25. Glossary

Description

In this chapter, we shortly describe the main terms used in this book with references to the relevant sections in the book.

The rightmost column is supposed to contain a link to the part(s) of the book where the term is introduced. It is currently incomplete.

Term	Explanation	References
Access modifiers	Supplementary properties that restrict access to attributes of an object including `%public`, `%private`, `%protected`, `%package_boundary` and `%package`.	
Action	The accomplishment of creating/destroying physical phenomena and/or changing properties of physical phenomena in a domain.	
Atomic operation	An operation that cannot be interrupted while it is being executed.	
Active object	An object that performs computations of its own: a program object, a coroutine or parallel object. An object that includes statements that are executed as part of the generation of the object, is not considered an active object.	
Activity	A partial ordered sequence of actions.	
Attribute	A local data-item, method, or class of an object.	
Attribute accessor	An expression that describes access to an attribute of a specific object.	
Actual parameter	If the formal parameter is a data-item, then an actual parameter is the value of an expression assigned to the formal parameter at the invocation of a method or instantiation of a class. If the formal parameter is a virtual method or virtual class, then an actual parameter is a binding of the virtual.	
Argument	Alternative term for *actual parameter*.	
Class	A template for *objects* with identical structure.	
Class instantiation	The computation of generating an object from a class, and executing its statements. Similar to method invocation.	
Control structure abstraction, control abstraction	Alternative term for a *control method*.	
Control flow	The order in which the *operations* of a program are performed. It determines how a program moves from one operation to another based on the state of the objects.	
Control structure	A *statement* that specifies the order of execution of other statements based on state of the objects and thus controls the *control flow* of a program.	
Control structure method, Control method	A method that defines a control structure.	
Computation	A partial ordering of *operations* executed by a computer.	
Coroutine	Shorthand for *coroutine object*	

Term	Explanation	References
Coroutine object	An active object that represents an activity that may be suspended and later resumed at the point of suspension.	
Data-item	A *constant* or *variable object reference* or *value object*.	
Datum	A piece of information that may be a *value* or an *object reference*.	
Declaration	A description of a constant or variable data-item, a class or a method in an object descriptor or method descriptor.	
Derived	An *object descriptor* and *class* is said to be derived from its *superclass*.	
Direct superclass	The direct superclass is the class from which the class/singular object is explicitly derived as specified in the object descriptor for the class/singular object	
Expression	A description/syntactic element that describes a computation of a value.	
Formal parameter	A formal parameter of a method or class may be a variable data-item or a virtual method or virtual class. Se also *actual parameter*.	
Function	A method that computes a value solely based on the values of its parameters. It does not read or modify data-items in its enclosing object or other objects.	
Ghost object	An object that is used solely for the purpose of explaining a given programming language mechanism.	
Identity	A physical phenomena has an *identity*. Physical phenomena are distinct which means that they have different identities.	
Inheritance	An alternative term for the *subclass* mechanism.	
Item	A *declaration* or a *statement* in an *object-descriptor*.	
Intrinsic object	An object that is a declared using **`obj`**, is generated as part of the generation of the object containing the declaration and exists during the lifetime of this object.	
Invoker	The object (class object or *method object*) executing the *method invocation* that generates a given method object – the *invokee*.	
Invokee	The *method object* being generated as part of a *method invocation* executed by an object or method object – the *invoker*.	
Mainpart	The *items* defined in an *object-descriptor* excluding the *direct superclass*.	
Method	A template for *computations* with identical structure.	
Method generation	The computation of generating a *method object*; includes generation of possible data-items and assignments of possible parameters. Similar to *object generation*.	
Method invocation	The computation of a *method generation* from a *method description*, followed by execution of its *statements*. Similar to *object instantiation*	
Material object	A *singular object* declared using `obj` or an instance of a *class*.	
Method object	An *object* created from a *method*.	
Module	An object descriptor for a singular object that is organized in the `qBetaWorld` hierarchy and mapped to a folder and file in the filesystem.	
Object	Objects are the basic, physical entities that exist in computers; objects are the building blocks of computational processes. An object has variable and/or constant datums, intrinsic objects, classes, methods, and a computation part. A material objects has in addition an identity. A method objects does not have an identity.	

Term	Explanation	References
Object-descriptor	A description of the structure of an object consisting of a possible superclass, and possible items.	
Object generation	The computation of generating *an object*; includes generation of possible data-items and assignments of possible parameters. Similar to *method generation*.	
Object instantiation	The computation of an *object generation* from a *object descript*or, followed by execution of its *statements*. Similar to *method invocation*.	
Object reference	A data-item that may hold a reference to an object. See also *Reference*.	
Operation	An action executed by a computer.	
Parallel object	An active object that executes actions in parallel with other parallel objects.	
Parameter	Shorthand for *formal parameter*. See also actual parameter and argument.	
Program	A module that is intended for execution.	
Reference	Short-hand for *object-reference*.	
Scope	The part of the `qBetaWorld` where a given name is visible.	
Singular object	An object that is one of its kind and not an instance of a class.	
Singular method object	A method object that is one of its kind and not an instance of a method.	
Statement	A description of an *activity* (action?) in an *object-descriptor*.	
Structure	Shorthand for *object structure*.	
Subclass	A class that is derived from another class – its direct superclass, and inherits attributes and statements from the superclass.	
Superclass	A class from which other classes are derived. See also *direct superclass*.	
Submethod	A method that is derived from another method – its direct supermethod, and inherits attributes and statements from the supermethod.	
Supermethod	A method from which other methods are derived.	
Type	A *concept* that covers a class of *values*. This is a simplified defintion of a type intended for programming – the mathematical concept of a type is much more complicated and exists in different variants.	
Value	A mathematical entity like a number, point, polygon, etc.	
Value object	An object that is an instance of a subclass of class `Value`; the object is embedded in a *holding object* and intended for representing a value.	
Value type	A class that is a subclass of class `Value`. See also *Type*.	