1

```java
import java.util.Random;

public class MultiThreadExample {
    public static void main(String[] args) {
        RandomNumberGenerator randomNumberGenerator = new RandomNumberGenerator
            ();
        SquareCalculator squareCalculator = new SquareCalculator
            (randomNumberGenerator);
        CubeCalculator cubeCalculator = new CubeCalculator
            (randomNumberGenerator);

        Thread thread1 = new Thread(randomNumberGenerator);
        Thread thread2 = new Thread(squareCalculator);
        Thread thread3 = new Thread(cubeCalculator);

        thread1.start();
        thread2.start();
        thread3.start();
    }
}

class RandomNumberGenerator implements Runnable {
    private volatile boolean running = true;
    private Random random = new Random();
```

```
java -cp /tmp/cYYgvJdpbl MultiThreadExample
Generated random number: 60Square: 3600
Generated random number: 71
Cube: 357911
Generated random number: 60Square: 3600
Generated random number: 83
Cube: 571787
Generated random number: 25
Cube: 15625
Generated random number: 54Square: 2916Generated random number: 90Square: 8100
Generated random number: 22
Square: 484Generated random number: 99
Cube: 970299
Generated random number: 66
Square: 4356
Generated random number: 43Cube: 79507
Generated random number: 29
Cube: 24389
Generated random number: 25
Cube: 15625
Generated random number: 38
Square: 1444
Generated random number: 36
Square: 1296
Generated random number: 95Cube: 857375
```

2

```java
import java.util.LinkedList;
import java.util.Queue;
public class ProducerConsumerExample {
    public static void main(String[] args) {
        Queue<Integer> buffer = new LinkedList<>();
        int capacity = 1; // Buffer capacity

        Producer producer = new Producer(buffer, capacity);
        Consumer consumer = new Consumer(buffer);

        Thread producerThread = new Thread(producer);
        Thread consumerThread = new Thread(consumer);

        producerThread.start();
        consumerThread.start();
    }
}
class Producer implements Runnable {
    private final Queue<Integer> buffer;
    private final int capacity;

    public Producer(Queue<Integer> buffer, int capacity) {
        this.buffer = buffer;
        this.capacity = capacity;
    }
```

```
Consuming: 92
Producing: 93
Consuming: 93
Producing: 94
Consuming: 94Producing: 95
Consuming: 95
Producing: 96
Consuming: 96Producing: 97
Consuming: 97
Producing: 98
Consuming: 98
Producing: 99
Consuming: 99Producing: 100Consuming: 100
Producing: 101
Consuming: 101
Producing: 102
Consuming: 102
Producing: 103
Consuming: 103
Producing: 104
Consuming: 104
Producing: 105Consuming: 105
Producing: 106
Consuming: 106
Producing: 107
```

3

```java
public class ThreadNameChangeExample {
    public static void main(String[] args) {
        Thread customThread = new Thread(new CustomRunnable());
        System.out.println("Original Thread Name: " + customThread.getName());
        customThread.start();
        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
        customThread.setName("NewThreadName");

        System.out.println("Changed Thread Name: " + customThread.getName());
    }
}
class CustomRunnable implements Runnable {
    @Override
    public void run() {
        System.out.println("Custom Thread is running.");
    }
}
```

```
java -cp /tmp/u0S0VX8Keq ThreadNameChangeExample
Original Thread Name: Thread-0
Custom Thread is running.
Changed Thread Name: NewThreadName
```

4

```java
public class ThreadSleepAndNameChangeExample {
    public static void main(String[] args) {
        Thread customThread = new Thread(new CustomRunnable());
        customThread.setName("OriginalThreadName");
        System.out.println("Original Thread Name: " + customThread.getName());
        customThread.start();
        try {
            for (int i = 5; i >= 1; i--) {
                Thread.sleep(6000);
                customThread.setName("NewThreadName" + i);
                System.out.println("Changed Thread Name: " + customThread
                    .getName());
            }
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
}
class CustomRunnable implements Runnable {
    @Override
    public void run() {
        System.out.println("Custom Thread is running.");
    }
}
```

```
java -cp /tmp/uOSOVX8Keq ThreadSleepAndNameChangeExample
Original Thread Name: OriginalThreadName
Custom Thread is running.
Changed Thread Name: NewThreadName5
Changed Thread Name: NewThreadName4
Changed Thread Name: NewThreadName3
Changed Thread Name: NewThreadName2
Changed Thread Name: NewThreadName1
```

5

```java
public class MultiThreadExample {
    public static void main(String[] args) {
        Thread userThread = new Thread(new UserRunnable());
        userThread.start();
        Thread mainThread = Thread.currentThread();
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
        System.out.println("Main thread woke up after sleeping for 1 second.");
    }
}
class UserRunnable implements Runnable {
    @Override
    public void run() {
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
        System.out.println("User thread woke up after sleeping for 1 second.");
    }
}
```

```
java -cp /tmp/uOSOVX8Keq MultiThreadExample
Main thread woke up after sleeping for 1 second.
User thread woke up after sleeping for 1 second.
```

6

```java
import java.util.concurrent.locks.Condition;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;
public class PrinterSynchronizationExample {
    public static void main(String[] args) {
        Printer printer = new Printer();
        Thread job1 = new Thread(new PrintJob(printer, "Job 1"));
        Thread job2 = new Thread(new PrintJob(printer, "Job 2"));
        Thread job3 = new Thread(new PrintJob(printer, "Job 3"));
        job1.start();
        job2.start();
        job3.start();
    }
}
class Printer {
    private Lock lock = new ReentrantLock();
    private Condition condition = lock.newCondition();
    private int currentJob = 1;
    public void print(String jobName) {
        lock.lock();
        try {
            while (!jobName.equals("Job " + currentJob)) {
                condition.await();
            }
```

```
java -cp /tmp/uOSOVX8Keq PrinterSynchronizationExample
Printing Job 1
Printing Job 2
Printing Job 3
```

7

```java
public class CharacterCountExample {
    private static String k = "Hello12345World";
    public static void main(String[] args) {
        ThreadA threadA = new ThreadA();
        threadA.start();
        ThreadB threadB = new ThreadB();
        threadB.start();
    }
    static class ThreadA extends Thread {
        @Override
        public void run() {
            int dc = 0;
            for (char c : k.toCharArray()) {
                if (Character.isDigit(c)) {
                    dc++;
                }
            }
            System.out.println("ThreadA:" + dc);
        }
    }
    static class ThreadB extends Thread {
        @Override
        public void run() {
            int cc = 0; // Alphabetic character count
```

```
java -cp /tmp/uOSOVX8Keq CharacterCountExample
ThreadB:10ThreadA:5
```

8

```java
import java.util.Scanner;
public class UserThreadPriorityExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a String: ");
        String userInputString = scanner.nextLine();
        System.out.print("Enter a Character: ");
        char userInputCharacter = scanner.next().charAt(0);
        UserThreadPriority threadobj1 = new UserThreadPriority("ThreadA",
            userInputString, userInputCharacter);
        UserThreadPriority threadobj2 = new UserThreadPriority("ThreadB",
            userInputString, userInputCharacter);
        threadobj1.start();
        threadobj2.start();
        scanner.close();
    }
}
class UserThreadPriority extends Thread {
    private String k;
    private char c;
    public UserThreadPriority(String threadName, String k, char c) {
        super(threadName);
        this.k = k;
        this.c = c;
    }
```

```
java -cp /tmp/uOSOVX8Keq UserThreadPriorityExample
Enter a String: 8
Enter a Character: c
Thread ThreadA: k = 8, c = cThread ThreadB: k = 8, c = c
```

9

```java
public class ThreadSleepExample {
    public static void main(String[] args) {
        MyThread myThread = new MyThread();
        myThread.start();
    }
}
class MyThread extends Thread {
    @Override
    public void run() {
        try {
            System.out.println("Thread is running.");
            sleep(0, 10);
            System.out.println("Thread woke up after sleeping for 10 ns.");
            sleep(0, 20);
            System.out.println("Thread woke up after sleeping for 20 ns.");
            sleep(0, 50);
            System.out.println("Thread woke up after sleeping for 50 ns.");
            sleep(0, 70);
            System.out.println("Thread woke up after sleeping for 70 ns.");
            sleep(0, 100);
            System.out.println("Thread woke up after sleeping for 100 ns.");
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
```

```
java -cp /tmp/uOSOVX8Keq ThreadSleepExample
Thread is running.
Thread woke up after sleeping for 10 ns.Thread woke up after sleeping for 20 ns.
Thread woke up after sleeping for 50 ns.
Thread woke up after sleeping for 70 ns.
Thread woke up after sleeping for 100 ns.
```

10

```java
public class ThreadPriorityExample {
    public static void main(String[] args) {
        Thread thread1 = new Thread(new MyRunnable(), "Thread 1");
        Thread thread2 = new Thread(new MyRunnable(), "Thread 2");
        Thread thread3 = new Thread(new MyRunnable(), "Thread 3");
        Thread thread4 = new Thread(new MyRunnable(), "Thread 4");
        Thread thread5 = new Thread(new MyRunnable(), "Thread 5");
        thread1.setPriority(Thread.MIN_PRIORITY); // Priority 1
        thread2.setPriority(3); // Priority 3
        thread3.setPriority(5); // Priority 5
        thread4.setPriority(7); // Priority 7
        thread5.setPriority(Thread.MAX_PRIORITY); // Priority 10
        thread1.start();
        thread2.start();
        thread3.start();
        thread4.start();
        thread5.start();
    }
    static class MyRunnable implements Runnable {
        @Override
        public void run() {
            Thread currentThread = Thread.currentThread();
            System.out.println(currentThread.getName() + " is running with
                priority " + currentThread.getPriority());
        }
```

```
java -cp /tmp/u0S0VX8Keq ThreadPriorityExample
Thread 2 is running with priority 3Thread 5 is running with priority 10Thread 1 is
    running with priority 1Thread 3 is running with priority 5Thread 4 is running with
    priority 7
```