**Title: Implementation of Reader writer problem**

**Aim**: To implement Reader Writer problem using semaphores in Java.

**Theory**:

* There is a data area shared among a number of processor registers.
* The data area could be a file, a block of main memory, or even a bank of processor registers.
* There are a number of processes that only read the data area (readers) and a number that only write to the data area (writers).
* The conditions that must be satisfied are
  + Any number of readers may read simultaneously read the file.
  + Only one write at a time may write to the file.
  + If a writer is writing to the file, no reader may read it.

**Semaphore:**

**Definition :** Semaphores are system variables used for synchronization of process

**Two types of Semaphore:**

* **Counting semaphore –** integer value can range over an unrestricted domain
* **Binary semaphore –** 
  + Integer value can range only between 0 and 1; can be simpler to implement
  + Also known as mutex locks

**Semaphore functions:**

**Package : import java.util.concurrent.Semaphore;**

1. **To initialize a semaphore :**

**Semaphore Sem1 = new Semaphore(1);**

1. **To wait on a semaphore:**

**Sem1.acquire();**

1. **To signal on a semaphore:**

**mutex.release();**

**Algorithm for Reader Writer:**

1. **import java.util.concurrent.Semaphore;**
2. **Create a class RW**
3. **Declare semaphores – mutex and wrt**
4. **Declare integer variable readcount = 0**
5. **Create a nested class Reader implements Runnable**
   1. **Override run method (Reader Logic)**
      1. wait(mutex);
      2. readcount := readcount +1;
      3. if readcount = 1 then
      4. wait(wrt);
      5. signal(mutex);
      6. …
      7. reading is performed
      8. …
      9. wait(mutex);
      10. readcount := readcount – 1;
      11. if readcount = 0 then signal(wrt);
      12. signal(mutex):
6. **Create a nested class Writer implements Runnable**
   1. **Override run method (Writer Logic)**
      1. wait(wrt);
      2. …
      3. writing is performed
      4. …
      5. signal(wrt);
7. **Create a class main**
   1. **Create Threads for Reader and Writer**
   2. **Start these thread**