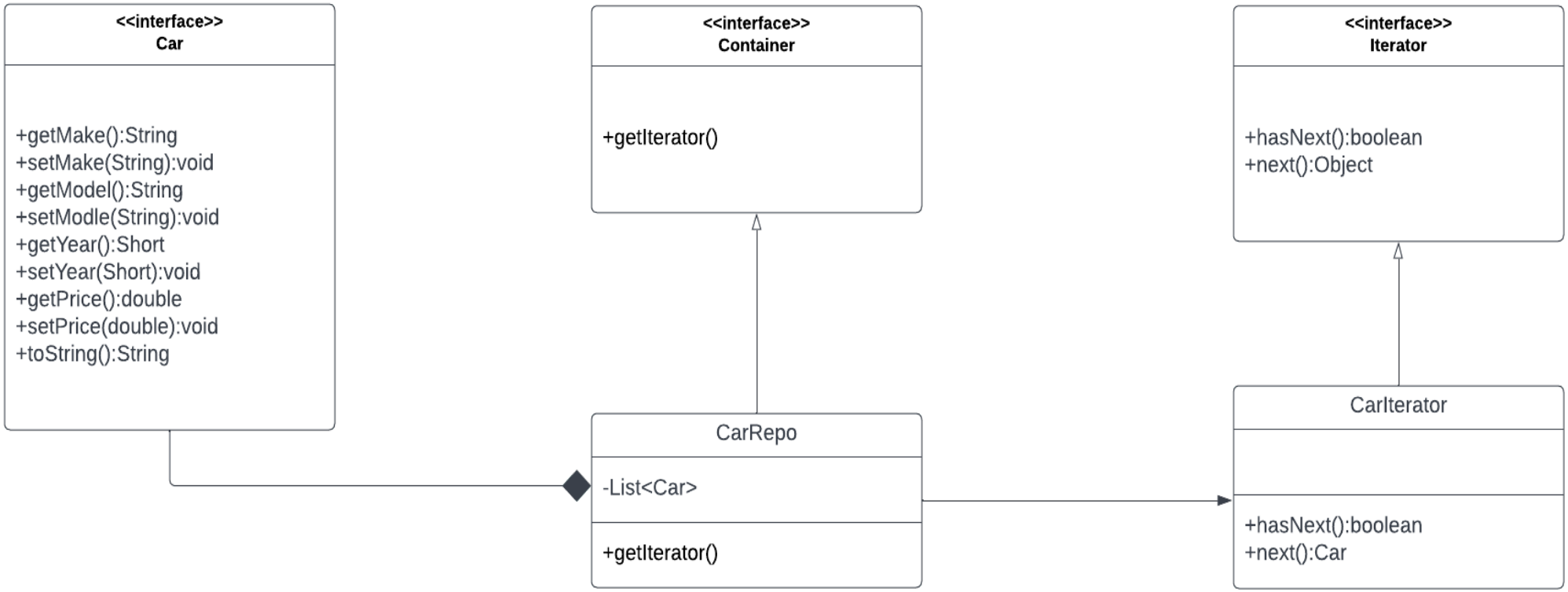
Used Car Dealership
Iterator Pattern

By Jarosław Rybak

Iterator Pattern UML Diargram



Iterator Pattern Features

- No matter what kind of car it is, as long that it is a car, it will be output from the next() method.
- In this example it is using any class that implements the List interface.


```
2
3  public interface Iterator
4  {
5      public boolean hasNext();
6      public Object next();
7  }
```

```
2
3  public interface Container
4  {
5      public Iterator getIerator();
6  }
7
```

```
public interface Car
{
    public String getMake();
    public void setMake(String make);
    public String getModel();
    public void setModel(String model);
    public short getYear();
    public void setYear(short year);
    public double getPrice();
    public void setPrice(double price);
    public String toString();
}
```

All the is relavent to the user

- Only 3 parts are relavent to the user.
- 1: getIterator() to make an instance.
- 2: hasNext() to check for the end of the list.
- 3: next() to output the current car be ready for the next car.

```
@Override
public Iterator getIerator()
{
    return new CarIterator();
}

private class CarIterator implements Iterator
{
    int index;

    @Override
    public boolean hasNext()
    {
        if(index < Cars.size())
        {
            return true;
        }
        return false;
    }

    @Override
    public Car next()
    {
        if(this.hasNext()){
            return Cars.get(index++);
         }
        return null;
    }
}
```

Component Testing

Make 2 different instances of the Iterator. Use them separately for different purposes without one interfering with the other. Iterator 1 will filter Cars from “Make”: “Beta” and return them. Iterator 2 will return Cars in ascending order of year. Neither will know of the other nor have an effect on the others output.