Define an abstract class **Box** with a text label, a number, a method providing the number of Boxes available and the pure virtual method getCapacity computing the capacity (volume) of a box.
Define the following classes inherited from the Box:

- **BrickBox** representing the Box being a cuboid with a length, a width and a height,
- **TubeBox** representing the Box being a cylinder with a base radius and a height,
- **BubbleBox** representing the Box being a hemisphere with a radius.

Override, for each of the above classes, the virtual method getCapacity, making it to return the volume of a box of the given class.
Implement all the constructors, destructors, getters, setters and exceptions which make the functionality of the classes complete, and all the other methods and exceptions necessary to run the code provided below.

Define the class **Store** with a description, a total capacity and a dynamic list of the boxes stored. Implement the following public methods of the class:

- a one enabling to add a new box of an arbitrary type to the store on the first position of the list (with the control of the total capacity of the store, CapacityError should be thrown if the total capacity could be exceeded),
- a one enabling to add a new box of an arbitrary type to the store on the last position of the list (with the control of the total capacity, too),
- a one enabling to remove the first box from the store (throwing the EmptyError exception if the list is empty),
- a one enabling to remove all the boxes from the list,
- a method returning the summary volume of all the boxes stored in the store.

Overload the indexing operator ([]) for the Store to have a direct access to the item on the particular position in the store list (throwing the IndexError exception if it doesn't exist). Overload the shift-left operator (<<) printing the data of the store and the details of all the boxes stored. Add all the other members which are necessary to make the functionality of the class complete or are required to run the code below.
Write a program which tests all the class capabilities, in particular the following code should be enabled:

```
Store store("My home store", 100);
cout << Box::count(); //0
try {
  store.addFirst(new BrickBox("books", 1234, 5, 3.5, 4)); //5 x 3.5 x 4
  store.addFirst(new TubeBox("posters", 2234, 1, 3)); //radius=1, height=3
  store.addLast(new BubbleBox("sweets", 5413, 1)); //radius=1
  store.addLast(new BrickBox("jewels", 2114, 2, 2, 2)); //2 x 2 x 2
  store.addFirst(new BrickBox("trinkets", 3456, 3, 4, 1)); //3 x 4 x 1
} catch(Store::CapacityError &e) {
  cout << e.what() << endl; //trinkets too large - only 10.5 free space
}

cout << store;
//My home store, total capacity: 100.0, free space: 10.5:
//1. posters, 2234, volume: 9.4
//2. books, 1234, volume: 70.0
//3. sweets, 5413, volume: 2.1
//4. jewels, 2114, volume: 8.0

cout << Box::count() << endl; //4
cout << store.summaryVolume() << endl; //89.5
store.removeFirst();
cout << Box::count() << endl; //3
cout << store.summaryVolume() << endl; //80.1

try {
  cout << store[1].getCapacity() << endl; //70.0
  cout << store[5].getCapacity() << endl; //IndexError exception
} catch(Store::IndexError &e) {
  cout << e.what(); //item no. 5 not found
}
store.clear();
cout << Box::count(); //0
```