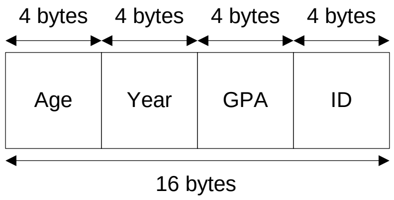
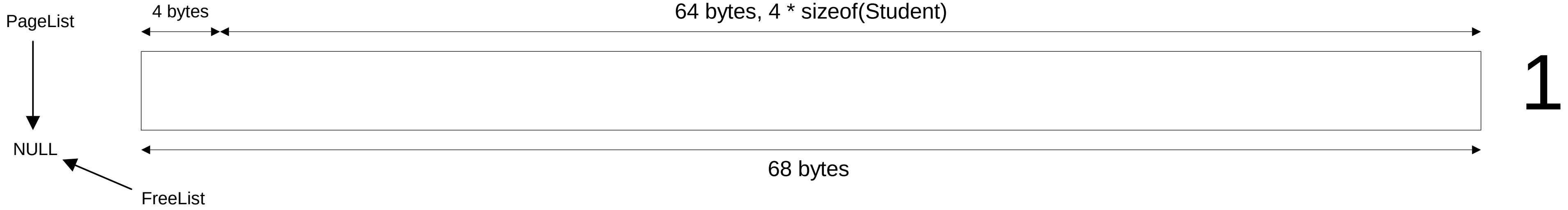
```
struct Student{
  int Age;
  long Year;
  float GPA;
  long ID;
};
```



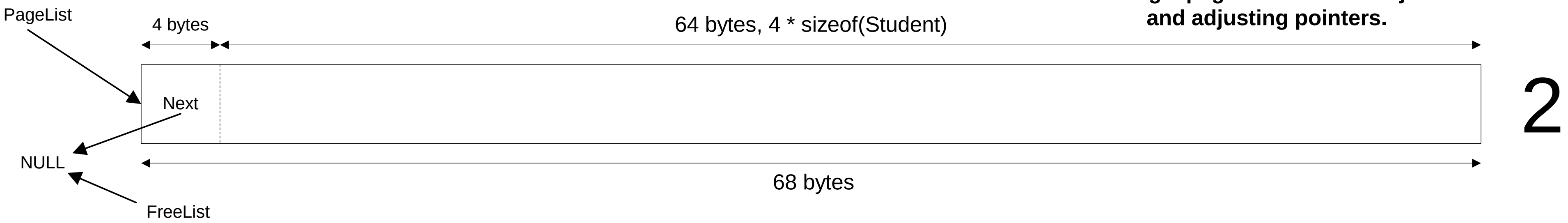
```
// Assume max_pages = 2, objects_per_page = 4 (8 objects total)
studentObjectMgr = new ObjectAllocator(sizeof(Student), config);
```

```
struct GenericObject {
  GenericObject *Next;
};
```

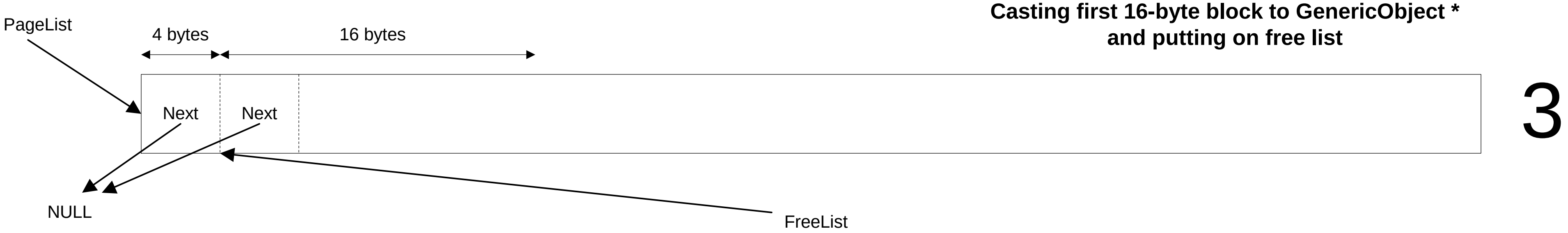
Allocating the first page



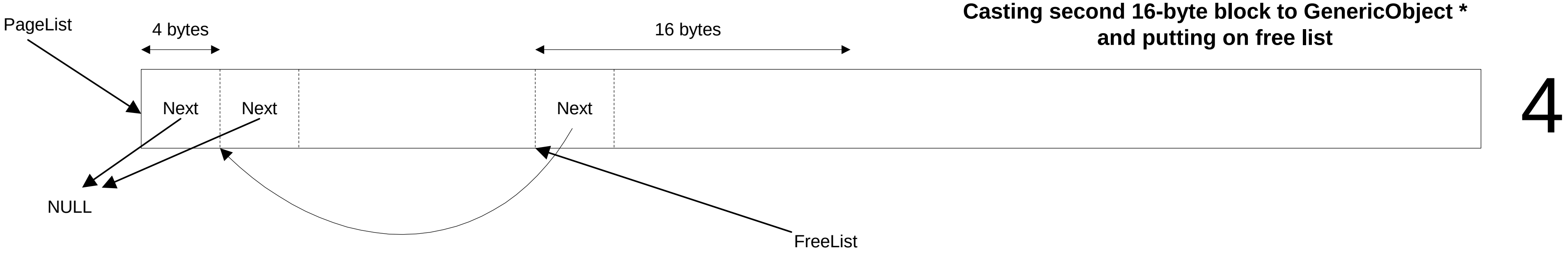
Casting a page to a GenericObject *
and adjusting pointers.



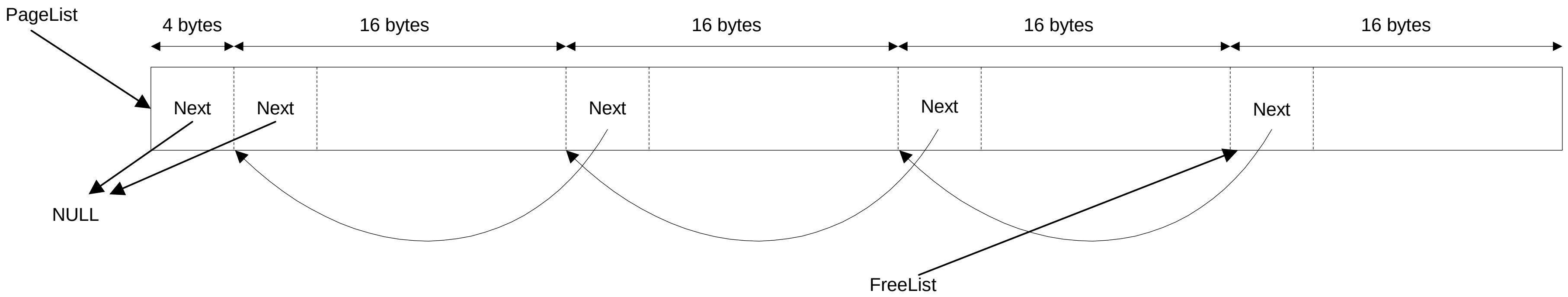
Casting first 16-byte block to GenericObject *
and putting on free list



Casting second 16-byte block to GenericObject *
and putting on free list

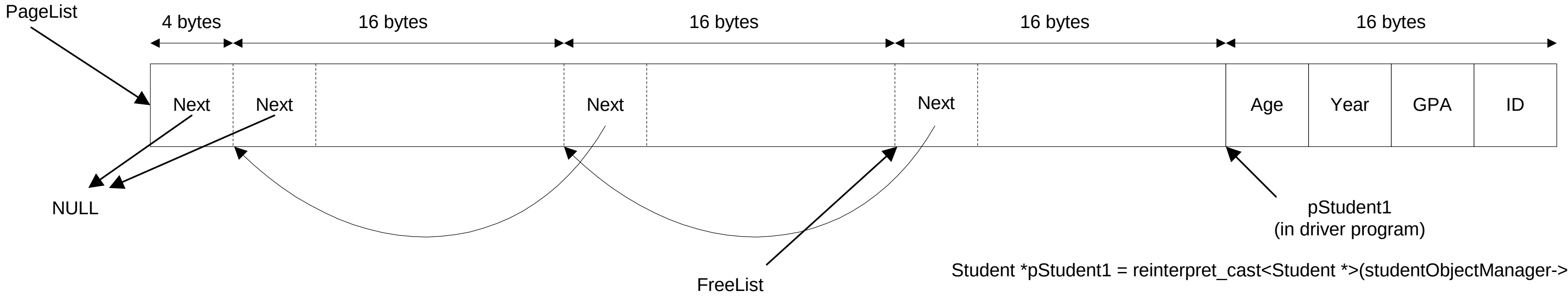


All new objects on the free list



5

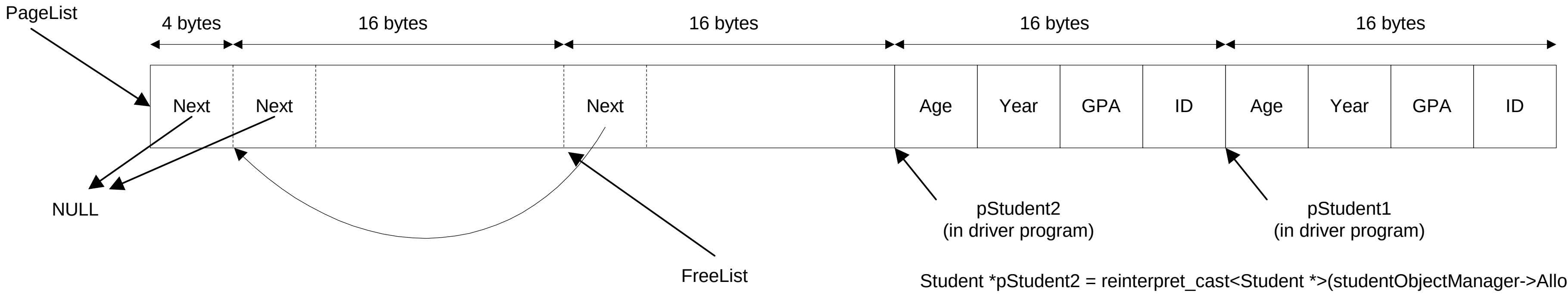
Remove first object from free list for client



6

Student *pStudent1 = reinterpret_cast<Student *>(studentObjectManager->Allocate());

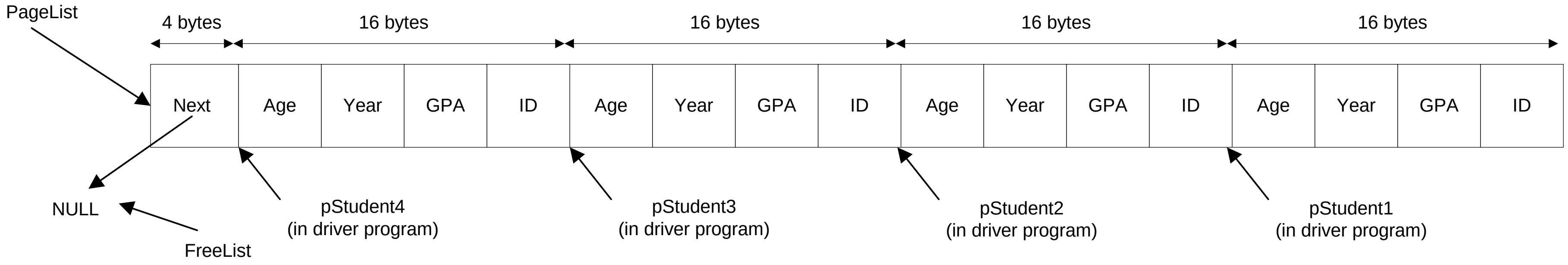
Remove second object from free list for client



7

Student *pStudent2 = reinterpret_cast<Student *>(studentObjectManager->Allocate());

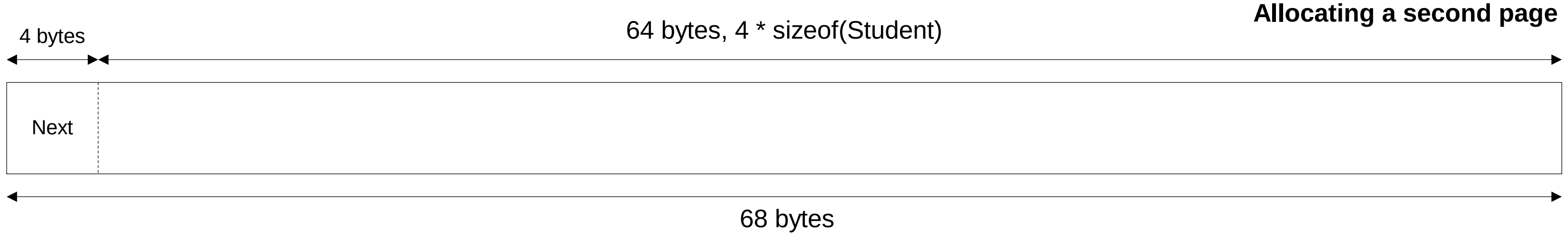
All objects removed from the free list.
Client has all of them.



8

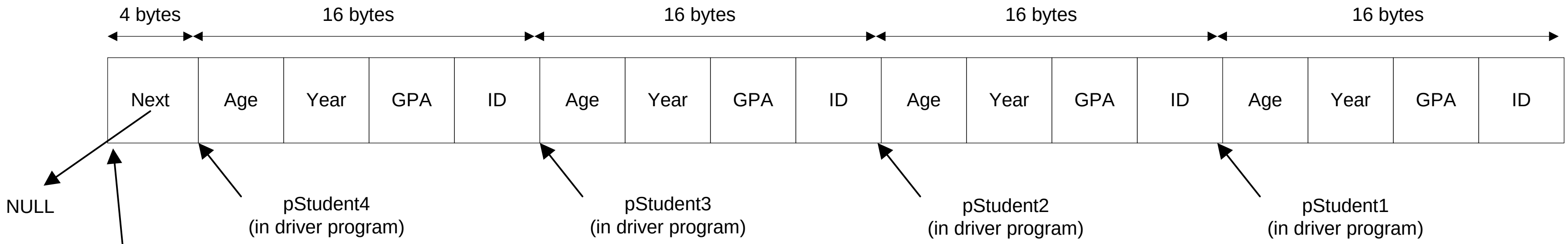
At this point, the client requests another object, but there are none left so we have to allocated another page first.

Allocating a second page



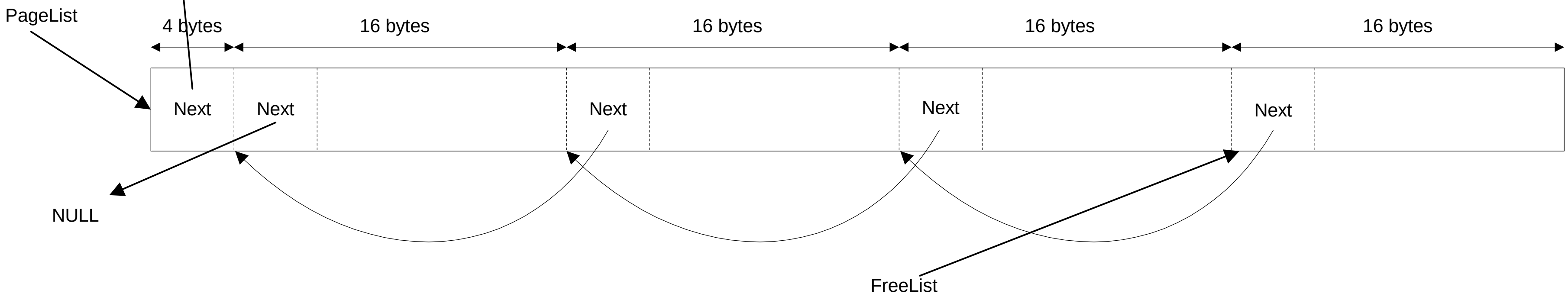
9

All objects removed from the free list



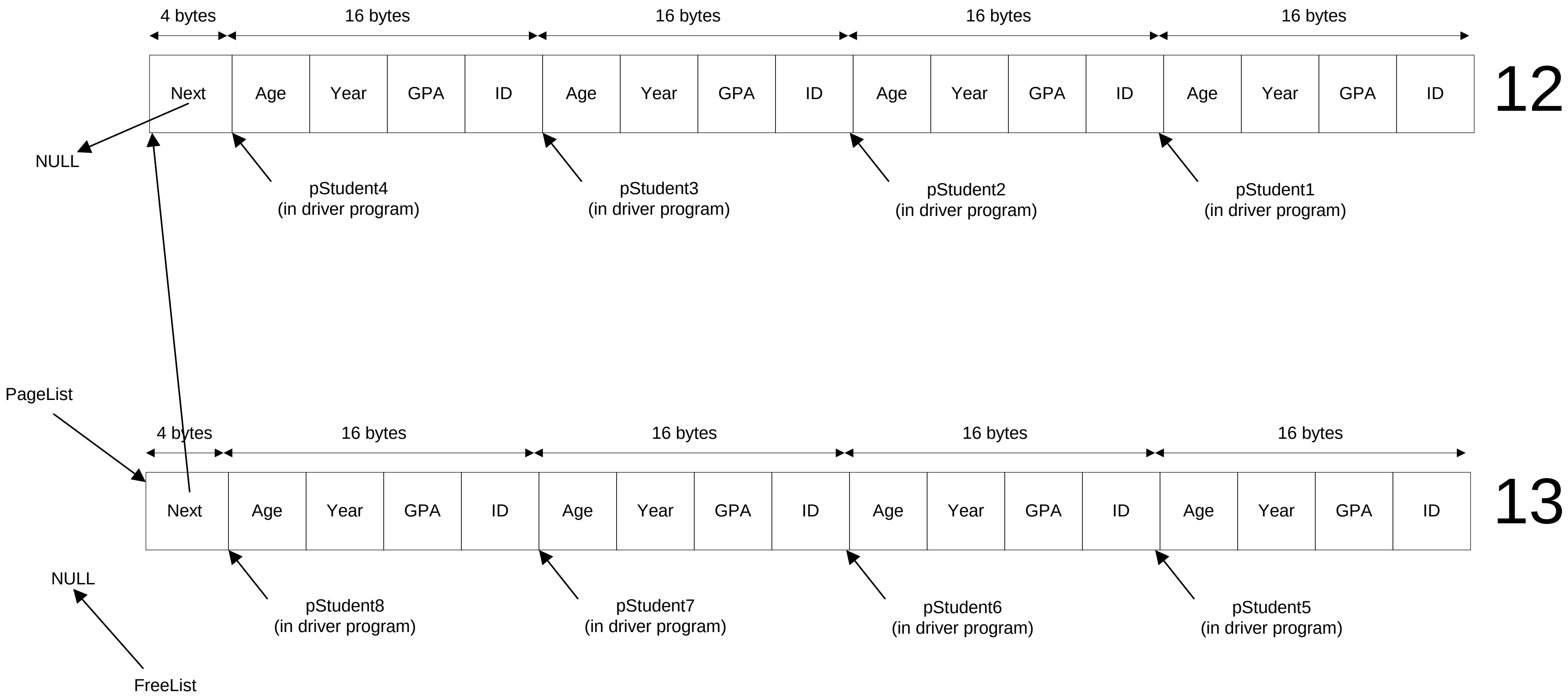
10

All new objects on the free list



11

Both pages have supplied objects to the client. No more objects on the free list.

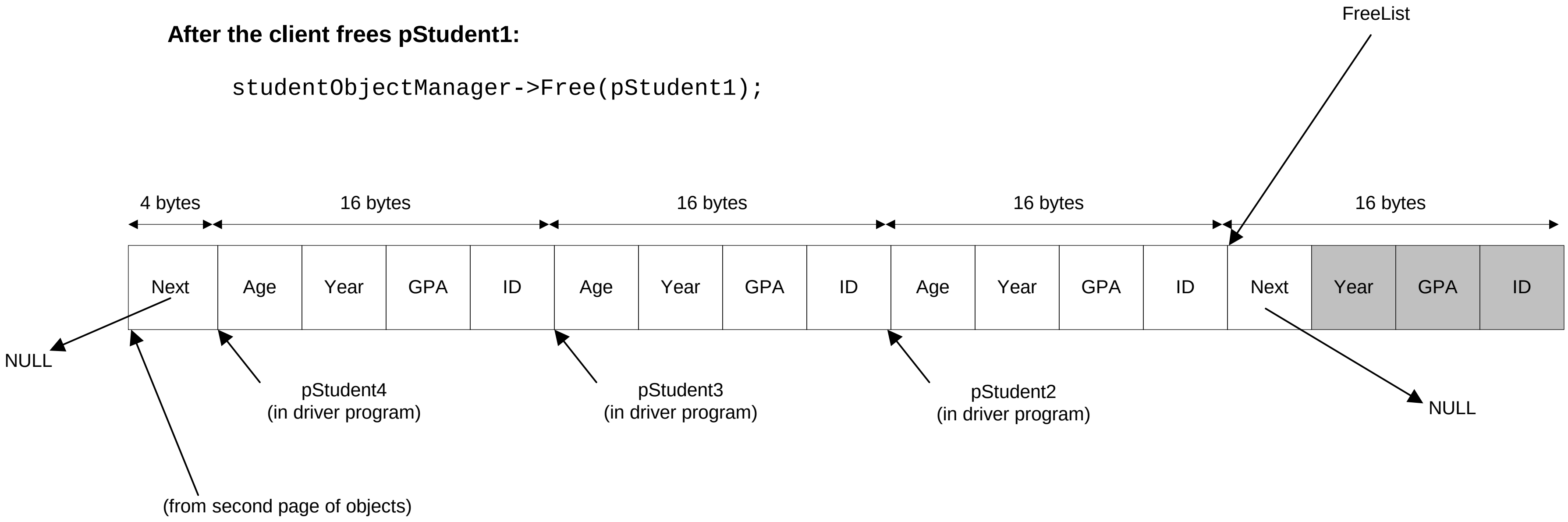


```
// this will only allow a maximum of 8 objects for the client
studentObjectMgr = new ObjectAllocator(sizeof(Student), config);
```

Given the parameters used to construct an instance of the ObjectAllocator, any more requests from the client will result in an exception being thrown. (There isn't any more memory available to the ObjectAllocator.)

After the client frees pStudent1:

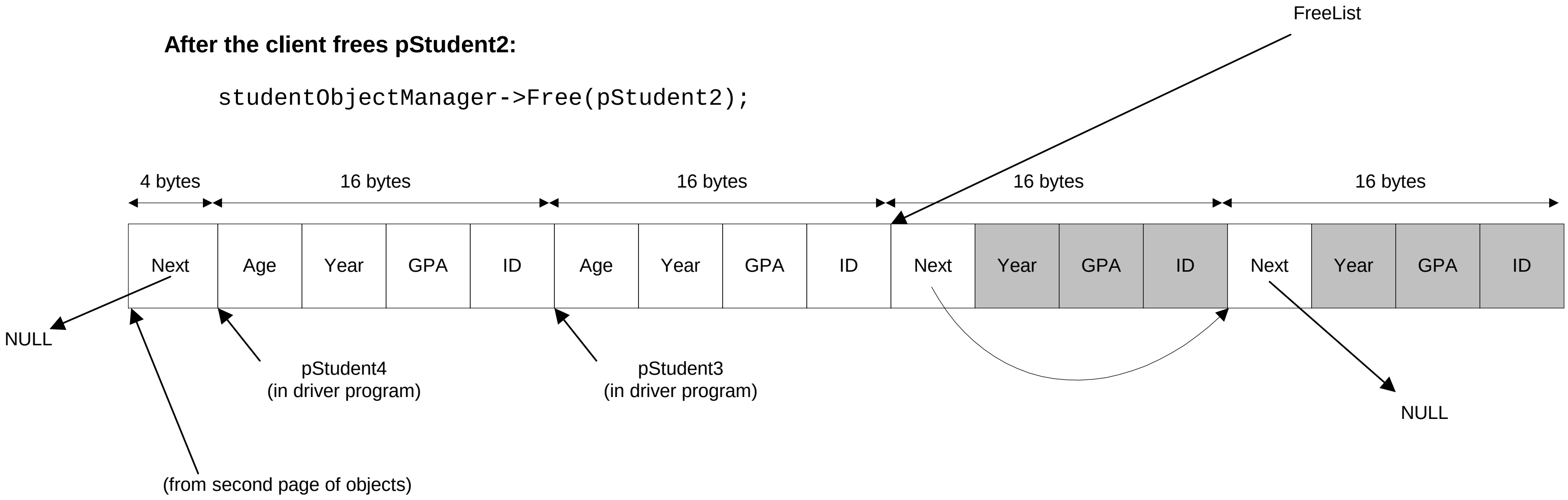
```
studentObjectManager->Free(pStudent1);
```



14

After the client frees pStudent2:

```
studentObjectManager->Free(pStudent2);
```

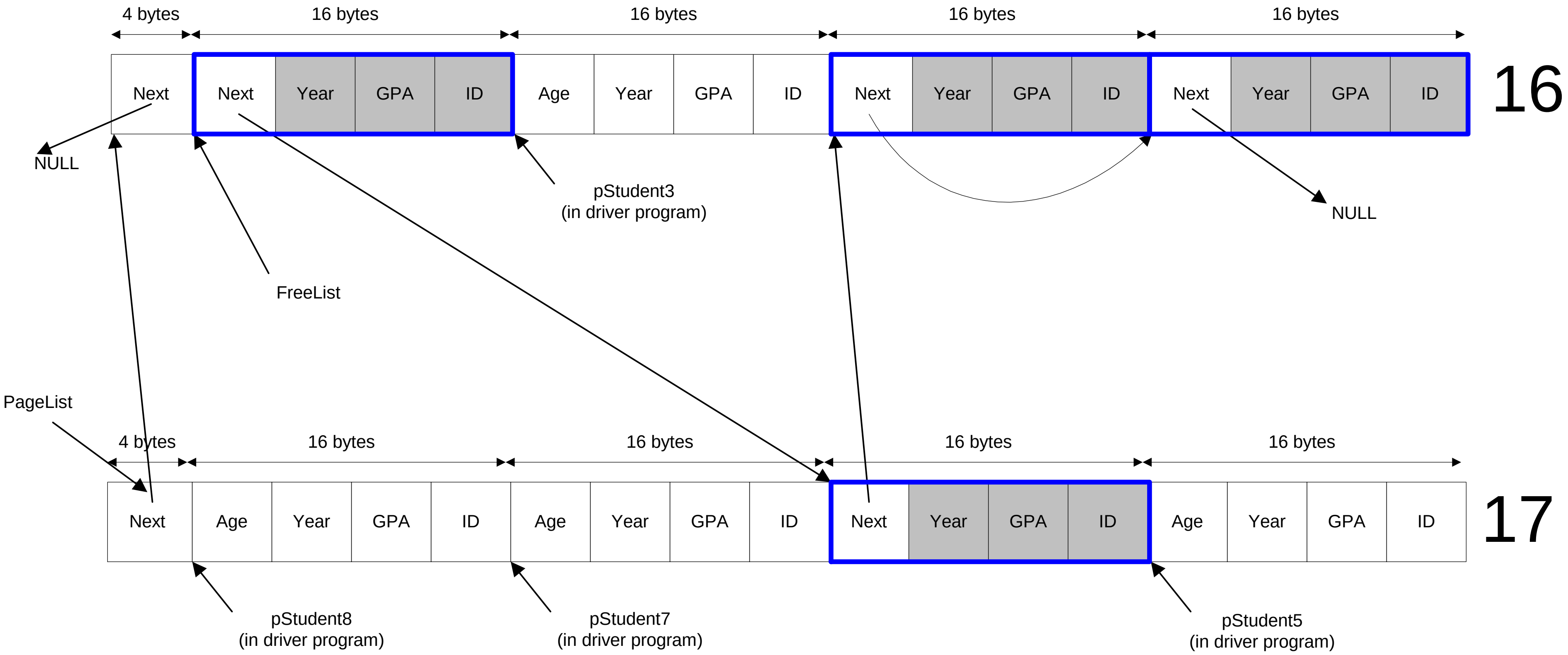


15

After the client frees more objects. Assume the order of execution was like this:

```
studentObjectManager->Free(pStudent1);
studentObjectManager->Free(pStudent2);
studentObjectManager->Free(pStudent6);
studentObjectManager->Free(pStudent4);
```

□ Indicates blocks on the free list



4 bytes
16 bytes
16 bytes
16 bytes
16 bytes
Next
Next
Year
GPA
ID
Age
Year
GPA
ID
Next
Year
GPA
ID
Next
Year
GPA
ID
16
NULL
pStudent3
(in driver program)
NULL
FreeList
PageList
4 bytes
16 bytes
16 bytes
16 bytes
16 bytes
Next
Age
Year
GPA
ID
Age
Year
GPA
ID
Next
Year
GPA
ID
Age
Year
GPA
ID
17
pStudent8
(in driver program)
pStudent7
(in driver program)
pStudent5
(in driver program)