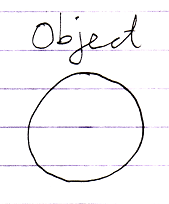
|  |
| --- |
| Circle Language Spec: System Objects |

## System Objects

This article demonstrates the system objects that all systems are composed of. The main purpose of system objects is to manage relations between objects.

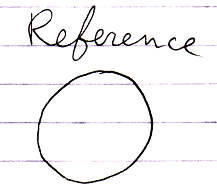
#### The Object

The most basic element of a computer program is the *object*. An object represents thing, an idea or a place, a number or a collection or anything else. All those things are called objects.



#### The Reference

You always deal with *references* to objects.



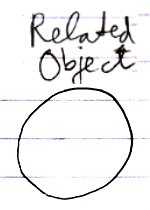
An object is never directly accessed.

An object *reference* is an *undecorated* pointer. It is used by other system

objects, that wrap an object reference into a context.

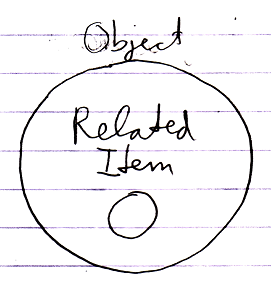
#### Related Object

A synonym for *object reference* is *related object*, except that a related object is in the context of another object.

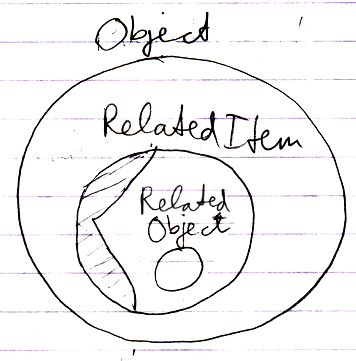


#### Related Item

A parent object can contain *related items*.



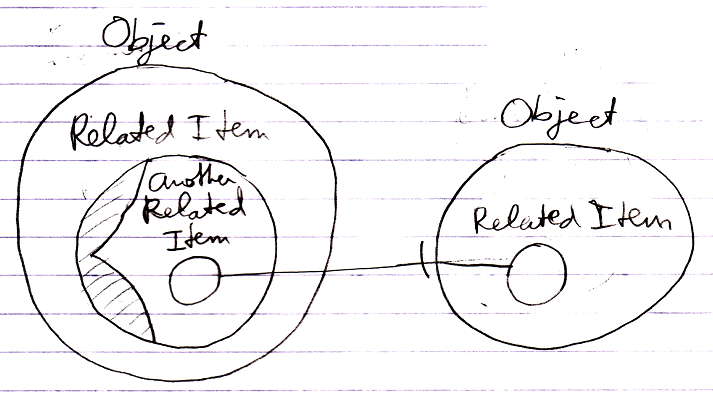
It is the result of a relation. A related item wraps a related object.



A related object is the undecorated object pointer wrapped inside a related item.

#### Pointer-to-Pointer

A related item can also wrap yet another related item, contained by another parent object.

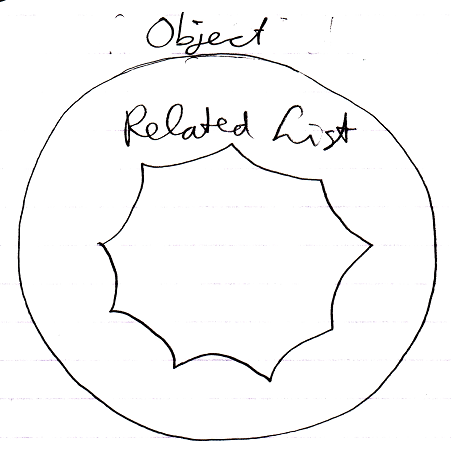


In that case it is said to be a *pointer-to-pointer*. The use of pointers to pointers makes you able to let something else determine what is eventually targeted.

You never work with objects directly, so even though the diagram above looks like a pointer, it is really a pointer-to-pointer.

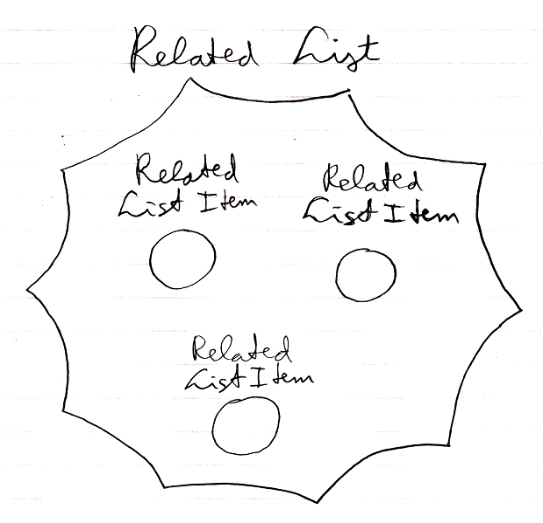
#### Related List

A parent object can also contain *related lists*.



#### Related List Item

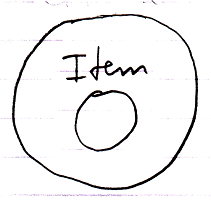
A related list can contains multiple *related list items*.



A *related list item* is much like a *related item*. Often, when the term related item is used, it also applies to related list items. In some cases though, the behavior is different.

#### Item

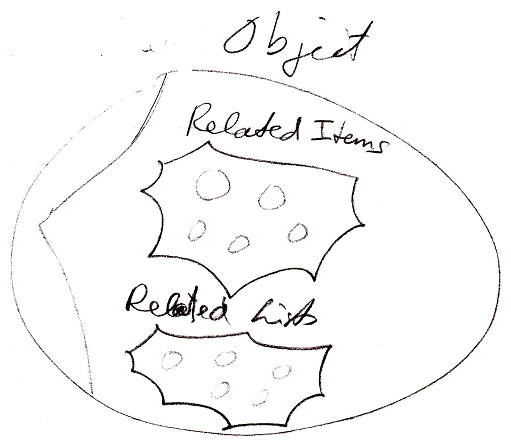
Another name for an object is an item. But when we speak of an item, we are usually think about an item inside a certain *context*.



So an item is a related object.

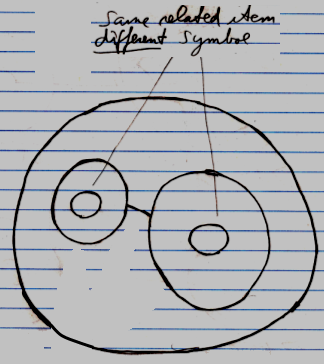
#### Related Items & Related Lists Collections

An object’s related items and related lists are actually stored as the object’s having two collections: Related Items and Related Lists. The Related Items collection contains all the related items of the object. The Related Lists collection contains all the related lists of the object.



#### Symbol

The term *symbol* should not to be confused with the term related item. A symbol is a shape displayed on screen. The same related item can be displayed on screen multiple times.



A symbol is not a system element, it is just a shape displayed on screen. *Symbol* is a diagram expression concept, not a coding concept.

However, sometimes the term symbol is used as a synonym for related item, because it just talks a lot easier. But they are not *really* synonyms.

#### System Object

All the terms above indicate different kinds of *system objects*, except for the term *symbol*.

Most of the system objects are part of the *code base*. System objects without any extensions need to be implemented right inside the code base. *Extensions* can be added later using the new computer language itself.

System objects are *actual* objects, that all other systems are based on.

A recap of all the terms introduced:

Object

Reference

Related Object

Related Item

Pointer-to-Pointer

Related List

Related List Item

Item

Related Items & Related Lists Collections

Symbol

System Object