

## 25. Glossary

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.

<b>Term</b>	<b>Explanation</b>	<b>References</b>
Access modifiers	Supplementary properties that restrict access to attributes of an object including <code>%public</code> , <code>%private</code> , <code>%protected</code> , <code>%package_boundary</code> and <code>%package</code> .	
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 <i>actual parameter</i> .	
Class	A template for <i>objects</i> 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 <i>control method</i> .	
Control flow	The order in which the <i>operations</i> 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 <i>statement</i> that specifies the order of execution of other statements based on state of the objects and thus controls the <i>control flow</i> of a program.	
Control structure method, Control method	A method that defines a control structure.	
Computation	A partial ordering of <i>operations</i> executed by a computer.	

Term	Explanation	References
Coroutine	Shorthand for <i>coroutine object</i>	
Coroutine object	An active object that represents an activity that may be suspended and later resumed at the point of suspension.	
Data-item	A <i>constant</i> or <i>variable object reference</i> or <i>value object</i> .	
Datum	A piece of information that may be a <i>value</i> or an <i>object reference</i> .	
Declaration	A description of a constant or variable data-item, a class or a method in an object descriptor or method descriptor.	
Derived	An <i>object descriptor</i> and <i>class</i> is said to be derived from its <i>superclass</i> .	
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 <i>actual parameter</i> .	
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 <i>identity</i> . Physical phenomena are distinct which means that they have different identities.	
Inheritance	An alternative term for the <i>subclass</i> mechanism.	
Item	A <i>declaration</i> or a <i>statement</i> in an <i>object-descriptor</i> .	
Intrinsic object	An object that is a declared using <b>obj</b> , 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 <i>method object</i> ) executing the <i>method invocation</i> that generates a given method object - the <i>invokee</i> .	
Invokee	The <i>method object</i> being generated as part of a <i>method invocation</i> executed by an object or method object - the <i>invoker</i> .	
Mainpart	The <i>items</i> defined in an <i>object-descriptor</i> excluding the <i>direct superclass</i> .	
Method	A template for <i>computations</i> with identical structure.	
Method generation	The computation of generating a <i>method object</i> ; includes generation of possible data-items and assignments of possible parameters. Similar to <i>object generation</i> .	
Method invocation	The computation of a <i>method generation</i> from a <i>method description</i> , followed by execution of its <i>statements</i> . Similar to <i>object instantiation</i>	
Material object	A <i>singular object</i> declared using <b>obj</b> or an instance of a <i>class</i> .	
Method object	An <i>object</i> created from a <i>method</i> .	

Term	Explanation	References
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.	
Object-descriptor	A description of the structure of an object consisting of a possible superclass, and possible items.	
Object generation	The computation of generating <i>an object</i> ; includes generation of possible data-items and assignments of possible parameters. Similar to <i>method generation</i> .	
Object instantiation	The computation of an <i>object generation</i> from a <i>object descriptor</i> , followed by execution of its <i>statements</i> . Similar to <i>method invocation</i> .	
Object reference	A data-item that may hold a reference to an object. See also <i>Reference</i> .	
Operation	An action executed by a computer.	
Parallel object	An active object that executes actions in parallel with other parallel objects.	
Parameter	Shorthand for <i>formal parameter</i> . See also actual parameter and argument.	
Program	A module that is intended for execution.	
Reference	Short-hand for <i>object-reference</i> .	
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 <i>activity</i> (action?) in an <i>object-descriptor</i> .	
Structure	Shorthand for <i>object structure</i> .	
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 <i>direct superclass</i> .	
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 <i>concept</i> that covers a class of <i>values</i> . 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.	

<b>Term</b>	<b>Explanation</b>	<b>References</b>
Value object	An object that is an instance of a subclass of class <code>Value</code> ; the object is embedded in a <i>holding object</i> and intended for representing a value.	
Value type	A class that is a subclass of class <code>Value</code> . See also <i>Type</i> .	