RAII. Smart Pointers – notes

*RAII:*

* Acquisition in constructor, deallocation in destructor
* Create a wrapper for your object using resource allocation:
  + allocation in constructor
  + deallocation in destructor
* Use the wrapper object (directly) wherever you need the object

*Smart pointers:*

* declared on the stack
* initialized with a raw pointer
* when it goes out of scope, its destructor is invoked
* 3 types:
  + std::unique\_ptr
    - It retains exclusive ownership of the object, it does not share it

A diagram of a computer program

Description automatically generated

* + - constructed with the *make\_unique()* function.
  + std::shared\_ptr
    - Several shared\_ptr objects may own the same object.
    - The owned object is deleted only when the last remaining owning shared\_ptr is destroyed or have given up ownership (has been reset).

A diagram of a computer generated object

Description automatically generated

* + - constructed with the make shared function.
  + std::weak\_ptr
    - Used to access the underlying object of a shared\_ptr without causing the reference count to be incremented.
    - used to avoid dependency cycles:
      * class Team holds shared\_ptr to Members
      * if Members want to hold pointers to the Team, then they must use weak\_ptr
      * otherwise, when Team goes out of scope, it produces memory leak

A diagram of a computer

Description automatically generated