1

```
package demo;
import java.util.HashMap;
import java.util.Map;
   public class Task {

           public static void main(String[] args) {
               String input = "best items in category are samsung,
Lenovo, samsung items are cool";

               String[] words = input.split("\\s+");
               Map<String, Integer> wordCount = new
HashMap<>();

               for (String word : words) {

                   word = word.toLowerCase();

                   word = word.replaceAll("[^a-zA-Z]", "");

                   if (!word.isEmpty()) {
                       wordCount.put(word,
wordCount.getOrDefault(word, 0) + 1);
                   }
               }

               for (Map.Entry<String, Integer> entry :
wordCount.entrySet()) {
                   System.out.println(entry.getKey().substring(0,
1).toUpperCase() + entry.getKey().substring(1) + "-" +
entry.getValue());
```

```
            }
        }
    }
```

Output:

```
Samsung-2
In-1
Are-2
Cool-1
Best-1
Category-1
Items-2
Lenovo-1
```

2)

```
public class ThreadDemo {
    public static void main(String[] args) {
        final Object lock = new Object();

        Thread thread1 = new Thread(() -> {
            synchronized (lock) {
                try {
                    System.out.println("Thread 1: Waiting");
                    lock.wait(); // Thread 1 waits for notification
                    System.out.println("Thread 1: Notified");
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                }
            }
        });

        Thread thread2 = new Thread(() -> {
```

```
        synchronized (lock) {
            try {
                Thread.sleep(2000); // Sleep for 2 seconds
                System.out.println("Thread 2: Sending
Notification");
                lock.notify(); // Thread 2 notifies Thread 1
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
    });

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

Output:

```
Thread 1: Waiting
Thread 2: Sending Notification
Thread 1: Notified
```

3)

```
package thread;
import java.io.IOException;


public class Thro {
            public static void main(String[] args) {
                try {
                    // Calling a method that throws an exception
                    divide(10, 0);
                } catch (ArithmeticException e) {
                    System.out.println("Caught an 
ArithmeticException: " + e.getMessage());
                }

                try {
                    // Calling a method that specifies an exception 
with throws
                    readFile("file.txt");
                } catch (IOException e) {
                    System.out.println("Caught an IOException: " 
+ e.getMessage());
                }

                // Throwing a custom exception
                try {
                    int age = -5;
                    if (age < 0) {
                        throw new IllegalArgumentException("Age 
cannot be negative.");
                    }
                } catch (IllegalArgumentException e) {
                    System.out.println("Caught a custom 
IllegalArgumentException: " + e.getMessage());
                }
            }

            // A method that throws an exception
            public static int divide(int numerator, int 
denominator) {
```

```
            if (denominator == 0) {
                throw new ArithmeticException("Division by
zero is not allowed.");
            }
            return numerator / denominator;
        }

        // A method that specifies an exception with throws
        public static void readFile(String fileName)
throws IOException {
            // Simulate a file reading operation
            if (fileName.equals("file.txt")) {
                throw new IOException("File not found.");
            }
        }
    }
```

Output:

```
Caught an ArithmeticException: Division by zero is not
allowed.
Caught an IOException: File not found.
Caught a custom IllegalArgumentException: Age cannot be
negative.
```