Introduction to Java

Special Care Set for Java Begineers/ Ex 05

・Notice!
The motivation of this page is to lead regular exercises.
We prepare many useful information in this page as hints of exercises, but this page has no answer.
If you have question, please don't hesitate to ask and discuss your teacher and teaching assistants.
このページは、通常の演習問題への導きとして作られています。
多くの有用な情報をヒントとして準備しておりますが、ここに回答が隠されているとかはありません。
もし、演習に対して質問等ございましたら、遠慮なく教員・TAに質問・議論をしてください。

Example Problem: クラスの再利用 - 色々な車のクラスを作る -
Class reutilization - How to construct variety "Car" class -

Special Care SetのEx 3では、Carクラスの作成を行いました。
もう一度、前回のCarクラスをおさらいしてみましょう。
In ex.3 of special care set, we construct a Car class.
Let's review previous Car class in here.

```
/* Carクラスの完成型 */
class Car{
    protected int speed;
    protected double fuel;
    protected double consumption;
    protected double mileage;

    Car(double consumption, double fuel){
        this.consumption = consumption;
        this.fuel = fuel;
        this.mileage = 0;
    }

public void speedUp(){
    if(speed < 100) this.speed += 10;
}

public void speedDown(){
    if(speed > 0) this.speed -= 10;
}

public void driveCar(int minute){
    double dist = speed * (double)minute / 60.0;
    if(this.fuel < dist / this.consumption){
        this.mileage += fuel * consumption;
        this.fuel = 0;
    }
    else{
        this.mileage += dist;
        this.fuel -= dist / this.consumption;
    }
}

public void putFuel(double fuel){
    this.fuel += fuel;
}

    public double getSpeed(){
        return this.speed;
    }

    public double getMileage(){
        return this.mileage;
    }

    public double getFuel(){
        return this.fuel;
    }

    public void printState(){
        System.out.println("Speed: " + this.speed + "km/h, Traveled: " + this.mileage + "km, Fuel left: " + this.fuel +"L.");
    }
}
```

このクラスでは、内部状態として、速度（speed）、燃料（fuel）、燃費（consumption）、走行距離（mileage）を持っています。これからの継承のために、前回はprivateを使いましたが、今回はprotectedにしています。

また、外部から使用可能なメソッドとして、speedUp, speedDown, driveCar, putFuel, getSpeed, getmileage, getFuel, printStateというメソッドを作り、使えるようにしました。
This class have speed, fuel, consumption and mileage for internal state, also this class is able to use methods as:
speedUp, speedDown, driveCar, putFuel, get Speed, getmileage, getFuel and printState.

1. 車にもいろいろな種類があります 〜オートマチック車とマニュアル車〜

さて、皆さんは運転免許を取りましたでしょうか？または運転免許を取りたいと思っておりますでしょうか？
日本の運転免許にはいろいろとありますが、みなさんが運転免許を取得するとなると、基本は『中型普通自動車免許』になると思います。
しかしながら、中型免許を取ろうと自動車教習所に行くと、何故かいつも2つ（以上）のコースがあることに気が付きます。そうですね。オートマチック（AT）車と、マニュアル（MT)車があるからですね。
運転免許には、AT限定というちょっと安い運転免許があります。AT限定は、ギアを自分で変更可能なMT車に乗れない代わりに、多少安く、割と簡単に運転免許を取得できます。
ということで、車の種類として、MT車とAT車を作ってみたいと思います。
Well, did you get driver's license or do you have a plan to get it now?

もし普通にMT車を作るとするなら、こんな感じになるでしょう。
If you create MT car, it wil be consist as follow.

```
/* ManualCarクラス */
class ManualCar{
    private int speed;
    private double fuel;
    private double consumption;
    private double mileage;
    private int gear;
    private final int maxGear;

    ManualCar(double consumption, double fuel, int maxGear){
        this.consumption = consumption;
        this.fuel = fuel;
        this.mileage = 0;
        this.gear = 1;
        this.maxGear = maxGear;
    }

    public void speedUp(){
        this.speed += gear * 5;
        if(speed > 160) this.speed = 160;
    }

    public void speedDown(){
        this.speed -= 10;
        if(speed < 0) this.speed = 0;
    }

    public void driveCar(int minute){
        double dist = speed * (double)minute / 60.0;
        if(this.fuel < dist / this.consumption){
            this.mileage += fuel * consumption;
            this.fuel = 0;
        }
        else{
            this.mileage += dist;
            this.fuel -= dist / this.consumption;
        }
    }

    public void putFuel(double fuel){
        this.fuel += fuel;
    }

    public double getSpeed(){
        return this.speed;
    }

    public double getMileage(){
        return this.mileage;
    }

    public double getFuel(){
        return this.fuel;
    }

    public void gearUp(){
        if(this.gear < this.maxGear) this.gear++;
    }

    public void gearDown(){
        if(this.gear > 1) this.gear --;
    }
```

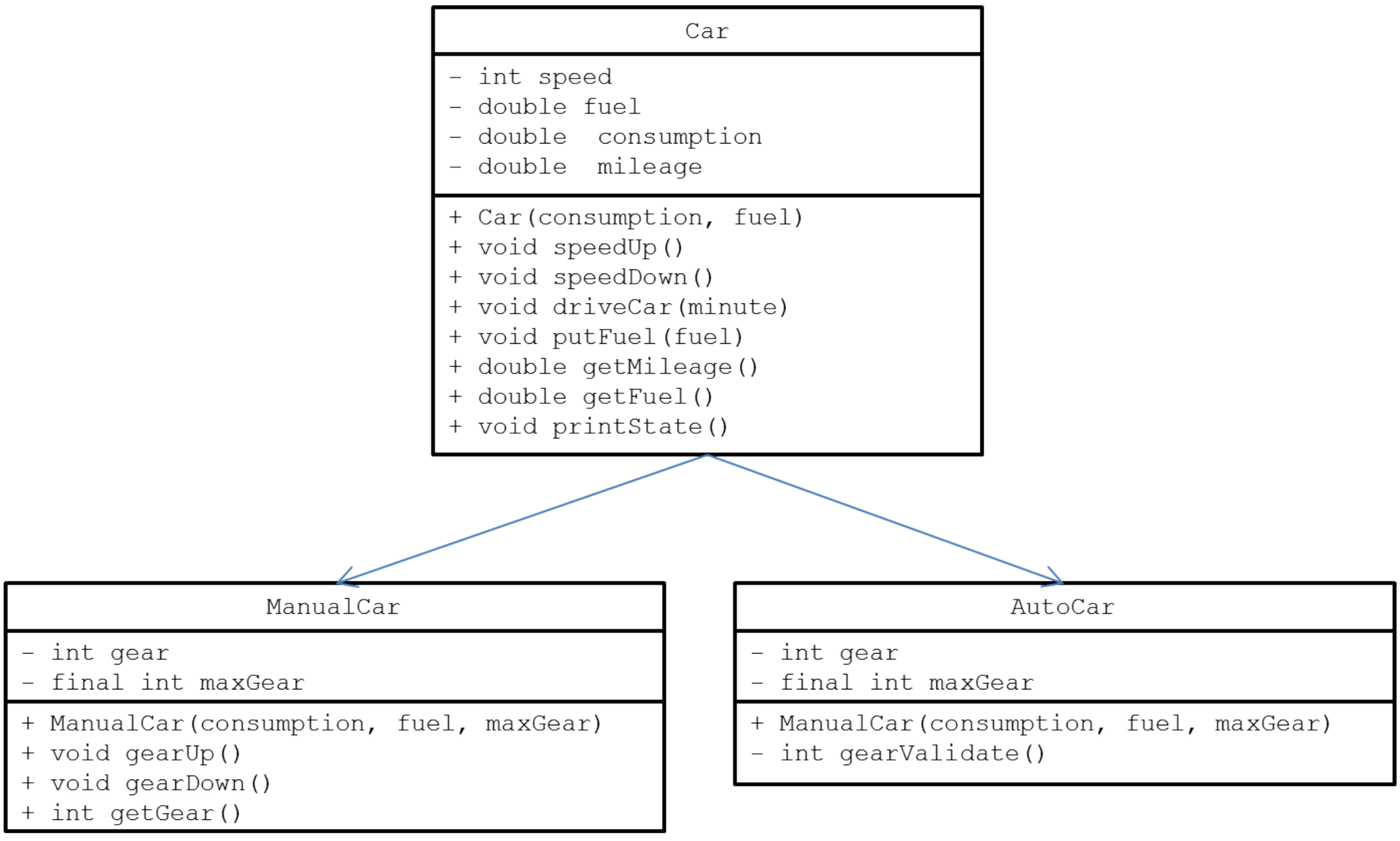
```
    public void printState(){
        System.out.println("Speed: " + this.speed + "km/h, Traveled: " + this.mileage + "km, Fuel left: " + this.fuel +"L.");
    }
}
```

と、このように、Carで作ったはずのメソッドや変数を全て再定義しなおして作らなければならず、非常に面倒ですね。AT車を作ろうとしても、また行数が掛かるので、今回はちょっと置いておきましょう。
In this way, it is very complecated to redefine all variables and methods from Car class. Creating AT car may be same way.

2. 一度作ったクラスを再利用する - クラスの継承を用いて

MT車もAT車も車であることには変わらず、アクセルやブレーキは必ず必要なものです。そういった共通の部分は既にCarクラスで作っておいていました。
MTとATの違いは何かというと、ギアを自分で変更できるか、そうでないかの差です。
ですから、MTには gearUp や、 gearDown というメソッドが必要なのに対して、AT車にはそれが必要ありません。その代わり、速度が上がるにつれて、ギアが変更されるようなメソッド gearValidateが必要になってくるでしょう。

図にしてみると、こんな感じになります。
This figure illustrates class inheritance between Car, ManualCar and AutoCar classes.



ギア自身は、速度の伸びだけではなく、トルク（回転力）にも影響しますから、厳密に定義しようとするといろんなものを追加しないといけないですが、
今回は単純に速度の伸び方だけを考えることにしておきましょう
ちなみにですが、多少書き方が違いますが、このような図を『クラス図』と言います。

3. 以前のCarクラスを継承したManualCarクラス

それでは、今回勉強した『継承』を使って、CarクラスからManualCarクラスを作成してみましょう。
Now, let's implement ManualCar via Car class using "Class Inheritance."

```
/* ManualCarクラス - 継承Ver. */
class ManualCar extends Car{


    private int gear;
    private final int maxGear;

    ManualCar(double consumption, double fuel, int maxGear){
        super(consumption, fuel);

        this.gear = 1;
        this.maxGear = maxGear;
    }
```

```
    public void speedUp(){
        this.speed += gear * 5;
        if(speed > 160) this.speed = 160;
    }

    public void speedDown(){
        this.speed -= 10;
        if(speed < 0) this.speed = 0;
    }


    public void gearUp(){
        if(this.gear < this.maxGear) this.gear++;
    }

    public void gearDown(){
        if(this.gear > 1) this.gear --;
    }

}
```

先に作成したCarクラスと仕様が違うメソッド、及び新しいメソッド以外は、この様に省略して書くことが出来ます。
また、コンストラクタも、Carクラスと同じ部分は、『スーパーコンストラクタ』として、Carクラスのコンストラクタを呼び出す形で、ある程度省略することが出来ます。

4. 以前のCarクラスを継承したAutoCarクラス

では、ManualCarクラスではなく、AutoCarクラスはどうなるでしょうか？
Then, how to construct AutoCar as AT Car class?

```
/* AutoCarクラス - 継承Ver. */
class ManualCar extends Car{


    private int gear;
    private final int maxGear;

    ManualCar(double consumption, double fuel, int maxGear){
        super(comsumption, fuel);
        this.gear = 1;
        this.maxGear = maxGear;
    }

    public void speedUp(){
        this.speed += gear * 5;
        if(speed > 160) this.speed = 160;
        this.gearValidate();
    }

    public void speedDown(){
        this.speed -= 10;
        if(speed < 0) this.speed = 0;
        this.gearValidate();
    }

    public void gearValidate(){
        gear = (speed - 25) / 10;
        if(gear < 1) gear = 1;
        else if (gear > maxGear) gear = maxGear;
    }


}
```

AT車のギアチェンジのタイミングは適当ですが、MT車のある程度最適と言える（らしい）ギアチェンジのタイミングをここでは設定してみました。
いかがでしょうか？

5. 他の種類の車はどうするか？

オートマチック車の中にも、いわゆるギアを持つAT車と、ギアが無段階なCVT車があります。
また、車の大きさや重さによって、軽自動車やトラック、セダン、スポーツ、ワンボックス、ミニバン、バン…など、車の種類は様々ですね。
こういった、他の種類の車も、オブジェクト指向ではうまく分類してあげることによって、根本的な『車』というクラスから、どんどん再利用しながら細かく分類し、作り上げていくことが可能になります。
是非、自分でトライアルしてみてください。