A shared memory mechanism - the store

Sometimes we need a shared memory space. It's that simple. True, we need to be careful about it, but let's admit - it is super useful. This sore mechanism was inspired by a similar mechanism used by JUnit 5.

There are 2 types of stores - a local per-thread store and a global store. It's pretty much self-explanatory.

Putting value in the store is done like this:

```
StoreManager.getStore(StoreType.LOCAL_THREAD).putValueInStore("key", "val1");
```

2 things to note here -

a. The store type is a local thread type, and only be visible to objects within the same thread.

b. The store is a key-value mechanism. The key can be any type of object, and so is the value.

Extracting value from the store is done like this:

```
String receivedName =
            StoreManager.getStore(StoreType.LOCAL_THREAD).getValueFromStore("key");
```

Note that even though the value can be any type if you know what type you are looking for, there is no need to do any casting.

If you want to override the value in the store, simply assign a new value with the same key, and the value will be updated.

The Global Store

In case you need a shared memory space for all threads, you may use the same mechanism with the `StoreType.LOCAL_THREAD` indication on. It is done in the following way:

```
StoreManager.getStore(StoreType.GLOBAL).putValueInStore("key", "val1");
String receivedName =
            StoreManager.getStore(StoreType.GLOBAL).getValueFromStore("key");
```

The Unique store identifier

Another option is to provide the store with a specific name. It is done as follows:

```
StoreManager.getStore("unique"").putValueInStore("key", "val1");
String receivedName =
            StoreManager.getStore("unique").getValueFromStore("key");
```