```
1
2
3       typedef double XFLOAT
4       typedef double OTA_FLOAT
5
6   using  std
7
8
9   {
10
11   SQFUNCS_POLQA_INTERNAL
12  {
13
14  SQStorage::SQStorage()
15  {
16  }
17
18  SQStorageSkeletonClass* SQStorage::Get(string const &id)
19  {
20      for (int i = 0  i < (int)lItems.size()  i++)
21          if (lItems[i].name == id)
22              return lItems[i].ptrData
23      return NULL
24  }
25
26  void SQStorage::Store(string const &id, SQStorageSkeletonClass* data)
27  {
28      if (HasData(id))
29          OPTTHROW(( string("ERROR in SQStorage::Store: Data with this ID already present in the
    SQStorage singleton.\n")))
30      lItems.push_back(SQStorageItem(id, data))
31  }
32
33  void SQStorage::Clear(string const &id)
34  {
35      bool success = false
36      vector<struct SQStorageItem>::iterator curItem = lItems.begin()
37      while(curItem != lItems.end())
38      {
39          if (curItem->name == id)
40          {
41              delete curItem->ptrData
42              curItem->ptrData = NULL
43              lItems.erase(curItem)
44              success = true
45              break
46          }
47          curItem++
48      }
49      if (!success)
50          OPTTHROW(( string("ERROR in SQStorage::Clear: Item \"" + id + "\" could not be found in the
    storage.\n")))
51
52  }
53
54  void SQStorage::ClearAllItemsWhoseIDStartsWith(string const &id)
55  {
56      int j
57      std::vector<struct SQStorageItem>::iterator it
58
59      for (int i = 0  i < (int)lItems.size()  i++)
60      {
61          if (lItems[i].name.size() >= id.size() &&
62              lItems[i].name.compare(0, id.size(), id) == 0)
63          {
64              delete lItems[i].ptrData
65              lItems[i].ptrData = NULL
66              for (it = lItems.begin(), j = 0
```

```
                it != lItems.end() && j != i
                j++, it++)
            if (it == lItems.end())
                OPTTHROW ((string("Internal error in
SQStorage::ClearAllItemsWhoseIDStartsWith!\n")))

            lItems.erase(it)
            i--
        }
    }

}

void SQStorage::ClearAll()
{
    for (int i = 0  i < (int)lItems.size()  i++)
    {
        delete lItems[i].ptrData
        lItems[i].ptrData = NULL
    }
    lItems.clear()
}

bool SQStorage::HasData(string const &id)
{
    for (int i = 0  i < (int)lItems.size()  i++)
        if (lItems[i].name == id)
            return true
    return false
}

}

}
```