Programs on Concurrency Utilities in Java

```
// A simple semaphore example showing usage of acquire () and release ().
import java.util.concurrent.*;
class SemDemo {
	public static void main(String args[]) {
		Semaphore sem = new Semaphore(1);
		new IncThread (sem, "A");
		new DecThread (sem, "B");
	}
}
// A shared resource.
class Shared {
		static int count = 0;
}

// A thread of execution that increments count.
class IncThread implements Runnable {
	String name;
	Semaphore sem;
	IncThread(Semaphore s, String n) {
		sem = s;
		name = n;
		new Thread(this).start();
	}
	public void run() {
		System.out.println("Starting " + name);
		try {
			// First, get a permit.
			System.out.println(name + " is waiting for a permit.");
			sem.acquire();
			System.out.println(name + " gets a permit.");
			// Now, access shared resource.
			for(int i=0; i < 5; i++) {
				Shared.count++;
				System.out.println(name + ": " + Shared.count);
				// Now, allow a context switch -- if possible.
				Thread.sleep(10);
			}
		} catch (InterruptedException exc) {
			System.out.println(exc);
		}
		// Release the permit.
		System.out.println(name + " releases the permit.");
		sem.release();
	}
}
```

```java
// A thread of execution that decrements count.
class DecThread implements Runnable {
	String name;
	Semaphore sem;
	DecThread(Semaphore s, String n) {
		sem = s;
		name = n;
		new Thread(this).start();
	}
	public void run() {
		System.out.println("Starting " + name);
		try {
			// First, get a permit.
			System.out.println(name + " is waiting for a permit.");
			sem.acquire();
			System.out.println(name + " gets a permit.");

			// Now, access shared resource.
			for(int i=0; i < 5; i++) {
				Shared.count--;
				System.out.println(name + ": " + Shared.count);
				// Now, allow a context switch -- if possible.
				Thread.sleep(10);
			}
		} catch (InterruptedException exc) {
			System.out.println(exc);
		}
		// Release the permit.
		System.out.println(name + " releases the permit.");
		sem.release();
	}
}
```

Problems Javadoc Declaration Console Console

```
<terminated> SemDemo [Java Application] C:\Program Files (x86)\Java\jre6\bin\javaw.exe (19-Oct-2020 8:42:06 AM)
Starting A
A is waiting for a permit.
A gets a permit.
Starting B
B is waiting for a permit.
A: 1
A: 2
A: 3
A: 4
A: 5
A releases the permit.
B gets a permit.
B: 4
B: 3
B: 2
B: 1
B: 0
B releases the permit.
```

```
// An implementation of a producer and consumer that use semaphores to control
// synchronization.

import java.util.concurrent.Semaphore;

class Q {
	int n;
	// Start with consumer semaphore unavailable.
	static Semaphore semCon = new Semaphore(0);
	static Semaphore semProd = new Semaphore(1);

	void get() {
		semProd.release();
		try {
			semCon.acquire();
		} catch(InterruptedException e) {
			System.out.println("InterruptedException caught");
		}
		System.out.println("Got: " + n);
	}

	void put(int n) {
		semCon.release();
		try {
			semProd.acquire();
		} catch(InterruptedException e) {
			System.out.println("InterruptedException caught");
		}
		this.n = n;
		System.out.println("Put: " + n);
	}
}

class Producer implements Runnable {
	Q q;
	Producer(Q q) {
		this.q = q;
		new Thread(this, "Producer").start();
	}

	public void run() {
		for(int i=0; i < 20; i++) q.put(i);
	}
}
```

```
class Consumer implements Runnable {
        Q q;
        Consumer(Q q) {
                this.q = q;
                new Thread(this, "Consumer").start();
        }

        public void run() {
                for(int i=0; i < 20; i++) q.get();
        }
}

class ProdCon {
        public static void main(String args[]) {
                Q q = new Q();
                new Consumer(q);
                new Producer(q);
        }
}
```

```
Problems  @ Javadoc  Declaration  Console  Console
<terminated> ProdCon [Java Application] C:\Program Files (x86)\Java\jre6\bin\javaw.exe (19-Oct-2020 8:49:35 AM)
Put: 0
Got: 0
Put: 1
Put: 2
Got: 1
Got: 2
Put: 3
Got: 3
Put: 4
Got: 4
Put: 5
Got: 5
Put: 6
Got: 6
Put: 7
Got: 7
Put: 8
Got: 8
Put: 9
Got: 9
```

// An example of CountDownLatch.

```
import java.util.concurrent.CountDownLatch;

class CDLDemo {
	public static void main(String args[]) {
		CountDownLatch cdl = new CountDownLatch(5);
		System.out.println("Starting");

		new MyThread(cdl);
		try {
			cdl.await();
		} catch (InterruptedException exc) {
			System.out.println(exc);
		}
		System.out.println("Done");
	}
}

class MyThread implements Runnable {
	CountDownLatch latch;
	MyThread(CountDownLatch c) {
		latch = c;
		new Thread(this).start();
	}

	public void run() {
		for(int i = 0; i<5; i++) {
			System.out.println(i);
			latch.countDown(); // decrement count
		}
	}
}
```

Problems @ Javadoc Declaration Console Console

```
<terminated> CDLDemo [Java Application] C:\Program Files (x86)\Java\jre6\bin\javaw.exe (19-Oct-2020 11:23:06 AM)
Starting
0
1
2
3
4
Done
```