



アクセス指定子とプロパティ

基本編7日目

メンバへのアクセスの許可

- **アクセス指定子**

- フィールド・メソッドのアクセス範囲を決める
- **public**、**protected**、**internal**、**protected internal**、**private**がある

アクセス修飾子の種類

アクセス指定子	呼び名	意味
public	パブリック	どこからでも呼び出せる
protected	プロテクティッド	同一クラスか、そのサブクラスからしか呼び出せない
internal	インターナル	同一のアセンブリ（DLL）内でアクセスが可能
protected internal	プロテクティッド インターナル	protected かつ internal
private	プライベート	同じクラス内からしか呼び出せない

カプセル化

- 利用者に関係のないものは見せないという考え方
- オブジェクト指向言語一般で用いられる
- 原則的にフィールドはprivateにして外部から隠蔽する
- アクセスが必要であれば**セッター**及び**ゲッター**を付ける

プロパティ

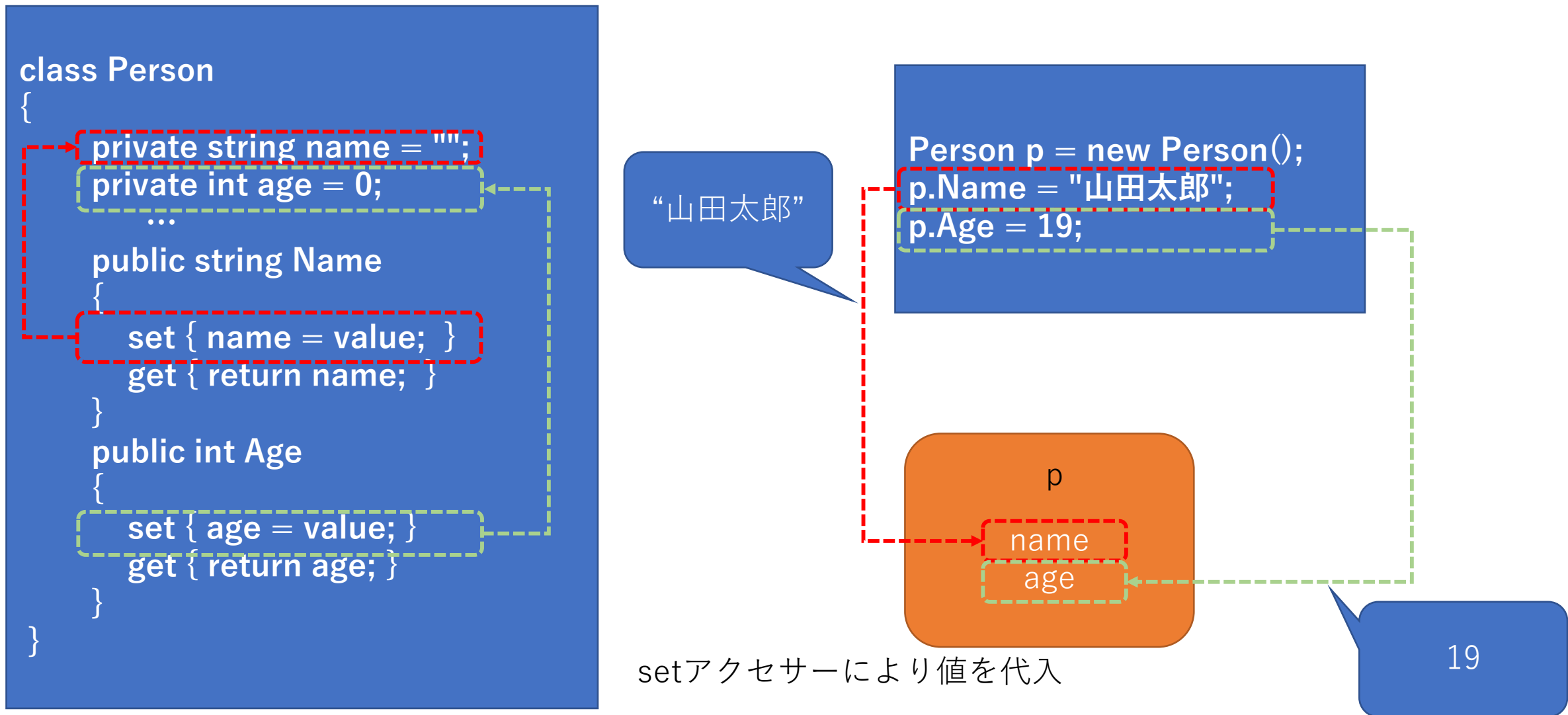
- 通常フィールドは、**private**で隠蔽
- 外部からアクセスする処理を介して値の変更・取得を行う
- **セッター** ... フィールドに値を書き込む
- **ゲッター** ... フィールドの値を取得する
- C#言語のセッター・ゲッターを定義するものが**プロパティ**

プロパティの定義方法

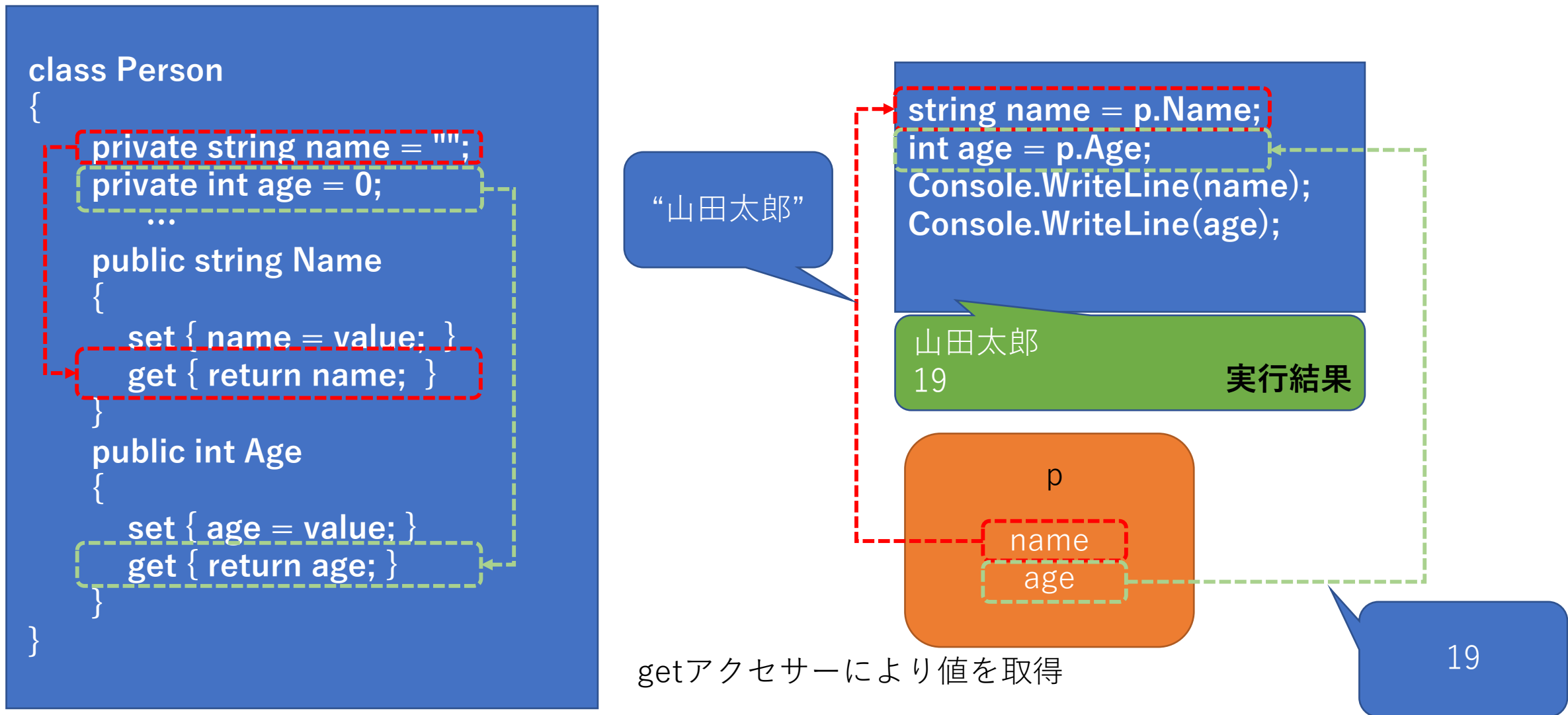
アクセス修飾子 型名 プロパティ名

```
{  set
  {
    // setアクセサー (setter とも言う)
    // ここに値の変更時の処理を書く。
    // value という名前の変数に代入された値が格納される。
  }
  get
  {
    // getアクセサー (getter とも言う)
    // ここに値の取得時の処理を書く。
    // メソッドの場合と同様に、値はreturnキーワードを用いて返す。
  }
}
```

プロパティの利用例①(セッター)



プロパティの利用例②(ゲッター)



setterのみ・getterのみのプロパティ

```
class Hoge
{
    private string name = "";
    public string Name
    {
        set { name = value; }
    }
}
```

```
Hoge h = new Hoge();
h.Name = "山田";
Console.WriteLine(h.Name);
```

○
×

セッターのみだと、値を代入できても
取得することが出来ない。

```
class Fuga
{
    private string name = "佐藤";
    public string Name
    {
        get { return name; }
    }
}
```

```
Fuga f = new Fuga();
f.Name = "山田";
Console.WriteLine(f.Name);
```

×
○

ゲッターのみだと、値を取得できても
代入することが出来ない。

自動実装プロパティ①

- 自動プロパティとも言う
- C#3.0から導入された機能
- 対応するフィールドを用意する必要がないプロパティ
- 読み込み専用なども定義できる


自動実装プロパティ ②

Person2クラスのName, Ageプロパティ

外部から書き込み・読み込み可能な例

```
public int Age
{
    set; get;
}
```

```
Person2 p = new Person2();
...
p.Age = 32;
Console.WriteLine(p.Age);
```




○
○

外部から読み込み専用

```
public string Name
{
    private set; get;
}
```

```
Person2 p = new Person2();
...
p.Name = “佐藤花子”;
Console.WriteLine(p.Name);
```



×
○

自動実装プロパティ ③

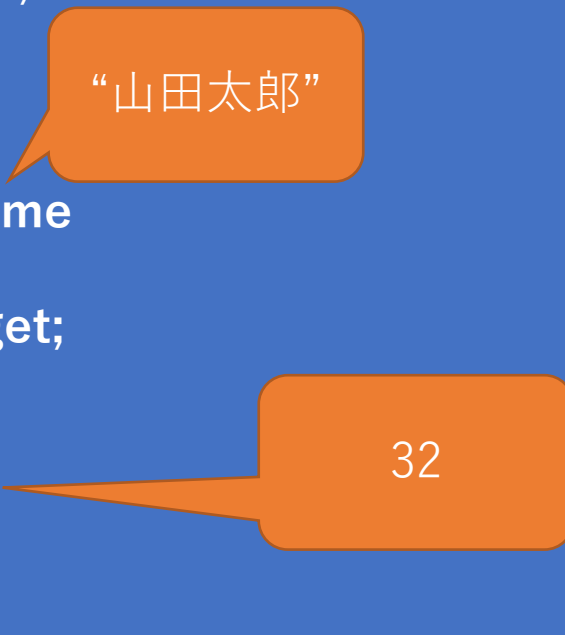
```
class Person2
{
    public void SetAgeAndName(string name, int age)
    {
        Name = name;
        Age = age;
    }
    ...
    public string Name
    {
        private set; get;
    }
    public int Age
    {
        set; get;
    }
}
```

The diagram illustrates the automatic implementation of properties. A red dashed line connects the `SetAgeAndName` method to the `set` accessors of the `Name` and `Age` properties. Orange callout boxes indicate the values being assigned: "山田太郎" for `Name` and 26 for `Age`.

```
Person2 p = new Person2();
p.SetAgeAndName("山田太郎", 26);
```

自動実装プロパティ④

```
class Person2
{
    public void SetAgeAndName(string name, int age)
    {
        Name = name;
        Age = age;
    }
    ...
    public string Name
    {
        private set ; get;
    }
    public int Age
    {
        set; get;
    }
}
```



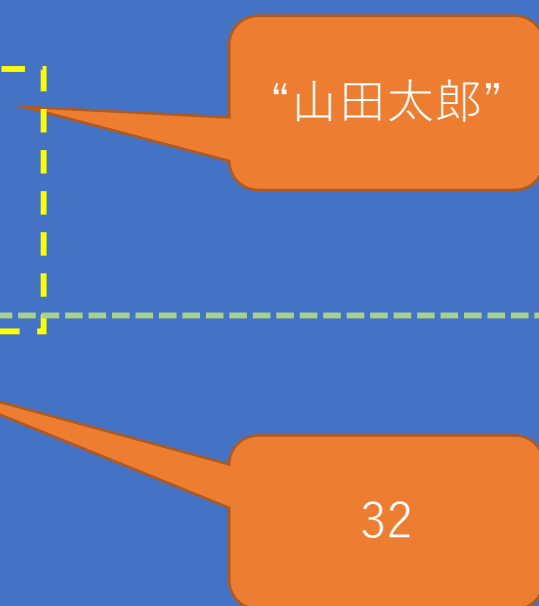
“山田太郎”

32

p.Age = 32;

自動実装プロパティ⑤

```
class Person2
{
    public void SetAgeAndName(string name, int age)
    {
        Name = name;
        Age = age;
    }
    ...
    public string Name
    {
        private set; get;
    }
    public int Age
    {
        set; get;
    }
}
```



```
p.Name = "佐藤花子";
```

