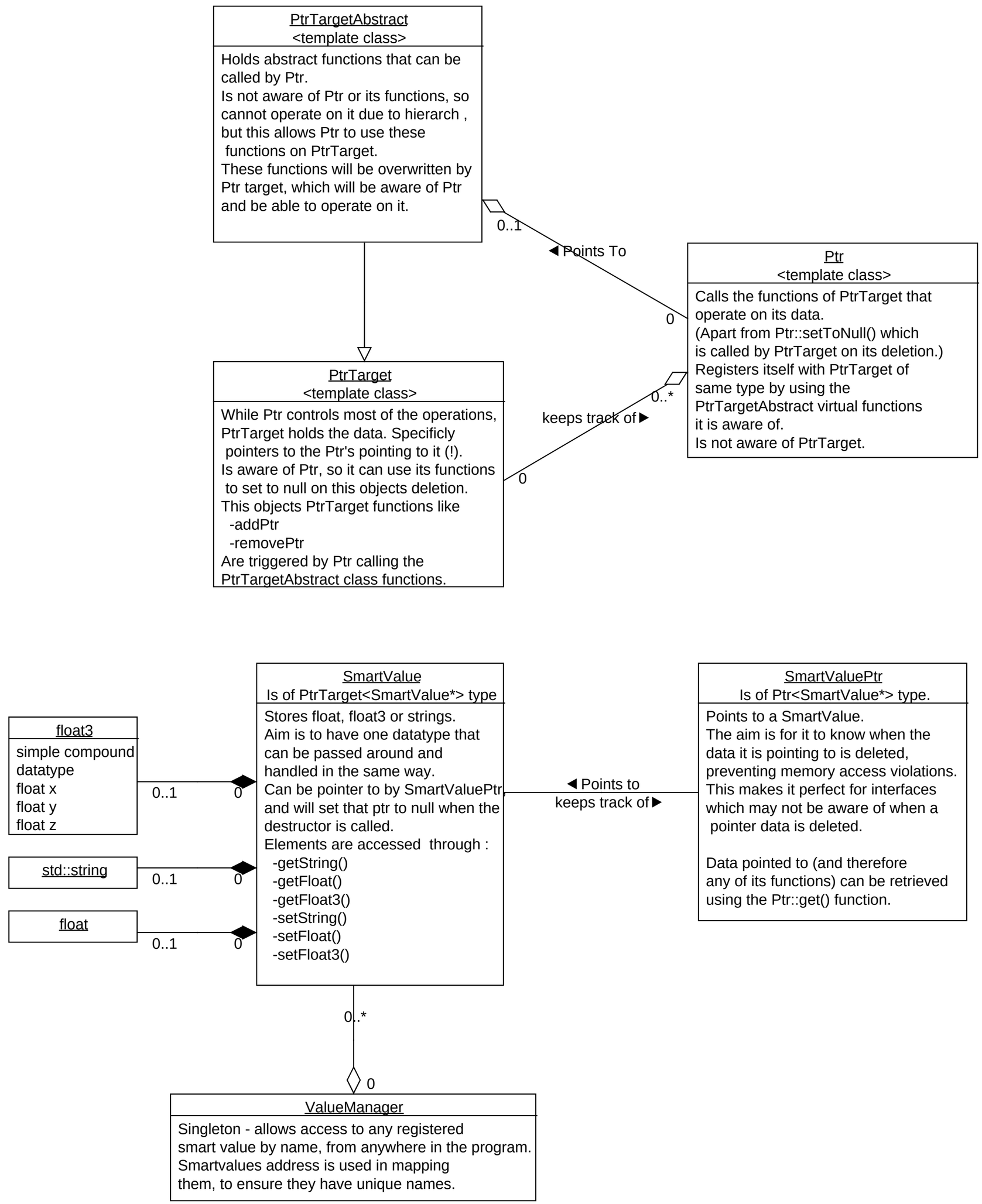
PtrTargetAbstract
<template class>
Holds abstract functions that can be called by Ptr.
Is not aware of Ptr or its functions, so cannot operate on it due to hierarch , but this allows Ptr to use these functions on PtrTarget.
These functions will be overwritten by Ptr target, which will be aware of Ptr and be able to operate on it.
0..1
◀ Points To
0
Ptr
<template class>
Calls the functions of PtrTarget that operate on its data.
(Apart from Ptr::setToNull() which is called by PtrTarget on its deletion.)
Registers itself with PtrTarget of same type by using the PtrTargetAbstract virtual functions it is aware of.
Is not aware of PtrTarget.
PtrTarget
<template class>
While Ptr controls most of the operations, PtrTarget holds the data. Specificly pointers to the Ptr's pointing to it (!).
Is aware of Ptr, so it can use its functions to set to null on this objects deletion.
This objects PtrTarget functions like
-addPtr
-removePtr
Are triggered by Ptr calling the PtrTargetAbstract class functions.
0..*
keeps track of ▶
0
float3
simple compound datatype
float x
float y
float z
0..1
0
std::string
0..1
0
float
0..1
0
SmartValue
Is of PtrTarget<SmartValue*> type
Stores float, float3 or strings.
Aim is to have one datatype that can be passed around and handled in the same way.
Can be pointer to by SmartValuePtr, and will set that ptr to null when the destructor is called.
Elements are accessed through :
-getString()
-getFloat()
-getFloat3()
-setString()
-setFloat()
-setFloat3()
◀ Points to
keeps track of ▶
SmartValuePtr
Is of Ptr<SmartValue*> type.
Points to a SmartValue.
The aim is for it to know when the data it is pointing to is deleted, preventing memory access violations.
This makes it perfect for interfaces which may not be aware of when a pointer data is deleted.
Data pointed to (and therefore any of its functions) can be retrieved using the Ptr::get() function.
0..*
0
ValueManager
Singleton - allows access to any registered smart value by name, from anywhere in the program.
Smartvalues address is used in mapping them, to ensure they have unique names.