Pdf and Codes

Introduction.pdf

Codes

1. **Performance Gains via Multi-Threading#**

```java
class Demonstration {
   public static void main( String args[] ) throws InterruptedException {
      SumUpExample.runTest();
   }
}

class SumUpExample {

   long startRange;
   long endRange;
   long counter = 0;
   static long MAX_NUM = Integer.MAX_VALUE;

   public SumUpExample(long startRange, long endRange) {
      this.startRange = startRange;
      this.endRange = endRange;
   }

   public void add() {

      for (long i = startRange; i <= endRange; i++) {
         counter += i;
      }
   }
```

```
static public void twoThreads() throws InterruptedException {

    long start = System.currentTimeMillis();
    SumUpExample s1 = new SumUpExample(1, MAX_NUM / 2);
    SumUpExample s2 = new SumUpExample(1 + (MAX_NUM / 2), MAX_NUM);

    Thread t1 = new Thread(() -> {
        s1.add();
    });

    Thread t2 = new Thread(() -> {
        s2.add();
    });

    t1.start();
    t2.start();

    t1.join();
    t2.join();

    long finalCount = s1.counter + s2.counter;
    long end = System.currentTimeMillis();
    System.out.println("Two threads final count = " + finalCount + " took " + (en
}

static public void oneThread() {

    long start = System.currentTimeMillis();
    SumUpExample s = new SumUpExample(1, MAX_NUM );
    s.add();
    long end = System.currentTimeMillis();
    System.out.println("Single thread final count = " + s.counter + " took " + (en
}


public static void runTest() throws InterruptedException {
        oneThread();
        twoThreads();

    }
}
```

Program vs Process vs Thread.pdf

1. **Thread unsafe class**

```
import java.util.Random;

class DemoThreadUnsafe {

    // We'll use this to randomly sleep our threads
    static Random random = new Random(System.currentTimeMillis());

    public static void main(String args[]) throws InterruptedException {

        // create object of unsafe counter
        ThreadUnsafeCounter badCounter = new ThreadUnsafeCounter();

        // setup thread1 to increment the badCounter 200 times
        Thread thread1 = new Thread(new Runnable() {

            @Override
            public void run() {
                for (int i = 0; i < 100; i++) {
                    badCounter.increment();
                    DemoThreadUnsafe.sleepRandomlyForLessThan10Secs();
                }
            }
        });

        // setup thread2 to decrement the badCounter 200 times
        Thread thread2 = new Thread(new Runnable() {

            @Override
            public void run() {
                for (int i = 0; i < 100; i++) {
                    badCounter.decrement();
                    DemoThreadUnsafe.sleepRandomlyForLessThan10Secs();
                }
            }
        });

        // run both threads
        thread1.start();
        thread2.start();

        // wait for t1 and t2 to complete.
        thread1.join();
        thread2.join();

        // print final value of counter
        badCounter.printFinalCounterValue();
    }

    public static void sleepRandomlyForLessThan10Secs() {
        try {
            Thread.sleep(random.nextInt(10));
        } catch (InterruptedException ie) {
        }
    }
}
```

```java
class ThreadUnsafeCounter {

    int count = 0;

    public void increment() {
        count++;
    }

    public void decrement() {
        count--;
    }

    void printFinalCounterValue() {
        System.out.println("counter is: " + count);
    }
}
```

3. **Concurrency vs Parallelism**

Concurrency vs Parallelism.pdf

4. **Cooperative Multitasking vs Preemptive Multitasking**

Cooperative Multitasking vs Preemptive Multitasking.pdf

5. **Synchronous vs Asynchronous**

Synchronous vs Asynchronous.pdf

6. **I/O Bound vs CPU Bound**

IO Bound vs CPU Bound.pdf

7. **Throughput vs Latency**

Throughput vs Latency.pdf

8. **Critical Sections & Race Conditions**

Critical Sections & Race Conditions.pdf

Codes:

1. **Example Thread Race#**

```
import java.util.*;

class Demonstration {

   public static void main(String args[]) throws InterruptedException {
        RaceCondition.runTest();
   }
}

class RaceCondition {

   int randInt;
   Random random = new Random(System.currentTimeMillis());

   void printer() {
        int i = 1000000;
        while (i != 0) {
            if (randInt % 5 == 0) {
                if (randInt % 5 != 0)
                  System.out.println(randInt);
            }
            i--;
        }
    }

    void modifier() {

        int i = 1000000;
        while (i != 0) {
            randInt = random.nextInt(1000);
            i--;
        }
    }

    public static void runTest() throws InterruptedException {


        final RaceCondition rc = new RaceCondition();
        Thread thread1 = new Thread(new Runnable() {

            @Override
            public void run() {
                rc.printer();
            }
        });
        Thread thread2 = new Thread(new Runnable() {

            @Override
            public void run() {
                rc.modifier();
            }
        });


        thread1.start();
        thread2.start();

        thread1.join();
        thread2.join();
    }
}
```

2.

```
import java.util.*;

class Demonstration {

    public static void main(String args[]) throws InterruptedException {
          RaceCondition.runTest();
    }
}

class RaceCondition {

    int randInt;
    Random random = new Random(System.currentTimeMillis());

    void printer() {

        int i = 1000000;
        while (i != 0) {
            synchronized(this) {
              if (randInt % 5 == 0) {
                  if (randInt % 5 != 0)
                    System.out.println(randInt);
              }
            }
            i--;
        }
    }

    void modifier() {

        int i = 1000000;
        while (i != 0) {
            synchronized(this) {
              randInt = random.nextInt(1000);
              i--;
            }
        }
    }

    public static void runTest() throws InterruptedException {



        final RaceCondition rc = new RaceCondition();
        Thread thread1 = new Thread(new Runnable() {

            @Override
            public void run() {
                rc.printer();
            }
        });
        Thread thread2 = new Thread(new Runnable() {

            @Override
            public void run() {
                rc.modifier();
            }
        });
```

```
        thread1.start();
        thread2.start();

        thread1.join();
        thread2.join();
    }
}
```

9. **Deadlocks, Liveness & Reentrant Locks**

Deadlocks, Liveness & Reentrant Locks.pdf

Codes

1. Example of a Deadlock

```
class Demonstration {

    public static void main(String args[]) {
        Deadlock deadlock = new Deadlock();
        try {
            deadlock.runTest();
        } catch (InterruptedException ie) {
        }
    }
}

class Deadlock {

    private int counter = 0;
    private Object lock1 = new Object();
    private Object lock2 = new Object();
```

```
Runnable incrementer = new Runnable() {

    @Override
    public void run() {
        try {
            for (int i = 0; i < 100; i++) {
                incrementCounter();
                System.out.println("Incrementing " + i);
            }
        } catch (InterruptedException ie) {
        }
    }
};

Runnable decrementer = new Runnable() {

    @Override
    public void run() {
        try {
            for (int i = 0; i < 100; i++) {
                decrementCounter();
                System.out.println("Decrementing " + i);
            }
        } catch (InterruptedException ie) {
        }

    }
};

public void runTest() throws InterruptedException {

    Thread thread1 = new Thread(incrementer);
    Thread thread2 = new Thread(decrementer);

    thread1.start();
```

```
        // sleep to make sure thread 1 gets a chance to acquire lock1
        Thread.sleep(100);
        thread2.start();

        thread1.join();
        thread2.join();

        System.out.println("Done : " + counter);
    }

    void incrementCounter() throws InterruptedException {
        synchronized (lock1) {
            System.out.println("Acquired lock1");
            Thread.sleep(100);

            synchronized (lock2) {
                counter++;
            }
        }
    }

    void decrementCounter() throws InterruptedException {
        synchronized (lock2) {
            System.out.println("Acquired lock2");

            Thread.sleep(100);
            synchronized (lock1) {
                counter--;
            }
        }
    }
}
```

2. Example of Deadlock with Non-Reentrant Lock

```
class Demonstration {

    public static void main(String args[]) throws Exception {
        NonReentrantLock nreLock = new NonReentrantLock();

        // First locking would be successful
        nreLock.lock();
        System.out.println("Acquired first lock");

        // Second locking results in a self deadlock
        System.out.println("Trying to acquire second lock");
        nreLock.lock();
        System.out.println("Acquired second lock");
    }
}

class NonReentrantLock {

    boolean isLocked;

    public NonReentrantLock() {
        isLocked = false;
    }

    public synchronized void lock() throws InterruptedException {

        while (isLocked) {
            wait();
        }
        isLocked = true;
    }

    public synchronized void unlock() {
        isLocked = false;
        notify();
    }
}
```

10. **Mutex vs Semaphore**

Mutex vs Semaphore.pdf

11. **Mutex vs Monitor**

Mutex vs Monitor.pdf

12. **Java's Monitor & Hoare vs Mesa Monitors**

Java's Monitor & Hoare vs Mesa Monitors.pdf

Code:

1. Illegal Monitor Exception

```
class BadSynchronization {

    public static void main(String args[]) throws InterruptedException {
        Object dummyObject = new Object();

        // Attempting to call wait() on the object
        // outside of a synchronized block.
        dummyObject.wait();
    }
}
```

2. **Bad Synchronization Example 2**

```
class BadSynchronization {

    public static void main(String args[]) {
        Object dummyObject = new Object();
        Object lock = new Object();

        synchronized (lock) {
            lock.notify();

            // Attempting to call notify() on the object
            // in synchronized block of another object
            dummyObject.notify();
        }
    }
}
```

13. **Semaphore vs Monitor**

Semaphore vs Monitor.pdf

14. **Amdahl's Law**

Amdahl's Law.pdf

15. **Moore's Law**

Moore's Law.pdf