Class Task 3

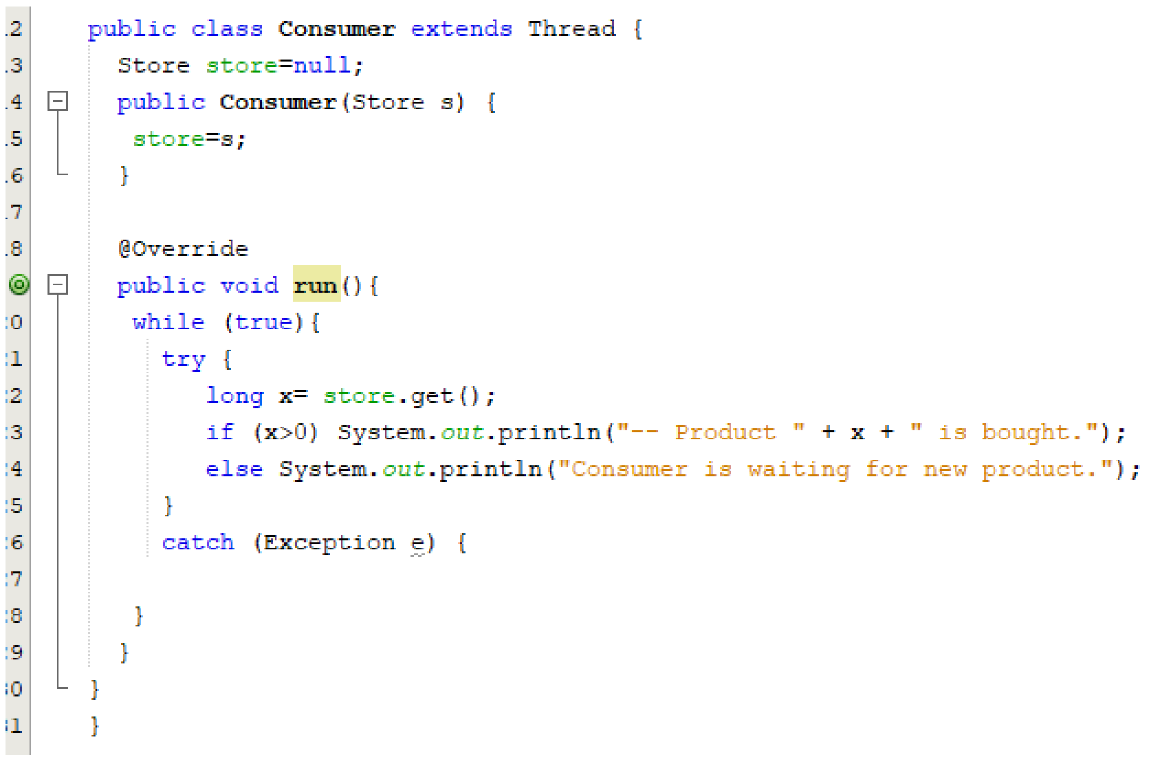
1. Write a program that will implement the following logic:
   - There is an int variable ‘counter’ which is set to 0
   - There are two threads, A & B
   - Two methods
     - Increase () for A
     - Decrease () for B
   - Thread A increases the value of first element of the array by one.
   - At the same time thread B is decreasing the value of first element of the array by one.
   - Both threads are executing these operations 1000 times.

   - So ideally the final result should be 0 since thread A increases the value 1000 times and thread B decreases the value 1000 times.

   - Synchronize the execution of these two threads

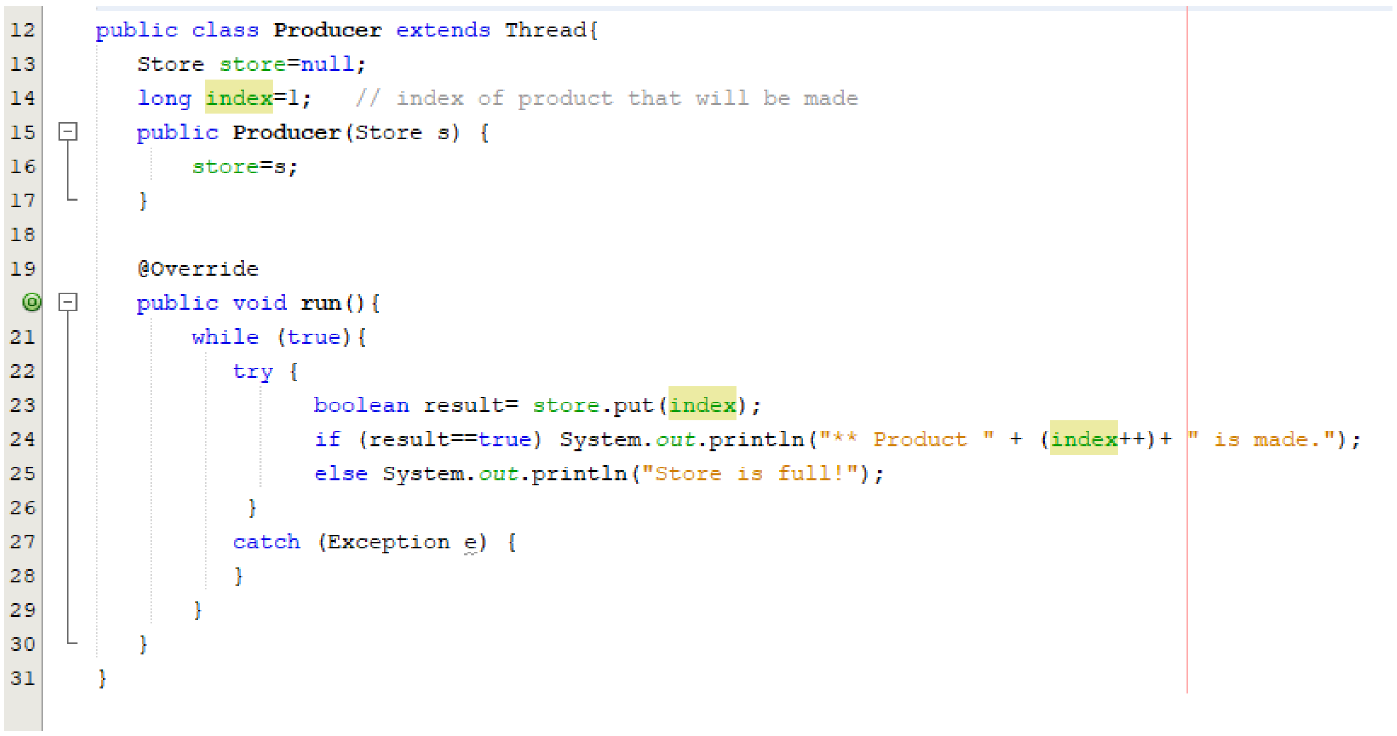
2. Finish the following partial codes to demonstrate Consumer-Producer problem
   - There are four classes (Store, Consumer, Producer, ProducerConsumerProblem)

- **Class Consumer**



```java
public class Consumer extends Thread {
  Store store=null;
  public Consumer(Store s) {
   store=s;
  }

  @Override
  public void run(){
   while (true){
     try {
        long x= store.get();
        if (x>0) System.out.println("-- Product " + x + " is bought.");
        else System.out.println("Consumer is waiting for new product.");
     }
     catch (Exception e) {

   }
  }
 }
}
```

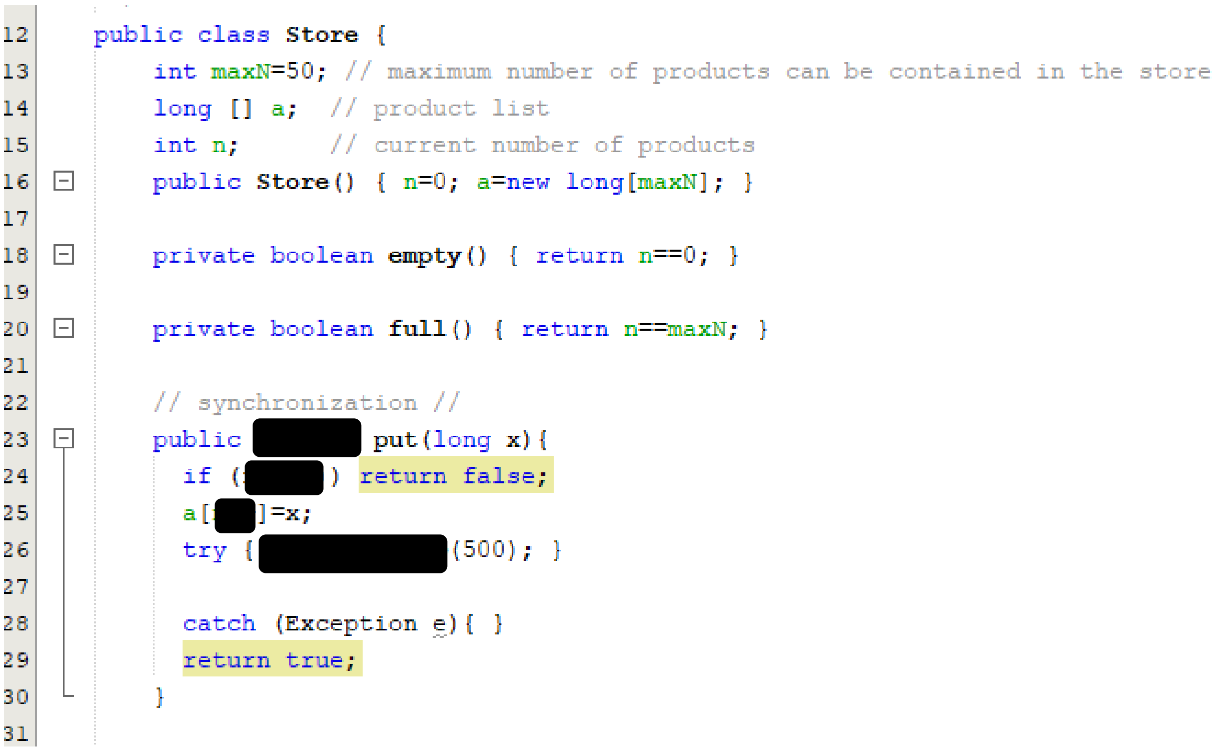
- **Class Producer**



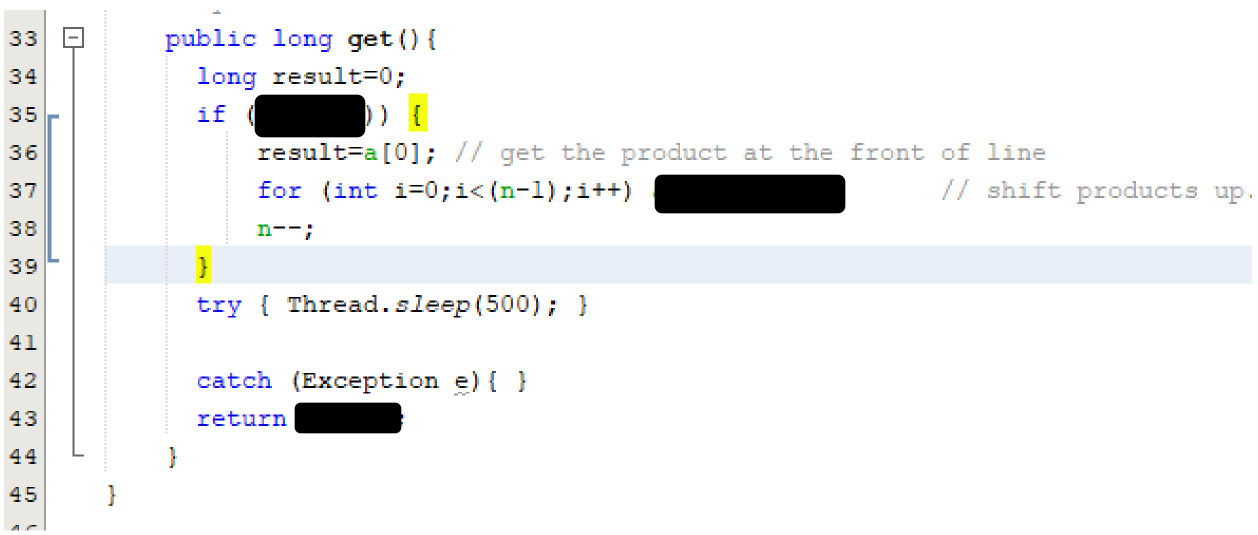
```java
public class Producer extends Thread{
    Store store=null;
    long index=1;    // index of product that will be made
    public Producer(Store s) {
        store=s;
    }

    @Override
    public void run(){
        while (true){
            try {
                boolean result= store.put(index);
                if (result==true) System.out.println("** Product " + (index++)+ " is made.");
                else System.out.println("Store is full!");
            }
            catch (Exception e) {
            }
        }
    }
}
```

Tasks:

1. Complete the **Store** class

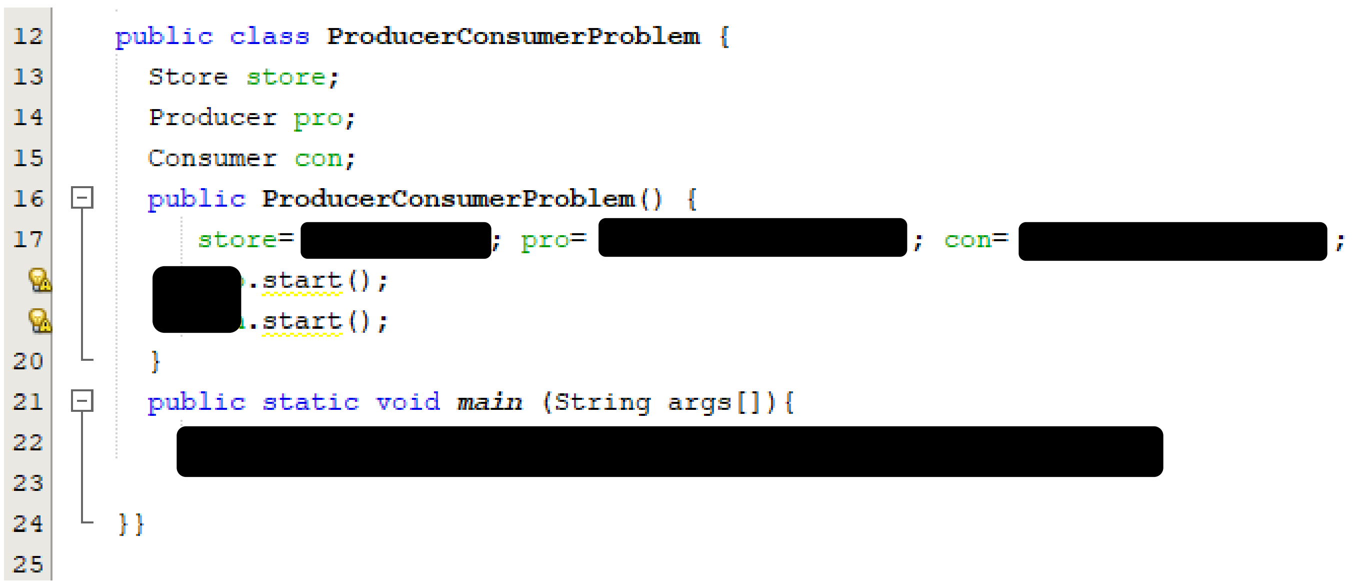


```java
public class Store {
    int maxN=50; // maximum number of products can be contained in the store
    long [] a;  // product list
    int n;      // current number of products
    public Store() { n=0; a=new long[maxN]; }

    private boolean empty() { return n==0; }

    private boolean full() { return n==maxN; }

    // synchronization //
    public [redacted] put(long x){
      if ([redacted]) return false;
      a[[redacted]]=x;
      try {[redacted](500); }

      catch (Exception e){ }
      return true;
    }
```



```
    public long get(){
      long result=0;
      if ([redacted])) {
         result=a[0]; // get the product at the front of line
         for (int i=0;i<(n-1);i++) [redacted]       // shift products up.
         n--;
      }
      try { Thread.sleep(500); }

      catch (Exception e){ }
      return [redacted];
    }
}
```

2. Complete the **ProducerConsumerProblem** class.



```
public class ProducerConsumerProblem {
  Store store;
  Producer pro;
  Consumer con;
  public ProducerConsumerProblem() {
     store=[redacted]; pro=[redacted]; con=[redacted];
  [redacted].start();
          .start();
  }
  public static void main (String args[]){
    [redacted]

}}
```