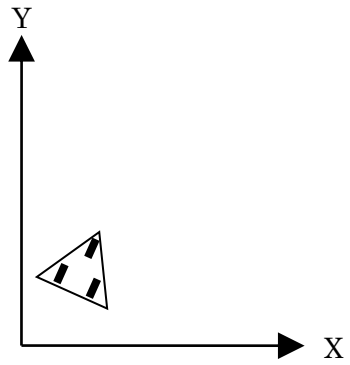
車シミュレーションソフト　設計・仕様書

車シミュレーションソフトの概要

車シミュレーションソフトは、アクセル・ブレーキ・ハンドルの各運転装置を操作すると、車の位置と向きがどのように変化するのかをシミュレートするソフトである。

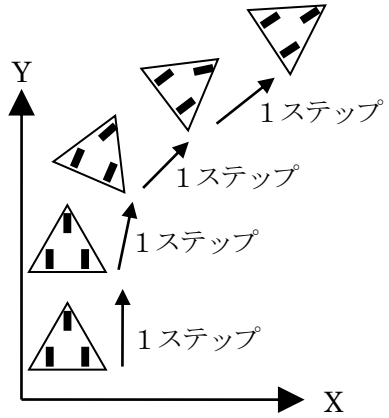
モデル概念

１．空間モデル



横軸X、縦軸Yの平面空間内に車体が存在する。車体位置は、車体中心の座標（X，Y）で表現される。

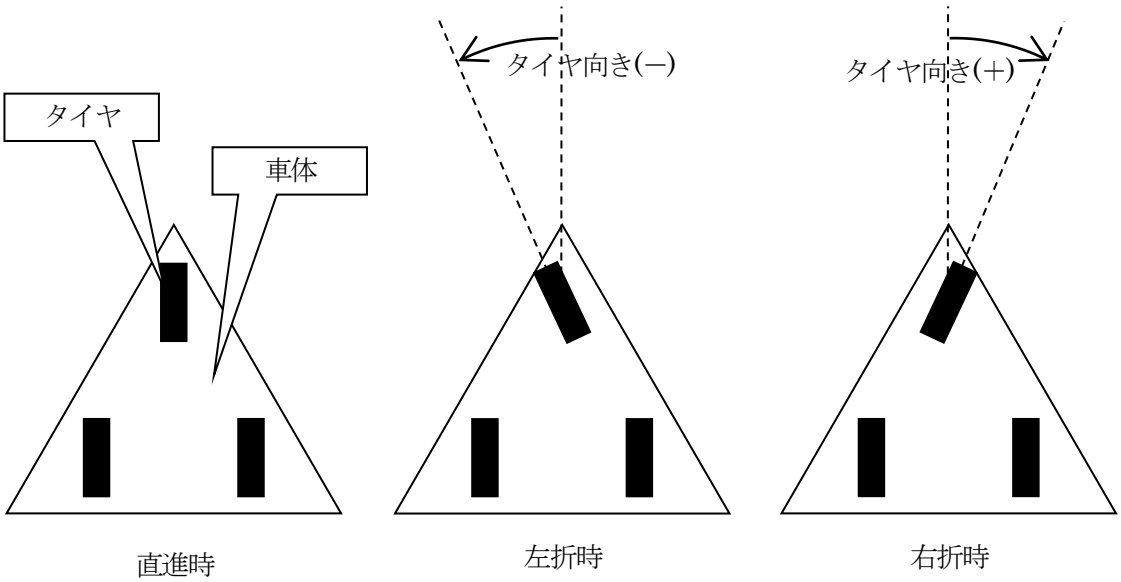
２．時間モデル



このソフトでは、時間については考慮しない。車体移動のメソッドmove()を実行するたびに、約１秒間分の車体移動の計算を行うが、move()メソッドの実行タイミングは任意である。１秒間隔より頻繁にmove()を実行してもよいし、１秒より長い間隔でmove()を実行してもよい。１回のmove()メソッドの実行を１ステップと呼ぶ。なお、move()メソッドはCarオブジェクトのメソッドである。

３．車モデル

以下のような三角型三輪車を考える。後輪２つは車体に付随するだけなので、このソフトでは後輪については何ら考慮していない。前輪は、その向きによって車体の向きを変える。前輪タイヤの向きは左折側を(－)、右折側を(＋)とする。例えば、前輪タイヤが－５° を向いていれば、車体は１ステップごとに左へ５° ずつ転回していく。



車の操縦方法

車の移動はCar オブジェクトの move()メソッドを実行することにより行う。move()メソッドはオーバーロードにより、以下の２種類ある。

ア．move()

車の操縦装置（アクセル・ブレーキ・ハンドル）を操作することなく、今までの状態を保ったまま（例えばハンドルを右に切ったままの状態）で、車体の位置移動を行う。

イ．move(int brake_d_push, int accel_d_push, int handle_d_rotate)

車の操縦装置を操作した上で、車体の位置移動を行う。メソッドの引数に入れるのは、それぞれ「ブレーキ踏み込みの変化量」「アクセル踏み込みの変化量」「ハンドル回転の変化量」である。例えばブレーキが１０ｃｍ踏み込まれている状態の時に move(-2, 0, 0)を実行すると、ブレーキの踏み込み量が８ｃｍに変更される。

操縦プログラム作成のヒント

以下の例は、アクセルを１５ｃｍ踏み込み、そのままの状態で５ステップ進んだあと、さらに右へ5° ハンドル切った状態で３ステップ進み、その後アクセルとハンドルを戻しつつブレーキを１５ｃｍ踏んで１０ステップ進むコードである。

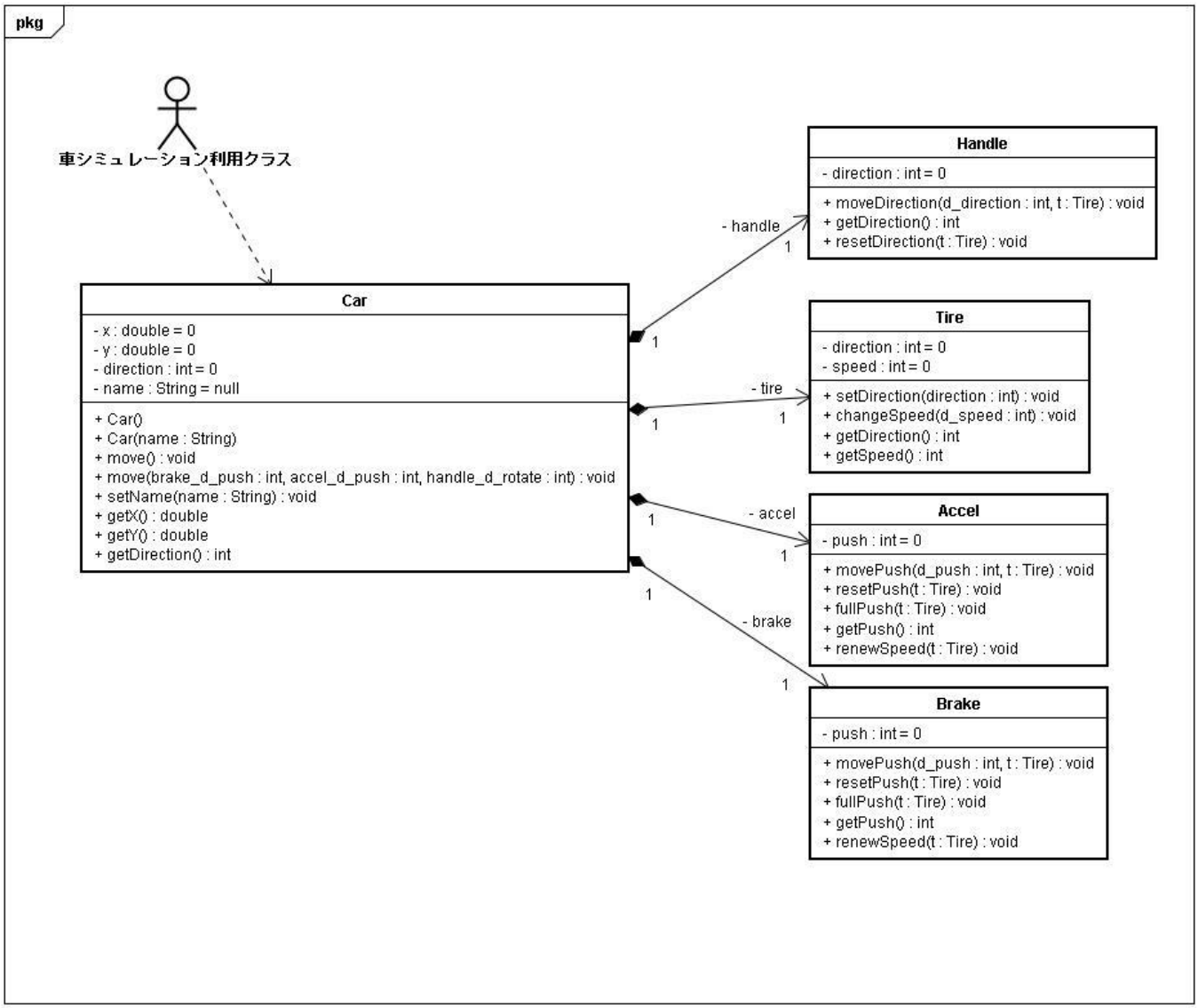
```
move(0, 15, 0);
for(int i = 0; i < 5; i++){
    move();
}
move(0, 0, 5);
for(int i = 0; i < 3; i++){
    move();
}
move(15, -15, -5);
for(int i = 0; i < 10; i++){
    move();
}
```

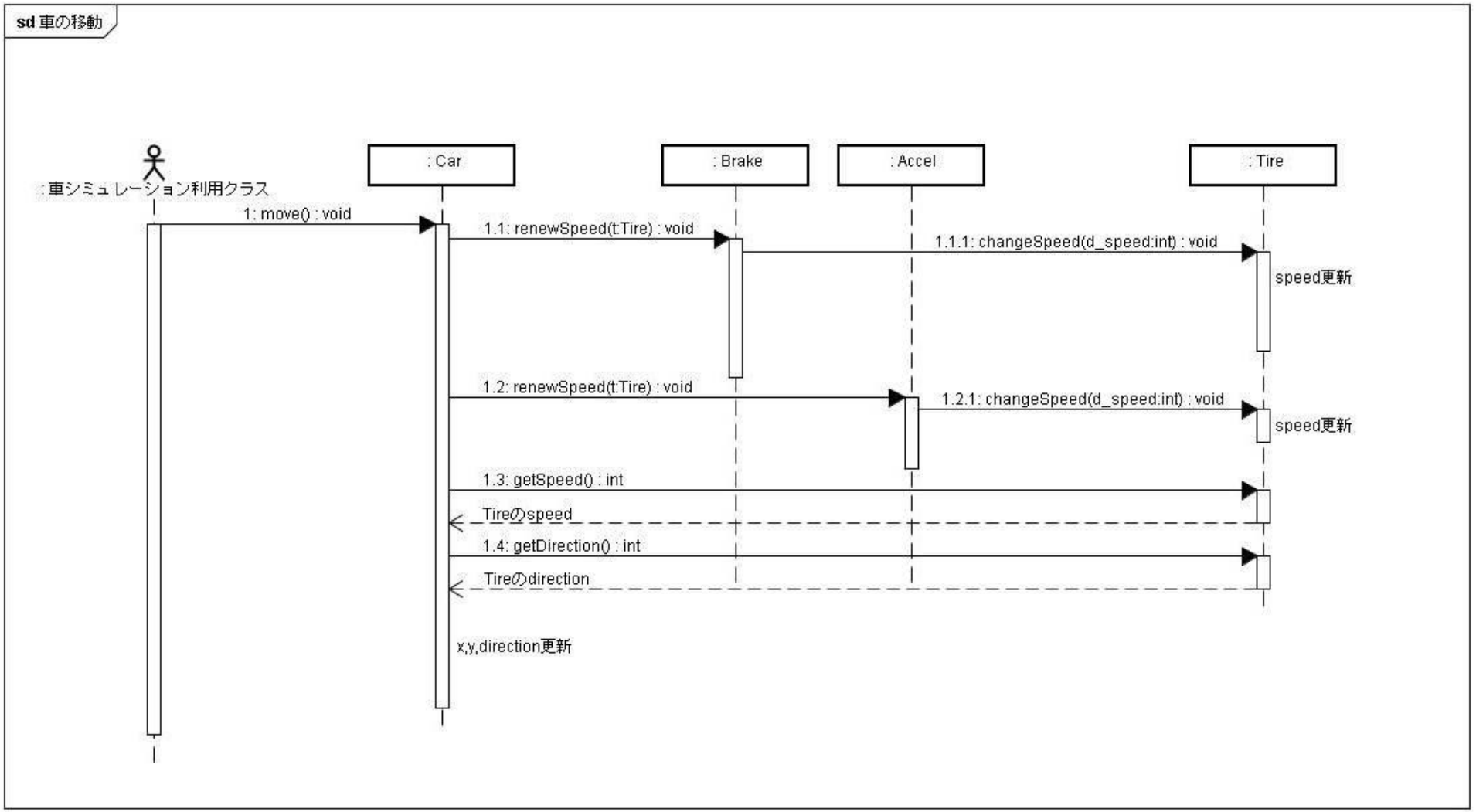
実際には、ステップ毎にX, Y座標および車体向きを取得し、画面表示もしなければならない。画面表示処理を追加したコードは以下のようになる。無駄なく見やすいプログラムにするよう工夫が必要である。

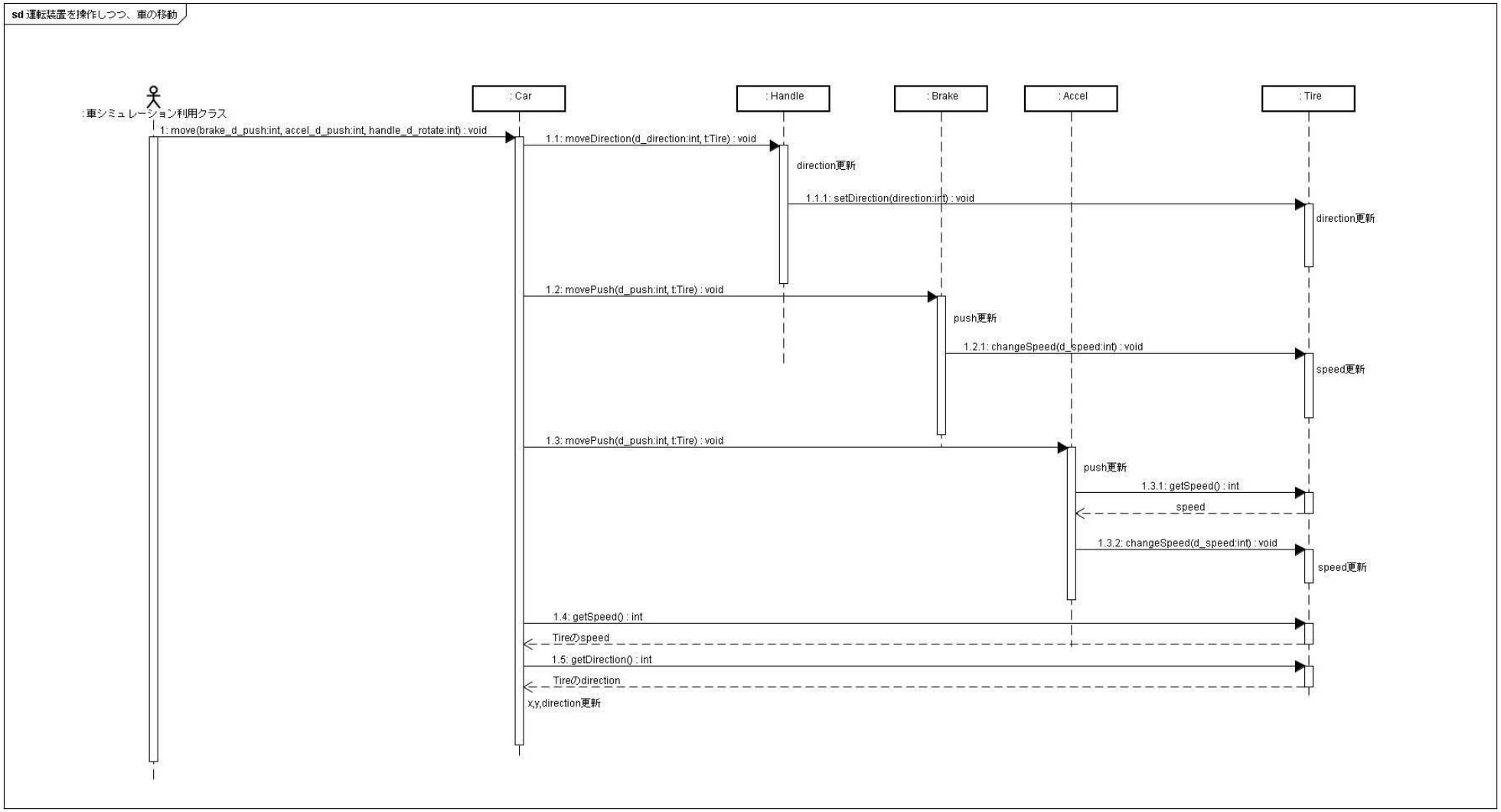
```
move(0, 15, 0);
System.out.println("座標("+getX()+", "+getY()+")  向き "+getDirection());
for(int i = 0; i < 5; i++){
    move();
    System.out.println("座標("+getX()+", "+getY()+")  向き "+getDirection());
}
move(0, 0, 5);
System.out.println("座標("+getX()+", "+getY()+")  向き "+getDirection());
for(int i = 0; i < 3; i++){
    move();
    System.out.println("座標("+getX()+", "+getY()+")  向き "+getDirection());
}
move(15, -15, -5);
System.out.println("座標("+getX()+", "+getY()+")  向き "+getDirection());
for(int i = 0; i < 10; i++){
    move();
    System.out.println("座標("+getX()+", "+getY()+")  向き "+getDirection());
}
```

なお、move()メソッドの実行間隔を調整するために一定時間プログラム実行を停止するには、次の手法が利用できる。このコードが実行されると一定時間停止する。太字部分が停止時間（単位ミリ秒）である。

```
try{
    Thread.sleep(1000);
}catch(Exception e){
    e.printStackTrace();
}
```


pkg
車シミュレーション利用クラス
Car
- x : double = 0
- y : double = 0
- direction : int = 0
- name : String = null
+ Car()
+ Car(name : String)
+ move() : void
+ move(brake_d_push : int, accel_d_push : int, handle_d_rotate : int) : void
+ setName(name : String) : void
+ getX() : double
+ getY() : double
+ getDirection() : int
Handle
- direction : int = 0
+ moveDirection(d_direction : int, t : Tire) : void
+ getDirection() : int
+ resetDirection(t : Tire) : void
Tire
- direction : int = 0
- speed : int = 0
+ setDirection(direction : int) : void
+ changeSpeed(d_speed : int) : void
+ getDirection() : int
+ getSpeed() : int
Accel
- push : int = 0
+ movePush(d_push : int, t : Tire) : void
+ resetPush(t : Tire) : void
+ fullPush(t : Tire) : void
+ getPush() : int
+ renewSpeed(t : Tire) : void
Brake
- push : int = 0
+ movePush(d_push : int, t : Tire) : void
+ resetPush(t : Tire) : void
+ fullPush(t : Tire) : void
+ getPush() : int
+ renewSpeed(t : Tire) : void
- handle
- tire
- accel
- brake
1


sd 車の移動
: 車シミュレーション利用クラス
: Car
: Brake
: Accel
: Tire
1: move() : void
1.1: renewSpeed(t:Tire) : void
1.1.1: changeSpeed(d_speed:int) : void
speed更新
1.2: renewSpeed(t:Tire) : void
1.2.1: changeSpeed(d_speed:int) : void
speed更新
1.3: getSpeed() : int
Tireのspeed
1.4: getDirection() : int
Tireのdirection
x,y,direction更新


sd 運転装置を操作しつつ、車の移動
:車シミュレーション利用クラス
: Car
: Handle
: Brake
: Accel
: Tire
1: move(brake_d_push:int, accel_d_push:int, handle_d_rotate:int) : void
1.1: moveDirection(d_direction:int, t:Tire) : void
direction更新
1.1.1: setDirection(direction:int) : void
direction更新
1.2: movePush(d_push:int, t:Tire) : void
push更新
1.2.1: changeSpeed(d_speed:int) : void
speed更新
1.3: movePush(d_push:int, t:Tire) : void
push更新
1.3.1: getSpeed() : int
speed
1.3.2: changeSpeed(d_speed:int) : void
speed更新
1.4: getSpeed() : int
Tireのspeed
1.5: getDirection() : int
Tireのdirection
x,y,direction更新

class Car Ｉ／Ｆ仕様

パッケージ	デフォルト
クラス名	public class Car 車を表すクラス。
フィールド	private double x 車のｘ座標。範囲規定なし。初期値0。単位ｍ
	private double y 車のｙ座標。範囲規定なし。初期値0。単位ｍ
	private int direction 車の向き。ｙ軸正方向を0とし、時計回りに正の値、反時計回りに負の値を取る。初期値0。 単位　：　° 値の有効範囲　：　-180 ～ 179
	private String name 車の名称。初期値null
	private Handle handle 車が持つハンドルオブジェクト。
	private Tire tire 車が持つタイヤオブジェクト。
	private Accel accel 車が持つアクセルオブジェクト。
	private Brake brake 車が持つブレーキオブジェクト。
コンストラクタ	public Car() 特に車の名称を指定しないコンストラクタ。
	public Car(String name) 車の名称を指定するコンストラクタ。 String name　：　車の名称。
メソッド	public void move() 車を移動および回転させるメソッド
	public void move(int brake_d_push, int accel_d_push, int handle_d_rotate) 車を移動および回転させ、かつブレーキ・アクセル・ハンドルを操作するメソッド。 int brake_d_push　：　ブレーキ踏み込み量の変更量。＋は踏み込み量を増やす。－は踏み込み量を減らす。単位はcm。 int accel_d_push　：　アクセル踏み込み量の変更量。＋は踏み込み量を増やす。－は踏み込み量を減らす。単位はcm。 int handle_d_rotate　：　ハンドル角の変更量。＋は右回転。－は左回転。単位ぱ
	public void setName(String name) 車の名称をセットする。 String name　：　名称。
	public double getX() 車のｘ座標値を得る。
	public double getY() 車のｙ座標値を得る。
	public int getDirection() 車の向きを得る。

class Car コンストラクタ詳細仕様

public Car()

特に車の名称を指定しないコンストラクタ。

１．フィールド handle に、Handle オブジェクトを生成して代入する。
２．フィールド tire に、Tire オブジェクトを生成して代入する。
３．フィールド accel に、Accel オブジェクトを生成して代入する。
４．フィールド brake に、Brake オブジェクトを生成して代入する。

public Car(String name)

車の名称を指定するコンストラクタ。

String name　：　車の名称。

１．引数なしコンストラクタ Car()を実行する。
２．フィールド name に、引数 name の値を代入する。

class Car メソッド詳細仕様

public void move()

車を移動および回転させるメソッド

1.Brake オブジェクトと Accel オブジェクトに対し、renewSpeed()メソッドを利用してタイヤの速度の更新を要求する。
2．Tire オブジェクトの getSpeed()メソッドを利用し、タイヤの速度を得る。
3．Tire オブジェクトの getDirection()メソッドを利用し、タイヤの向きを得る。
4．タイヤの速さから、車の速さを得る。次式によって計算する。
（車の速さ）＝（タイヤの速さ）
5．タイヤの向きから、車の向きを更新する。次式によって計算する。
（新しい車の向き）＝（現在の車の向き）＋（タイヤの向き）
ただし、車の向きは 179+1=-180 となり、-180-1=179 となることに留意すること。
6．車の速度と向きから、車の x 座標値と y 座標値を更新する。次式によって計算する。
x 座標　：　（新しい x 座標）＝（現在の x 座標）＋（車の速さ）＊SIN（車の向き）＊ 5 ／ 18
この式は、Java では次のように記述する。
x = x + (double)speed * Math.sin((double)direction * Math.PI / 180.0) * 5.0 / 18.0
y 座標　：　（新しい y 座標）＝（現在の y 座標）＋（車の速さ）＊COS（車の向き）＊ 5 ／ 18
この式は、Java では次のように記述する。
y = y + (double)speed * Math.cos((double)direction * Math.PI / 180.0) * 5.0 / 18.0

※SJC-P 試験では三角関数の正しい使い方までは問われないが、Math クラス自体は試験範囲に含まれている。sin()と cos()の引数と戻り値が double 型であることは知っておく必要がある。
※式の最後に付いている 5/18 は、[km/h]を[m/s]に変換している。

public void move(int brake_d_push, int accel_d_push, int handle_d_rotate)

車を移動および回転させ、かつブレーキ・アクセル・ハンドルを操作するメソッド。
int brake_d_push　：　ブレーキ踏み込み量の変更量。＋は踏み込み量を増やす。－は踏み込み量を減らす。単位は cm。
int accel_d_push　：　アクセル踏み込み量の変更量。＋は踏み込み量を増やす。－は踏み込み量を減らす。単位は cm。
int handle_d_rotate　：　ハンドル角の変更量。＋は右回転。－は左回転。単位は°

1．Handle オブジェクトに対し、moveDirection()メソッドを利用してハンドルを回転させる。
2．Brake オブジェクトに対し、movePush()メソッドを利用してブレーキの踏み込み量を変更する。
3．Accel オブジェクトに対し、movePush()メソッドを利用してアクセルの踏み込み量を変更する。
4．タイヤオブジェクトから速度と向きを得、車の向きと座標を更新する。その手順は move()メソッドと同様。

public void setName(String name)

車の名称をセットする。
String name　：　名称。

public double getX()

車の x 座標値を得る。

public double getY()

車の y 座標値を得る。

public int getDirection()

車の向きを得る。

class HandleＩ／Ｆ仕様

パッケージ	デフォルト
クラス名	public class Handle ハンドルを表すクラス。
フィールド	private int direction ハンドルの回転角度。右に切れているとき＋、左に切れているとき－。 初期値　：　0 有効範囲　：　-90 ～ +90 単位　：　°
コンストラクタ	public Handle() 特に処理はない。
メソッド	public void moveDirection(int d_direction, Tire t) ハンドルを回転させるメソッド。 int d_drection　：　ハンドル角度の変更量。＋なら右回転、－なら左回転。単位は° Tire t　：　ハンドル回転によって角度が変わるタイヤのオブジェクト。
	public int getDirection() ハンドルの回転角度を得る。
	public void resetDirection(Tire t) ハンドルを中立に戻すメソッド。 Tire t　：　ハンドル回転によって角度が変わるタイヤのオブジェクト。

class Handle メソッド詳細仕様

public void moveDirection(int d_direction, Tire t)

ハンドルを回転させるメソッド。

int d_direction　：　ハンドル角度の変更量。＋なら右回転、－なら左回転。単位は°

Tire t　：　ハンドル回転によって角度が変わるタイヤのオブジェクト。

１．ハンドルの回転角度の値を更新する。更新は次式によって計算する。

（新しいハンドルの角度）＝（現在のハンドルの角度）＋　d_direction

ただし、有効範囲を超えて回転させることはできない。(+90 を超えたら+90、-90 未満になったら-90 にする。)

２．Tire オブジェクトの setDirection()メソッドを利用し、タイヤの向きを設定する。例えば次式によって設定する。

setDirection(direction / 2)

この式は自由に変更して良い。この式を変更することによって、ハンドルに対するタイヤの反応が変わる。

public int getDirection()

ハンドルの回転角度を得る。

public void resetDirection(Tire t)

ハンドルを中立に戻すメソッド。

Tire t　：　ハンドル回転によって角度が変わるタイヤのオブジェクト。

１．ハンドルの回転角度の値を 0 にする。

２．Tire オブジェクトの setDirection()メソッドを利用し、タイヤの向きを設定する。要領は moveDirection()と同様。

class TireＩ／Ｆ仕様

パッケージ	デフォルト
クラス名	public class Tire タイヤを表すクラス。
フィールド	private int direction タイヤの向き。右に切れているとき＋、左に切れているときー。 初期値　：　0 有効範囲　：　-45 ～ +45 単位　：　°
	private int speed タイヤの速さ。 初期値　：　0 有効範囲　：　0以上 単位　：　km/h
コンストラクタ	public Tire() 特に処理はない。
メソッド	public void setDirection(int direction) タイヤの向きを設定するメソッド。 int drection　：　タイヤの向き。＋なら右に回転、－なら左に回転。単位は°
	public void changeSpeed(int d_speed) タイヤの速さを変更するメソッド。 int d_speed　：　タイヤの速さの変更量。＋なら加速、－なら減速。単位はkm/h
	public int getDirection() タイヤの向きを得る。
	public int getSpeed() タイヤの速さを得る。

class Tireメソッド詳細仕様

public void setDirection(int direction) タイヤの向きを設定するメソッド。 int direction　：　タイヤの向き。＋なら右に回転、－なら左に回転。単位は° １．タイヤの向きを更新する。更新は次式によって計算する。 （新しいタイヤの向き）＝　direction ただし、タイヤの向きの有効範囲を超えてはいけない。(+45を超えたら+45、-45未満になったら-45にする。)
public void changeSpeed(int d_speed) タイヤの速さを変更するメソッド。 int d_speed　：　タイヤの速さの変更量。＋なら加速、－なら減速。単位はkm/h １．タイヤの速さを更新する。更新は次式によって計算する。 （新しいタイヤの速さ）＝（現在のタイヤの速さ）＋　d_speed ただし、タイヤの速さの有効範囲を超えてはいけない。(0未満になったら0にする。)
public int getDirection() タイヤの向きを得る。
public int getSpeed() タイヤの速さを得る。

class Accel Ｉ／Ｆ仕様

パッケージ	デフォルト
クラス名	public class Accel アクセルを表すクラス。
フィールド	private int push アクセルの踏み込み量。 初期値　：　0 有効範囲　：　0 ～ 15 単位　：　cm
コンストラクタ	public Accel() 特に処理はない。
メソッド	public void movePush(int d_push, Tire t) アクセルの踏み込み量を変更するメソッド。 int d_push　：　アクセル踏み込み変更量。＋なら踏み込み量増やす、－なら踏み込み量減らす。 単位はcm。 Tire t　：　アクセル操作によって速さが変わるタイヤのオブジェクト。
	public void resetPush(Tire t) アクセルの踏み込み量を0にするメソッド。 Tire t　：　アクセル操作によって速さが変わるタイヤのオブジェクト。
	public void fullPush(Tire t) アクセルの踏み込み量を最大にするメソッド。 Tire t　：　アクセル操作によって速さが変わるタイヤのオブジェクト。
	public int getPush() アクセルの踏み込み量を得る。
	public void renewSpeed(Tire t) アクセルの踏み込み量に従って、タイヤの速さを変更するメソッド。 Tire t　：　アクセル操作によって速さが変わるタイヤのオブジェクト。

class Accel メソッド詳細仕様

public void movePush(int d_push, Tire t)

アクセルの踏み込み量を変更するメソッド。

int d_push ： アクセル踏み込み変更量。＋なら踏み込み量増やす、－なら踏み込み量減らす。単位は cm。

Tire t ： アクセル操作によって速さが変わるタイヤのオブジェクト。

１．アクセル踏み込み量を更新する。更新は次式によって計算する。

（新しい踏み込み量）＝（現在の踏み込み量）＋ d_push

ただし、踏み込み量の有効範囲を超えてはいけない。(0 未満になったら 0、15 を超えたら 15 にする。)

２．タイヤオブジェクトの changeSpeed()メソッドを利用して、タイヤの速さを更新する。例えば更新は次のように行う。

changeSpeed((12 *（アクセル踏み込み量）- (現在のタイヤの速さ)) / 18)

この式はフルアクセルのときに最大 180km/h 出るように調整してある。最初は加速度が大きいが、速度が上がるにつれて加速も弱くなる。アクセルを戻せば徐々に減速する。

この式は自由に変更して良い。この式を変更することによって、アクセルに対するタイヤの反応が変わる。

public void resetPush(Tire t)

アクセルの踏み込み量を 0 にするメソッド。

Tire t ： アクセル操作によって速さが変わるタイヤのオブジェクト。

１．アクセル踏み込み量を 0 に更新する。

２．タイヤオブジェクトの changeSpeed()メソッドを利用して、タイヤの速さを更新する。要領は movePush()と同様。

public void fullPush(Tire t)

アクセルの踏み込み量を最大にするメソッド。

Tire t ： アクセル操作によって速さが変わるタイヤのオブジェクト。

１．アクセル踏み込み量を最大値に更新する。

２．タイヤオブジェクトの changeSpeed()メソッドを利用して、タイヤの速さを更新する。要領は movePush()と同様。

public int getPush()

アクセルの踏み込み量を得る。

public void renewSpeed(Tire t)

アクセルの踏み込み量に従って、タイヤの速さを変更するメソッド。

Tire t ： アクセル操作によって速さが変わるタイヤのオブジェクト。

１．タイヤオブジェクトの changeSpeed()メソッドを利用して、タイヤの速さを更新する。要領は movePush()と同様。

class Brake Ｉ／Ｆ仕様

パッケージ	デフォルト
クラス名	public class Brake ブレーキを表すクラス。
フィールド	private int push ブレーキの踏み込み量。 初期値　：　0 有効範囲　：　0 ～ 15 単位　：　cm
コンストラクタ	public Brake() 特に処理はない。
メソッド	public void movePush(int d_push, Tire t) ブレーキの踏み込み量を変更するメソッド。 int d_push　：　ブレーキ踏み込み変更量。＋なら踏み込み量増やす、－なら踏み込み量減らす。 単位はcm。 Tire t　：　ブレーキ操作によって速さが変わるタイヤのオブジェクト。
	public void resetPush(Tire t) ブレーキの踏み込み量を0にするメソッド。 Tire t　：　ブレーキ操作によって速さが変わるタイヤのオブジェクト。
	public void fullPush(Tire t) ブレーキの踏み込み量を最大にするメソッド。 Tire t　：　ブレーキ操作によって速さが変わるタイヤのオブジェクト。
	public int getPush() ブレーキの踏み込み量を得る。
	public void renewSpeed(Tire t) ブレーキの踏み込み量に従って、タイヤの速さを変更するメソッド。 Tire t　：　ブレーキ操作によって速さが変わるタイヤのオブジェクト。

class Brake メソッド詳細仕様

public void movePush(int d_push, Tire t)

ブレーキの踏み込み量を変更するメソッド。

int d_push ： ブレーキ踏み込み変更量。＋なら踏み込み量増やす、－なら踏み込み量減らす。単位は cm。

Tire t ： ブレーキ操作によって速さが変わるタイヤのオブジェクト。

１．ブレーキ踏み込み量を更新する。更新は次式によって計算する。

（新しい踏み込み量）＝（現在の踏み込み量）＋ d_push

ただし、踏み込み量の有効範囲を超えてはいけない。(0 未満になったら 0、15 を超えたら 15 にする。)

２．タイヤオブジェクトの changeSpeed()メソッドを利用して、タイヤの速さを更新する。例えば更新は次のように行う。

changeSpeed(－（ブレーキ踏み込み量）)

この式は自由に変更して良い。この式を変更することによって、ブレーキに対するタイヤの反応が変わる。

public void resetPush(Tire t)

ブレーキの踏み込み量を 0 にするメソッド。

Tire t ： ブレーキ操作によって速さが変わるタイヤのオブジェクト。

１．ブレーキ踏み込み量を 0 に更新する。

２．タイヤオブジェクトの changeSpeed()メソッドを利用して、タイヤの速さを更新する。要領は movePush()と同様。

public void fullPush(Tire t)

ブレーキの踏み込み量を最大にするメソッド。

Tire t ： ブレーキ操作によって速さが変わるタイヤのオブジェクト。

１．ブレーキ踏み込み量を最大値に更新する。

２．タイヤオブジェクトの changeSpeed()メソッドを利用して、タイヤの速さを更新する。要領は movePush()と同様。

public int getPush()

ブレーキの踏み込み量を得る。

public void renewSpeed(Tire t)

ブレーキの踏み込み量に従って、タイヤの速さを変更するメソッド。

Tire t ： ブレーキ操作によって速さが変わるタイヤのオブジェクト。

１．タイヤオブジェクトの changeSpeed()メソッドを利用して、タイヤの速さを更新する。要領は movePush()と同様。