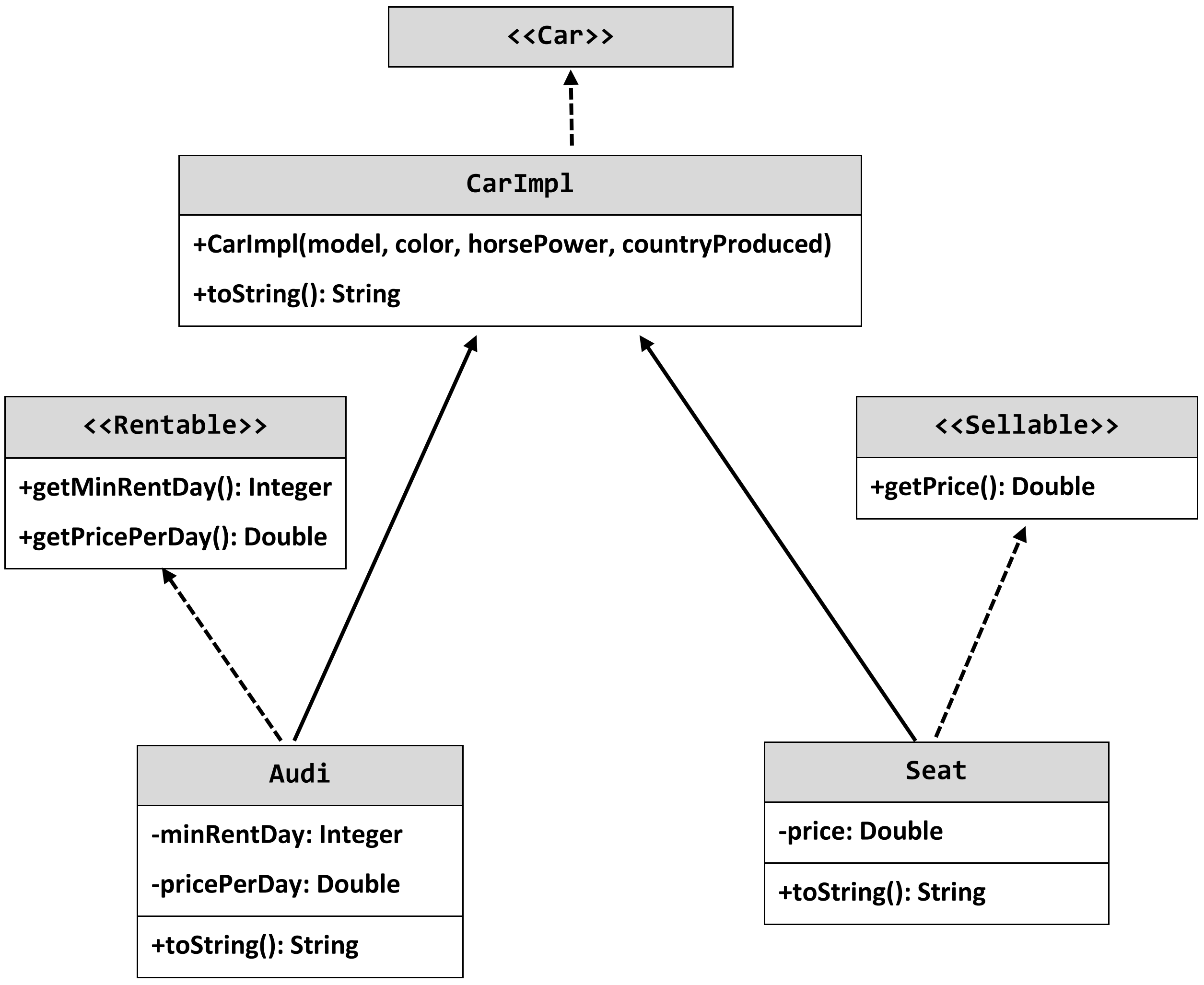
Solution

```
public interface Car {
    int TIRES = 4;

    String getModel();

    String getColor();

    Integer getHorsePower();
}
```

Note: consider using the wrapper classes in the **Seat** constructor.

2. Car Shop Extend

Extend the previous problem:



Your hierarchy has to be used with this code:

Main.java

```
public static void main(String[] args) {
    Sellable seat = new Seat("Leon", "Gray", 110, "Spain", 11111.1);
    Rentable audi = new Audi("A4", "Gray", 110, "Germany", 3, 99.9);

    printCarInfo(seat);
    printCarInfo(audi);
}

private static void printCarInfo(Car car) {
    System.out.println(String.format(
            "%s is %s color and have %s horse power",
            car.getModel(),
            car.getColor(),
            car.getHorsePower()));
    System.out.println(car.toString());
}
```

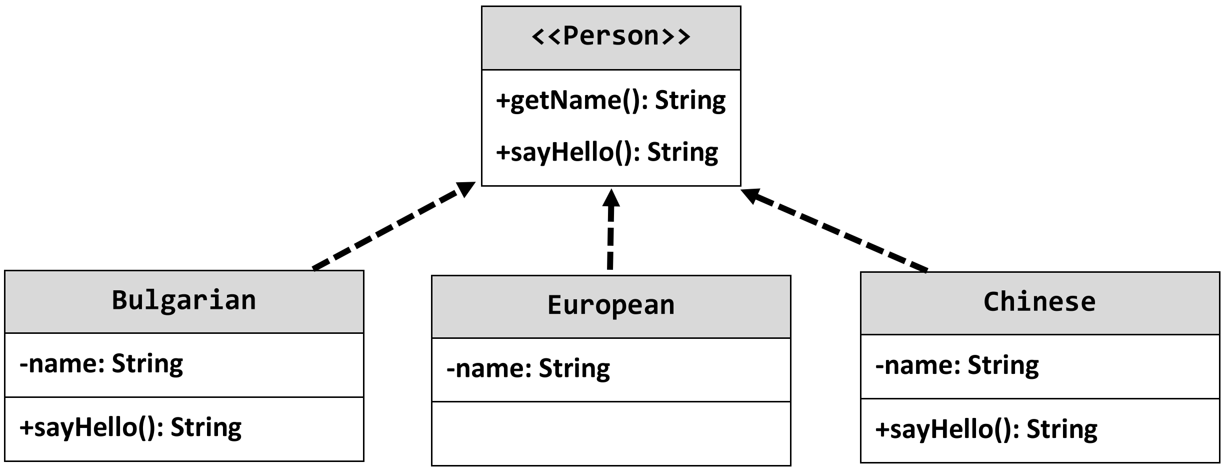
Examples

Input	Output

	`Leon is Gray color and have 110 horse power`
	`This is Leon produced in Spain and have 4 tires`
	`Leon sells for 11111,100000`
	`A4 is Gray color and have 110 horse power`
	`This is A4 produced in Germany and have 4 tires`
	`Minimum rental period of 3 days. Price per day 99,900000`

3. Say Hello

Build hierarchy from classes and interfaces for this **UML** diagram:



Your hierarchy has to be used with this code:

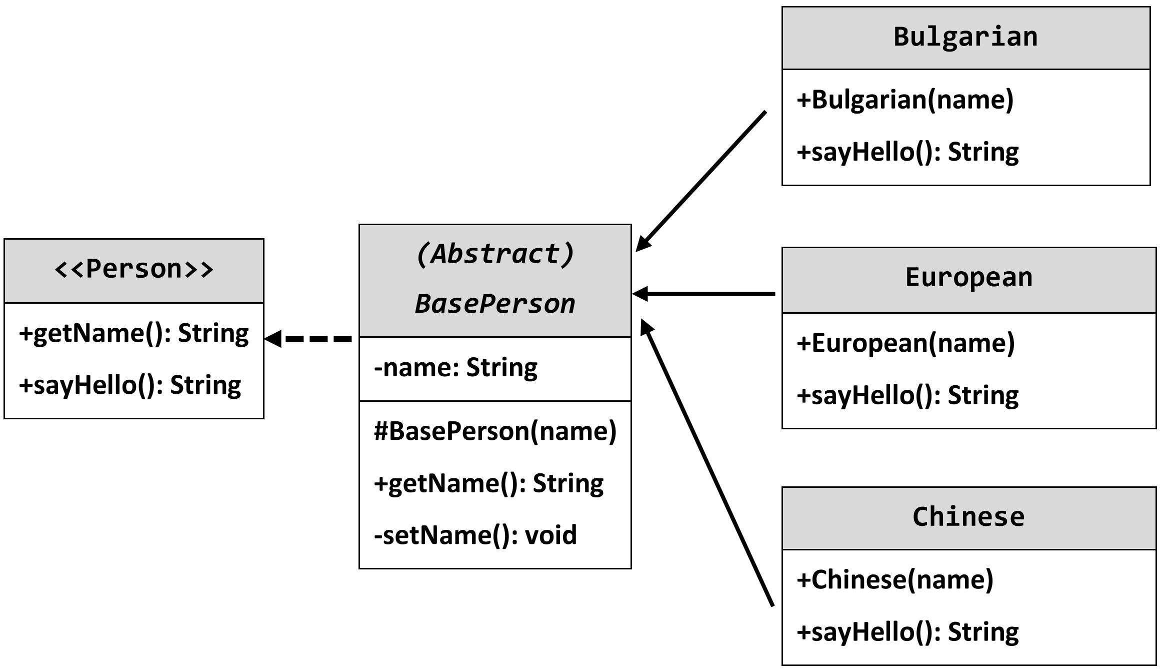
Main.java
```
public static void main(String[] args) {
    List<Person> persons = new ArrayList<>();


    persons.add(new Bulgarian("Peter"));
    persons.add(new European("Peter"));
    persons.add(new Chinese("Peter"));

    for (Person person : persons) {
        print(person);
    }
}

private static void print(Person person) {
    System.out.println(person.sayHello());
}
```

Examples

Input	Output
	Здравей Hello Djydjybydjy

4. Say Hello Extend

Build hierarchy from classes and interfaces for this **UML** diagram



Your hierarchy has to be used with this code:

Main.java

```java
public static void main(String[] args) {
    List<Person> persons = new ArrayList<>();


    persons.add(new Bulgarian("Peter"));
    persons.add(new European("Peter"));
    persons.add(new Chinese("Peter"));

    for (Person person : persons) {
        print(person);
    }
}

private static void print(Person person) {
    System.out.println(person.sayHello());
}
```

Examples

Input	Output
	Здравей Hello Djydjybydjy

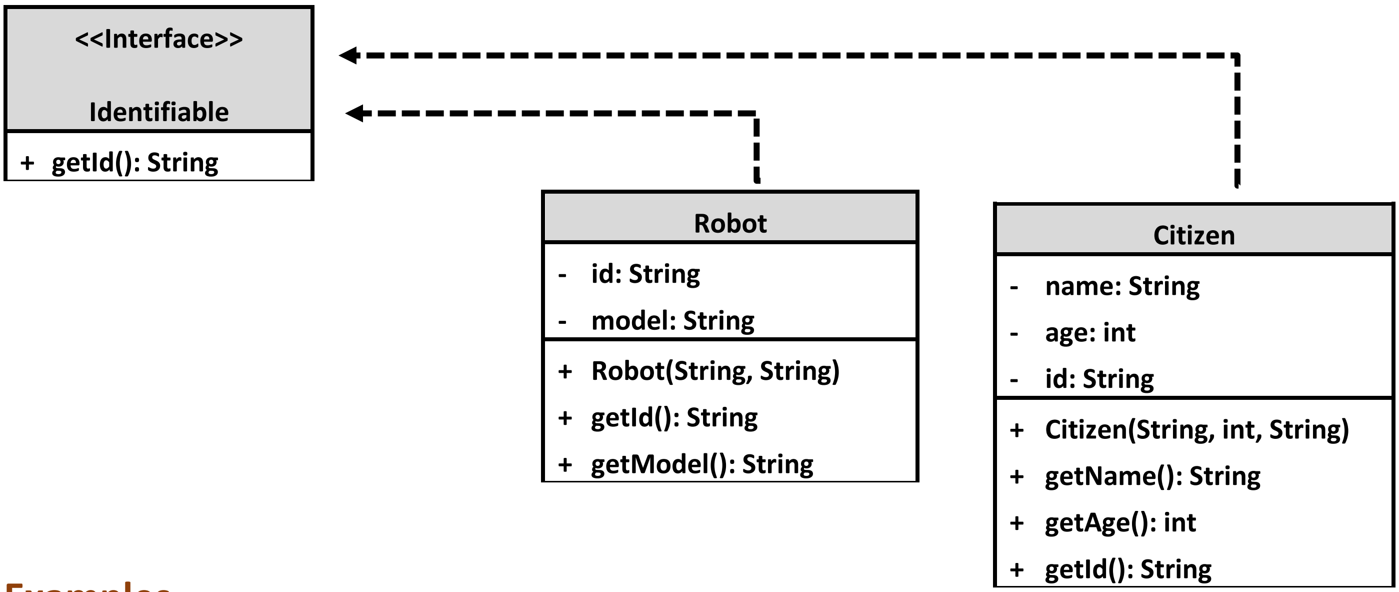
5. Border Control

It's the future, you're the ruler of a totalitarian dystopian society inhabited by **citizens** and **robots**, since you're afraid of rebellions you decide to implement strict control of who enters your city. Your soldiers check the **Id**s of everyone who enters and leaves.

You will receive from the console an **unknown** amount of lines until the command "**End**" is received, on each line, there will be the information for either **a citizen** or **a robot** who tries to enter your city in the format "**`{name} {age} {id}`**" for citizens and "**`{model} {id}`**" for robots.

After the end command on the next line, you will receive a single number representing **the last digits of fake ids**, all citizens or robots whose **Id** ends with the specified digits must be detained.

The output of your program should consist of all detained **Id**s each on a separate line (the order of printing doesn't matter).



Examples

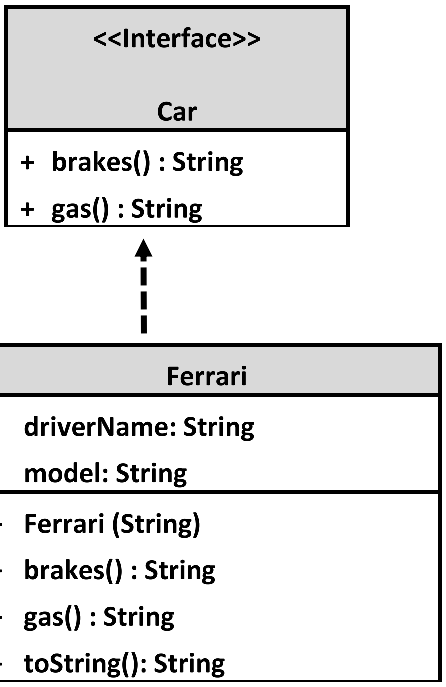
Input	Output
Peter 22 9010101122 MK-13 558833251 MK-12 33283122 End 122	9010101122 33283122
Teo 31 7801211340 Anna 29 8007181534	7801211340

IV-228 999999 Simon 54 3401018380 KKK-666 80808080 End 340	

6. Ferrari

Model an application that contains a **class Ferrari** and an **interface**. Your task is simple, you have a **car - Ferrari**, its model is "**`488-Spider`**" and it has a **driver**. Your Ferrari should have the functionality to **use brakes** and **push the gas pedal**. When the **brakes** are pushed down print "**`Brakes!`**", and when the **gas pedal** is pushed down - "**`brum-brum-brum-brrrrr`**". As you may have guessed this functionality is typical for all cars, so you should **implement an interface** to describe it.

Your task is to **create a Ferrari** and **set the driver's name** to the passed one in the input. After that, print the info. Take a look at the Examples to understand the task better.



Input

On the **single input line**, you will be given the **driver's name**.

Output

On the **single output line**, print the model, the messages from the brakes and gas pedal methods, and the driver's name. In the following format:

"**`{model}/{brakes}/{gas}/{driver's name}`**"

Constraints

The input will always be valid, no need to check it explicitly! The Driver's name may contain any ASCII characters.

Example

Input	Output
`Dominic Toretto`	`488-Spider/Brakes!/brum-brum-brum-brrrrr/Dominic Toretto`
`Brian O'Conner`	`488-Spider/Brakes!/brum-brum-brum-brrrrr/Brian O'Conner`