# Lab CC 5: Shared Objects

## Objectives

In this lab, you will create several versions of an object. In one version, different threads have their own copies, and in the second version, different threads share a single copy of the object

## Part One: Non-shared Objects

### Step 1 Create the MyObject class

1. Text

   Description automatically generatedThis object is not a thread, it’s just an arbitrary object that occupies memory on the heap when instantiated.
2. Graphical user interface, text, application, email

   Description automatically generatedCreate the runnable Task class that will use MyObject. In this first version, each Task object creates its own MyObject instance. These objects are not shared across threads
3. Note that the output statement prints the name of Task object and the address of the MyObject object.
4. Text

   Description automatically generatedNow create the runner class which creates and starts the threads
5. The important thing to note is that two objects are at different memory addresses which means they are distinct objects.

## Part Two: Shared Objects

1. Recall that the main thread is also a thread so it shares heap memory with the other threads spawned from it.
2. This section uses the same definition of the MyObject class as before.
3. However, there is one change to the Task class. Instead of each Task object creating its own copy of the MyObject, the main thread will create the object and pass a reference to the MyObject to each of the Task objects.
4. This is an example of Constructor dependency injection.

Graphical user interface, text, application, email

Description automatically generated

1. Now when the threads run, we can see by the addresses that they are both sharing the same MyObject instance.

Text

Description automatically generated