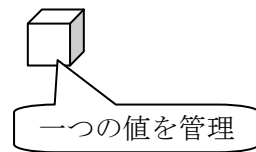


● クラスとは

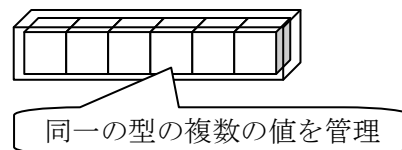
クラス

変数と関数を併せもつデータ型

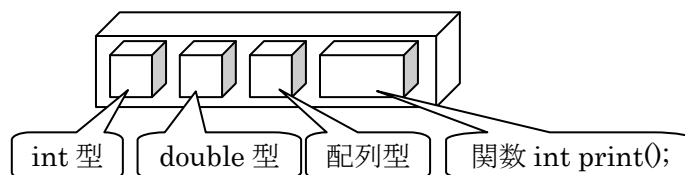
直観的に表現すると、
int 型や double 型は、



配列型は、



クラスは、



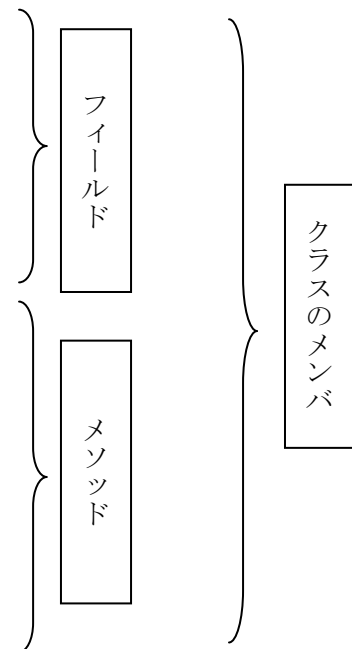
● クラスの宣言

クラスの宣言

クラスのフィールド（変数）とメソッド（関数）を定義して、新しいデータ型として利用できるようにする。
フィールドとメソッドはクラスのメンバと呼ばれる。

クラスの宣言はキーワード `class` を指定して次のように行う

```
class クラス名
{
    型 フィールド名;
    .
    .
    .
    戻り値の型 メソッド名(引数リスト)
    {
        文;
        .
        .
        return 式;
    }
    .
    .
}
```



※メソッドは後期の授業で解説する

例えば、ナンバーとガソリン量をフィールドにもつクラス **Car** は次のように宣言できる

ソースコード例

ソースファイル名 : Sample12_1.java

```
// クラス Car の宣言
class Car
{
    int number;    // ナンバーを格納する int 型の変数
    double gas;    // ガソリン量を格納する double 型の変数
}
```

次に、宣言したクラスの利用の仕方を説明する。

● クラスの利用

クラスの利用手順

クラスのオブジェクトを扱うクラス型の変数の宣言



クラスのオブジェクトの生成



オブジェクトの各フィールドの初期化

クラスのオブジェクト

クラスのメンバを格納するための領域

※ クラスのオブジェクトはインスタンスとも呼ばれる

クラスのオブジェクトを扱う変数の宣言は、クラス名と識別子を指定して次のように行う

```
クラス名 識別子;
```

例えば、

```
Car    car1;    // クラス Car 型のオブジェクトを扱う変数を宣言
```

クラスのオブジェクトの生成は、new 演算子とクラス名を指定して次のように行う

```
識別子 = new クラス名(); // ← 丸括弧“()” ※ 配列の場合は “[ ]” です 違いに注意！！
```

例えば、

```
car1 = new Car();    // クラス Car 型のオブジェクトを生成
```

オブジェクトがもつ各フィールドは、オブジェクトを扱う変数の識別子とピリオド“.”、フィールド名を用いて参照する。フィールドの初期化は各フィールドを参照して次のように行う。

```
識別子.フィールド名 = 値;
```

例えば、

```
car1.number = 9129;  
car1.gas=30.0;
```

ソースコード例

ソースファイル名 : Sample12_2.java

```
// 車クラスの宣言とその利用

// クラス Car の宣言
class Car
{
    int number; // ナンバー
    double gas; // ガソリン量
}

class Sample12_2
{
    public static void main(String[] args)
    {
        Car car1;           // クラス Car 型のオブジェクトを扱う変数
        car1 = new Car();    // クラス Car 型のオブジェクトを生成

        // 上記を同時に行うこともできる
        // Car car1 = new Car();

        // 各フィールドに値を代入
        car1.number = 9129;
        car1.gas = 30.0;

        // 各フィールドの値を出力
        System.out.println("車のナンバーは" + car1.number + "です。");
        System.out.println("ガソリン量は" + car1.gas + "です。");
    }
}
```

実行画面

```
>java Sample12_1
車のナンバーは 9129 です。
ガソリン量は 30.0 です。
-- Press any key to exit (Input "c" to continue) --
```

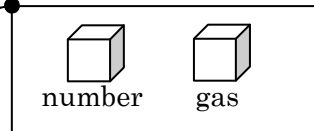
参考：例題 Sample12_2 の変数の振る舞いは図的に次のように理解できる

`Car car1;` // クラス `Car` 型の変数



`car1`

`car1 = new Car();` // クラス `Car` 型のオブジェクトを確保



オブジェクトまたはインスタンスという

○ クラスの配列を作ってみよう

クラスは int 型や double 型と同じデータ型の一つ → 同様にしてクラスの配列を作成できる

ソースコード例

ソースファイル名 : Sample12_3.java

```
// 車クラスの配列

// クラス Car の宣言
class Car
{
    int number; // ナンバー
    double gas; // ガソリン量
}

class Sample12_3
{
    public static void main(String[] args)
    {
        Car[] car;          // 配列変数 : クラス Car 型の配列を扱う変数
        car = new Car[2];    // 配列要素 : クラス Car 型のオブジェクトを扱う変数

        car[0] = new Car();  // クラス Car 型のオブジェクト 0 を生成
        car[1] = new Car();  // クラス Car 型のオブジェクト 1 を生成

        // 各フィールドに値を代入
        car[0].number = 9129;
        car[0].gas = 30.0;

        car[1].number = 1234;
        car[1].gas = 15.5;

        // 各フィールドの値を出力
        for(int i=0;i<car.length;i++)
        {
            System.out.println(i+"番目の車情報 : ");
            System.out.println("車のナンバーは" + car[i].number + "です。");
            System.out.println("ガソリン量は" + car[i].gas + "です。");
        }
    }
}
```

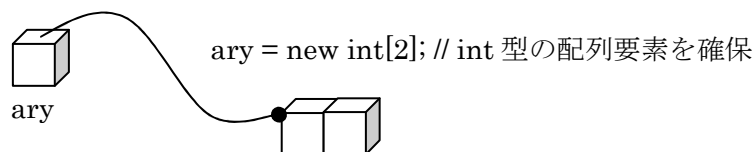
実行画面

```
>java Sample12_3
0 番目の車情報：
車のナンバーは 9129 です。
ガソリン量は 30.0 です。
1 番目の車情報：
車のナンバーは 1234 です。
ガソリン量は 15.5 です。
-- Press any key to exit (Input "c" to continue) --
```

参考：例題 Sample12_3 の変数の振る舞いは図的に次のように理解できる

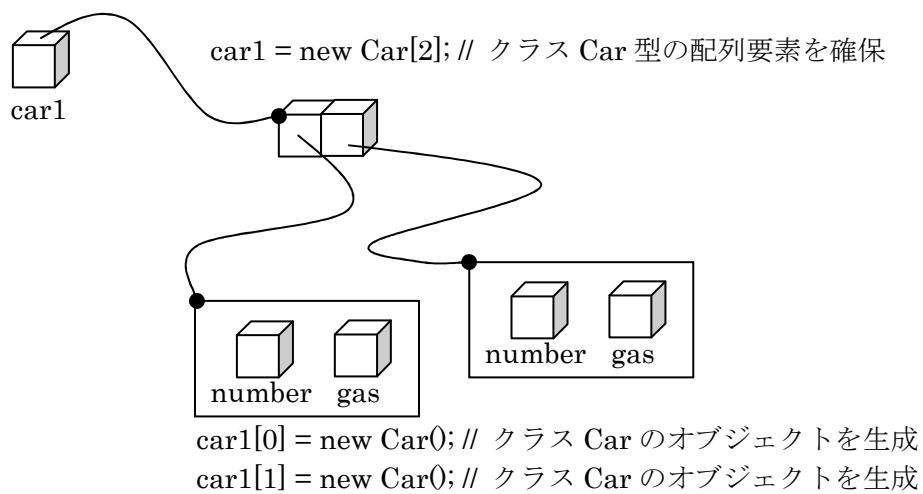
【int 型の配列の場合】

```
int[] ary; // Int 配列型の変数
```



【クラスの配列の場合】

```
Car[] car1; // クラス Car 配列型の変数
```



○ クラスのメンバに他のクラスを宣言してみよう

クラスは int 型や double 型と同じデータ型の一つ → クラスを他のクラスのメンバにできる

ソースコード例

ソースファイル名 : Sample12_4.java

```
// 車クラスを別のクラスのメンバにする

// クラス Car の宣言
class Car
{
    int number; // ナンバー
    double gas; // ガソリン量
}

// クラス Car をメンバに持つクラス Owner の宣言
class Owner
{
    String name;
    int age;
    Car mycar; // クラス Car 型のオブジェクトを扱う変数をメンバにもつ
}

class Sample12_4
{
    public static void main(String[] args)
    {
        Owner owner1;           // クラス Owner 型のオブジェクトを扱う変数
        owner1 = new Owner();     // クラス Owner 型のオブジェクトを生成

        owner1.name = "Java";
        owner1.age = 21;
        owner1.mycar = new Car(); // クラス Car 型のオブジェクトを生成
        owner1.mycar.number = 9129;
        owner1.mycar.gas = 30.0;

        // 各フィールドの値を出力
        System.out.println("所有者");
        System.out.println("名 前:"+owner1.name);
        System.out.println("年 齢:"+owner1.age);
        System.out.println("車");
        System.out.println("車ナンバー:"+owner1.mycar.number);
        System.out.println("ガソリン量:"+owner1.mycar.gas);
    }
}
```

実行画面

```
>java Sample12_4  
所有者  
名 前:Java  
年 齢:21  
車  
車ナンバー:9129  
ガソリン量:30.0  
-- Press any key to exit (Input "c" to continue) --
```