■インスタンス（instance）

●モノの実体としてのインスタンス

お弁当屋さんでお弁当を買うことを考えて見ましょう。

メニューには幕の内弁当、から揚げ弁当、コロッケ弁当があるとします。
Aさんは幕の内弁当、Bさんはから揚げ弁当、Cさんはコロッケ弁当
を購入しました。
それぞれ持ち帰ることが出来るお弁当の実体は別々に存在します。

一方、幕の内弁当にはから揚げとコロッケが含まれます。
から揚げ弁当にはから揚げ、コロッケ弁当はコロッケがそれぞれお弁当の
おかずの主役です。
幕の内弁当に入っているコロッケとコロッケ弁当に入っているコロッケは
同じ種類のコロッケで別々に実体が存在します。

「お弁当」という枠組みでの実体とお弁当の中身の「おかず」としての実体は
どちらもインスタンスの具体例となります。
また、インスタンスの捕らえ方の範囲について、お弁当という中身の詳細を
問わない概念、おかずの組み合わせに着目した概念、お弁当のおかず単体に
着目するような概念など段階があることも分かります。
この概念的な段階を「抽象度」とここでは定義しておきます。

●オブジェクト指向から独立したインスタンス

以上のようなたとえ話はインスタンスという言葉とともにオブジェクト指向
の説明として語られることが多いようです。
また、言語処理系、例えばC++やJAVAなどに依存して語られることも一般的です。

ですが、インスタンスという言葉をオブジェクト指向とは無関係に
使うこともできます。

例えば、リアルタイムOS上のタスクとオブジェクト指向のオブジェクトは
概念も実装方式も異なりますが、タスクの実体を指すときにインスタンス
という言葉を使った方が大雑把な概念的な理解のきっかけとなります。

「タスクはメソッドだけを実体化したインスタンス」

というようなオブジェクト指向風に大胆に簡略化した説明を行った方が、
大筋では会話が成立してしまいます。
本当はリアルタイムOSの仕組み上のTCB(タスク・コントロール・ブロック)
の説明とレジスタ退避、スケジューラによるコンテキストスイッチなどの
正確な説明を行う必要があるのですが、全体的な設計上の概要把握だけが
目的の場合はOSの仕組みまでは理解していなくても話を進められます。

つまり、オブジェクト指向用語が他の実装や設計に対しての
「たとえ話」に使えるようになってきたということです。

また、このようなアプローチをすると、リアルタイムOSのタスク生成に
おける「同一機能タスクの複数生成」という概念も説明しやすくなります。

なお、お弁当の例は振る舞いのないデータだけのインスタンスの
たとえ話でしたが、タスクの例は振る舞いだけのインスタンスの
たとえ話になります。
リアルタイムOSでは振る舞いのインスタンスとデータのインスタンスは
別々に管理されています。

●組込みシステム特有の問題

これまでのたとえ話のインスタンスをソフトウェア実装の視点で見れば
全てメモリ領域に独立して存在するデータの集合と言うことが出来ます。
おかずを表す変数データとおかずやご飯を格納する「容器」は全て
メモリ上のデータやデータの並びになります。

オブジェクト指向言語仕様上はこのメモリ確保と管理の仕組みを
どのように表現するかということに重きが置かれています。
ですが、どのメモリにインスタンスを配置するかということはあまり
議論されません。

インスタンスを高速なSRAMに配置するのか、普通のDRAMに配置するのか、
もしくは静的にインスタンスを生成した上でROMに配置するのか、などの
議論です。
これらの議論は組込みシステムでは必要な場合があります。
組込みシステムではお弁当の容器を選択したい場合があるからです。

プラスチック容器、紙の容器、竹の容器、持ち込みの自分用の容器など
出来合いのお弁当屋さんの既定容器以外の容器が使える必要があります。

つまり、メモリ配置について意図的に指定する必要があるということです。
単なるクラス全体のメモリ配置の問題だけなのであれば、メモリ配置のための
コンパイラの拡張書式を使えば解決します。
通常、組込みシステム用のコンパイラにはこのような目的の拡張書式があります。

但し、クラスや機能全体ではなく一部の特定部分だけが性能上の問題になる
場合には、抽象度を上げた実装をわざわざ解体し、抽象度を低くした上で
メモリ配置を変える必要があるかもしれません。