2.3.4.8.3 Field Data

Return to Object Data Taxonomy

Overview

Return to Top

Field Data, or **Element Data**, are the values stored within the Object and in many ways represent the reason for the Object (i.e., Class in C++ or Java) to exist. In Procedural Language) **Field Data** can be thought of as a Variable with some variables having their values set at compile time (i.e., constants) and those during the execution of the procedure. **Note:** Methods or Operations in Objects are procedural in nature, generally represented by a block of code that executes from the top to the bottom, but can be circuitous within the block of code having logical expressions, looping and exceptions to control the flow within the block.

Constant Data is a quality of the data to not be modifiable during the execution of the procedure. There are two broad categories of constants: **Named Constants** and **Literal COnstants**. **Note: Constants** can be scoped to the particular procedure, the package of procedures or during the execution of the program.

Often the two terms: **Attribute** and **Property** are used interchangeable in Computer Science because we often use them to describe the fields (i.e., elements) of an Object in an Object-Oriented Programming (OOP). However, it is useful to understand the difference between the two grammatically:

Attribute Data is the quality or object that we attribute to someone or something. For example, intelligence is an attribute of a person. You can not go to the store and buy intelligence, it is an attribute of the individual. Another example would be a planet belonging to a solar system. Planets do not usually exist outside a solar system.

Property Data is a quality that exists without any attribution. Geometric shapes are properties in their own right. They can be used to independently describe many things. For example, a planet or a ball can be described as a sphere[1)]. A definition of the sphere exists without the attribution to a planet or a ball. Therefore, we say that a planet or a ball has spherical properties.

In Unified Modeling Language (UML), an **Attribute** and a **Property** both represent an structural association between two entities. **Attributes** are most often represented as **Composition**[2)], while **Property** is best represented by **Aggregation** [3)].

According to Lenny Delligatti[4)], “UML Attributes are Data Types (e.g., Strings, Integers, etc.) whereas Systems Modeling Language (SysML) properties are ValueTypes (e.g., can be assigned computed values).”

There are different ways that **Field Data** can be stored. In most programming paradigms, the **Field Data** is **Mutable**, meaning the values are simply replaced or overwritten with new values. The Object

Accessor Methods and responsible for making sure the new values are syntactically and semantically correct. Figure 1 graphically illustrates the concept of **`mutable`**. Each time the **`setter`** is called, the contents of the **Field Data** within the **Data Object** are changed. **Note:** There can be **`setters`** for **Attribute** and **Property** data. In the example, the first time the **`setter`** is called, the value of the **attribute** is set to **2**, the next time it is **4**.

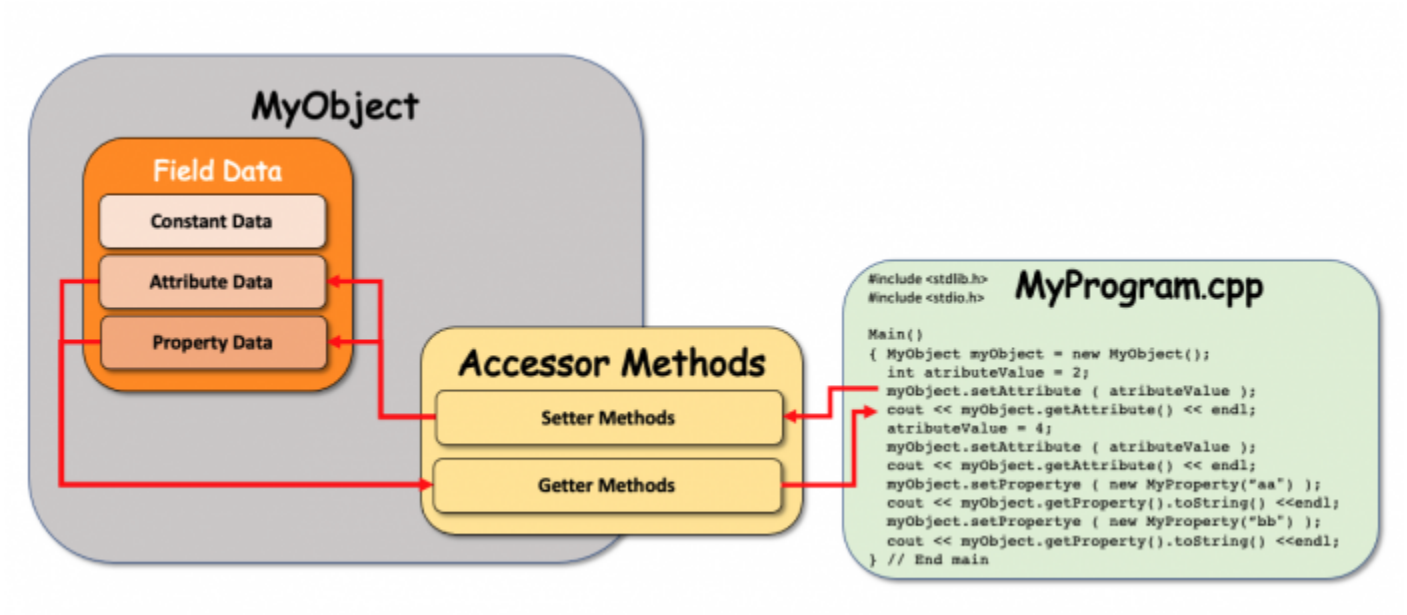

Figure 1: Each time the **`setter`** is called, the value of the **`Filed Data`** is changed.

However, in DIDOs, most of the **Field Data** is considered **immutable**, meaning that updates and modifications to the **Field Data** are made while not destroying the previous data, but a new entry is made for the Field Data that points back to the original value. See Section 2.1.6 Immutable Data Objects and Figure 2. **Note:** The concepts of **mutable** and **immutable** are different from those built into C/C++, which describes **immutable** more like constants.

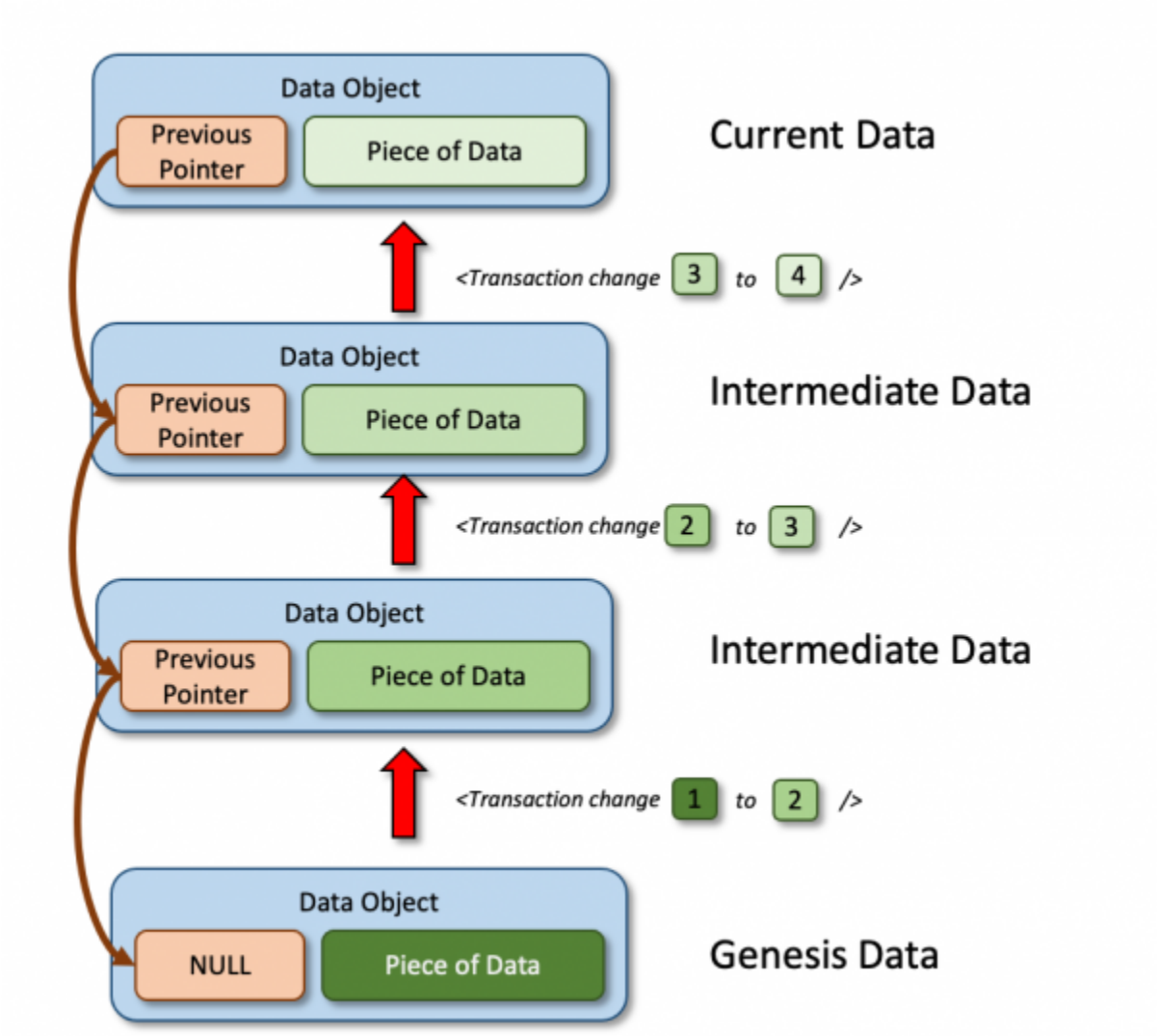

Figure 2: The Immutable Data Chain where the current value of a **Field Data** points to the previous value.

DIDO Specifics

Return to Top

To be added/expanded in future revisions of the DIDO RA

☒ [nick][✓ char, 2022-03-22]~~Is this page still under construction?~~
☒ [char][✓ char, 2022-03-22]~~New Section -- review~~

1)
Sphere, In geometry, the set of all points in three-dimensional space lying the same distance (the radius) from a given point (the center), or the result of rotating a circle about one of its diameters. The components and properties of a sphere are analogous to those of a circle.https://www.britannica.com/science/sphere
2)

The **Composition** is a part of aggregation, and it portrays the whole-part relationship. It depicts dependency between a composite (parent) and its parts (children), which means that if the composite is discarded, so will its parts get deleted. It exists between similar objects.
https://www.javatpoint.com/uml-association-vs-aggregation-vs-composition
3)

Aggregation is a subset of association; it is a collection of different things. It represents a “has a” relationship.

- It is more specific than an association
- It describes a part-whole or part-of relationship
- It is a binary association, i.e., it only involves two classes
- It is a kind of relationship in which the child is independent of their parent.

https://www.javatpoint.com/uml-association-vs-aggregation-vs-composition
4)

Lenny Delligatti, SysML Distilled: A Brief Guide to the Systems Modeling Language, FIrst Edition, Addison-Wesley Professional , 8 November 2013, ISBN-13: 978-0321927866