14章 クラスを理解しよう

14章 クラスを理解しよう

クラスを理解し、似たようなモノ（オブジェクト）を大量生産する方法を学びます

本章の目標

- クラスを理解し、似たようなモノ（オブジェクト）を大量生産できるようになること
- クラスをより柔軟に使いこなせるようになること

大量のデータをゼロから個別に作るのは大変です

- Amazonのようなショッピングサイトで大量の商品データを作る場合
- ゼロから個別の商品データを作るのは極めて大変
- クラスを使えば、類似するデータを大量に作れる

オブジェクト

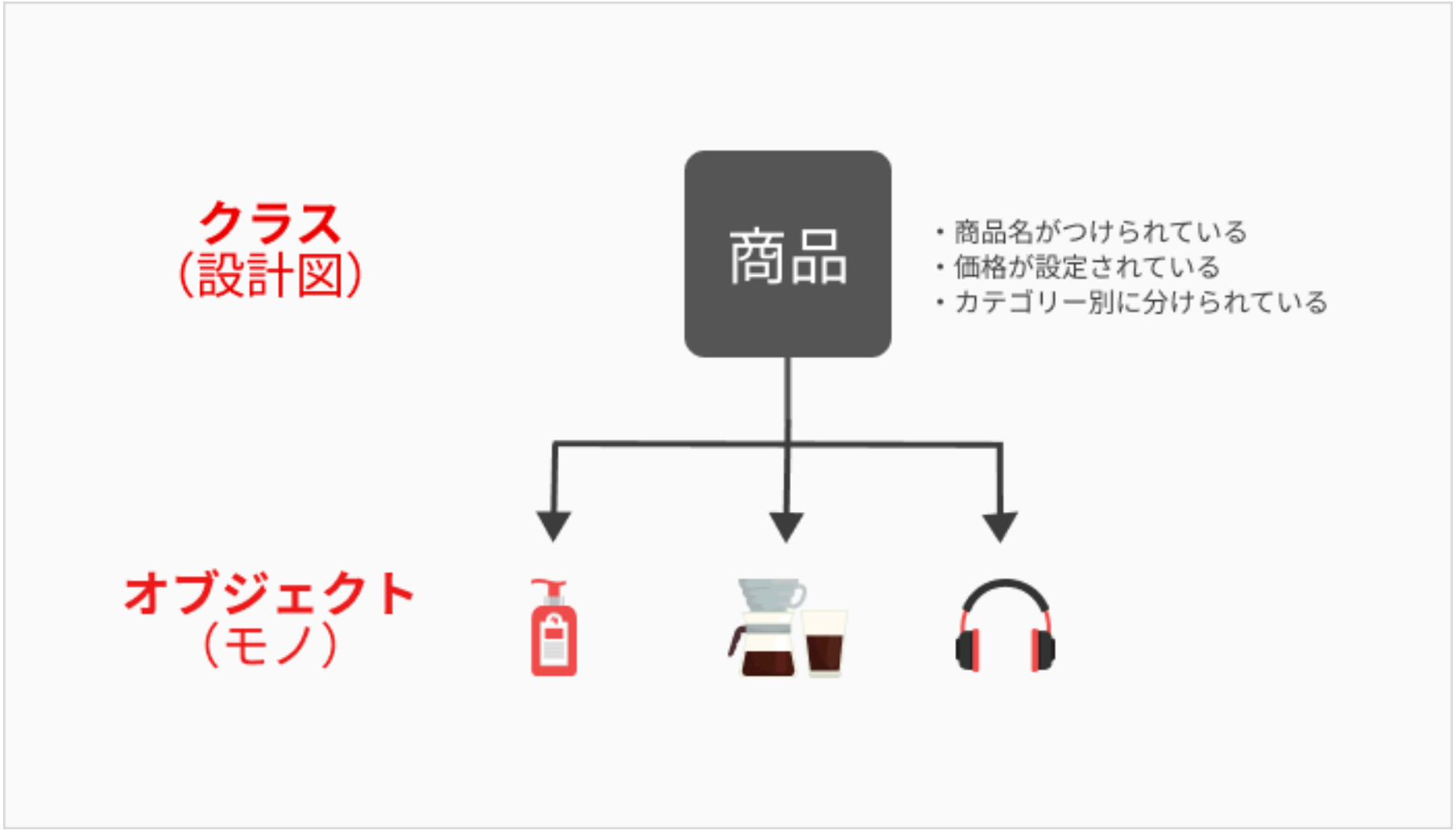
変数

- 商品名
- 価格
- カテゴリー
- 在庫数 など

関数

在庫数を更新する関数 など

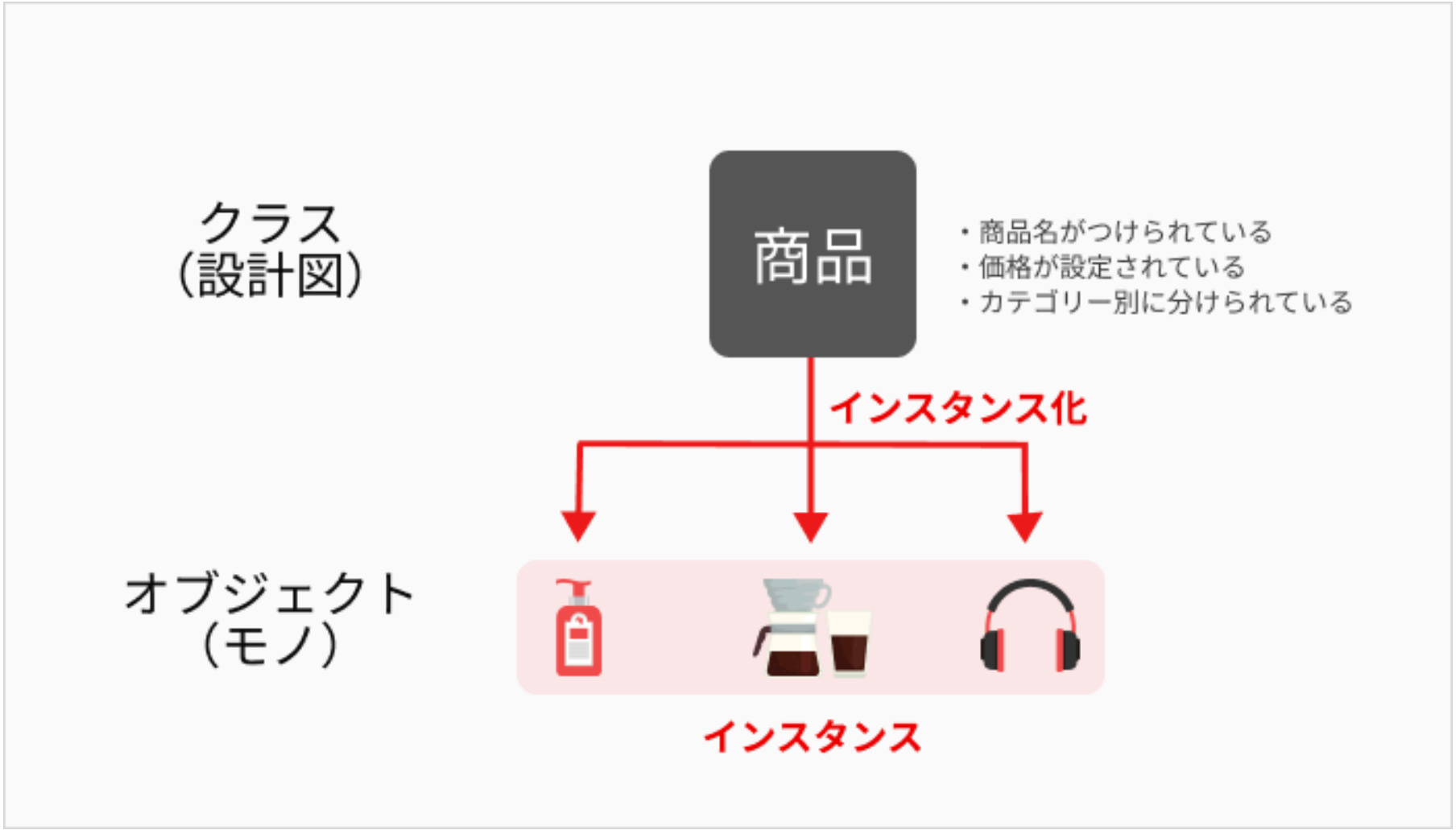
14章 クラスとは


クラス
（設計図）
商品
・商品名がつけられている
・価格が設定されている
・カテゴリー別に分けられている
オブジェクト
（モノ）

クラス（設計図）から同じ特徴を持ったオブジェクトを大量生産できます

- インスタンス：クラスをもとに作られたオブジェクトのこと
- インスタンス化：クラスをもとにインスタンスを作ること
- インスタンス＝「実体」という意味

14章 クラスとインスタンス



classキーワードを使ってクラスを定義します

```
class クラス名:
    クラスの特徴
```

- クラス名は慣習的に先頭を大文字にする

```
class Product:
    クラスの特徴
```

「クラス名()」と記述してインスタンス化します

```
# クラスを定義する
class Product:
    クラスの特徴

# インスタンス化する
shampoo = Product()
```

オブジェクト

属性

変数

- 商品名
- 価格
- カテゴリー
- 在庫数 など

関数

在庫数を更新する関数 など

クラス内のコンストラクタで属性を定義します

```
class Product:
    def __init__(self):
        # 属性を定義する
        self.name = ""
```

- `def __init__(self):` はコンストラクタ（後述）
- `self.name` が属性

インスタンスと属性名を「.（ドット）」でつなぎます

```
class Product:
    def __init__(self):
        self.name = ""

shampoo = Product()

# 属性にアクセスし、値を代入する
shampoo.name = "シャンプー"

# 属性にアクセスし、値を出力する
print(shampoo.name)
```

14章 属性を使ってみよう

```
class Product:
    def __init__(self):
        # 属性を定義する
        self.name = ""

shampoo = Product()

# 属性にアクセスし、値を代入する
shampoo.name = "シャンプー"

# 属性にアクセスし、値を出力する
print(shampoo.name)
```

```
シャンプー
```

オブ

オブジェクト

変数

- 商品名
- 価格
- カテゴリー
- 在庫数 など

メソッド

関数

在庫数を更新する関数 など

クラス内でdefを使ってメソッドを定義します

```
class Product:
    def __init__(self):
        self.name = ""

    # メソッドを定義する
    def set_name(self, name):
        self.name = name
```

インスタンスとメソッド名を「.（ドット）」でつなぎます

```
class Product:
    def __init__(self):
        self.name = ""

    def set_name(self, name):
        self.name = name

shampoo = Product()

# メソッドにアクセスして実行する
shampoo.set_name("シャンプー")
```

属性に値を代入・出力するメソッドを作成します

```
class Product:
    def __init__(self):
        self.name = ""

    def set_name(self, name):
        self.name = name

    def show_name(self):
        print(self.name)

coffee = Product()
coffee.set_name("コーヒー")
coffee.show_name()
```

14章 メソッドを使ってみよう

```
class Product:
    def __init__(self):
        self.name = ""

    # メソッドを定義する
    def set_name(self, name):
        self.name = name

    def show_name(self):
        print(self.name)


# インスタンス化する
coffee = Product()

# メソッドにアクセスして実行する
coffee.set_name("コーヒー")
coffee.show_name()
```

```
コーヒー
```

14章 コンストラクタとは

インスタンス化する際に処理を行うメソッドのことです

- コンストラクタによって実行される最初の処理を初期化という
- 「商品を作ると同時に出品する」といった最初の処理を設定できる
- constructor＝「建設者」という意味

`def __init__(self):`から始まります

```
class Product:
    def __init__(self):
        # 属性を定義する
        self.name = ""
```

- `self`は初期化されるインスタンスそのものを指す
- 引数を追加して属性に値をセットできる

14章 コンストラクタで属性に値をセット

引数を使って属性に値を代入できます

```
def __init__(self, name, age, gender):
    # self.~~~は全てこのインスタンスの属性になる
    self.name = name
    self.age = age
    self.gender = gender
```

Userクラスを作成し、属性に値を代入します

```
class User:
    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender

# インスタンス化する
user = User("侍太郎", 36, "男性")

# 属性にアクセスし、値を出力する
print(user.name)
print(user.age)
print(user.gender)
```

14章 コンストラクタを使ってみよう

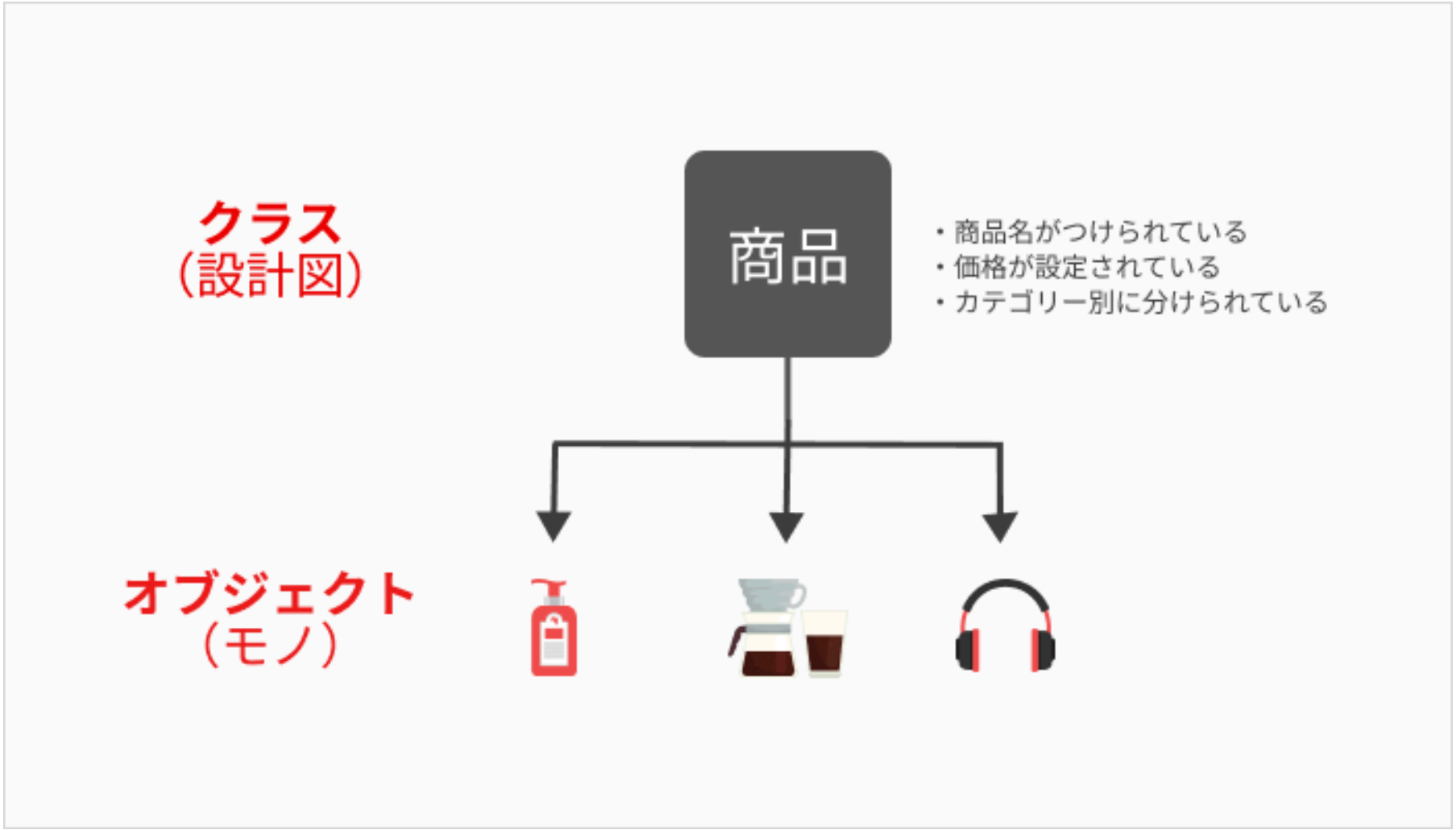
```python
class User:
    # コンストラクタを定義する
    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender

# インスタンス化する
user = User("侍太郎", 36, "男性")

# 属性にアクセスし、値を出力する
print(user.name)
print(user.age)
print(user.gender)
```

```
侍太郎
36
男性
```

14章 クラスのまとめ図解



14章 まとめ

本章では以下の内容を学習しました

オブジェクトとクラス

- オブジェクト：独自の変数や関数をひとまとめにしたもの
- クラス：モノ（オブジェクト）の設計図
- インスタンス：クラスをもとに作られたオブジェクト
- インスタンス化：クラスをもとにオブジェクトを作ること

14章 まとめ（続き）

本章では以下の内容を学習しました

属性・メソッド・コンストラクタ

- 属性：オブジェクトが持つデータ（変数）
- メソッド：オブジェクトが持つ関数
- コンストラクタ：インスタンス化するときに初期化を行うメソッド

```
class クラス名:
    def __init__(self):
        初期化の内容

    def メソッド名():
        一連の処理
```