🚦 Traffic Light Control System using HashSet

You are tasked with building a **Traffic Light Control System** to manage active traffic signals at various intersections using a **HashSet** in Java.
Each intersection is identified by a **unique name (String)**, and the system maintains the set of active signals in a `HashSet<String>`.

The system must allow operators to:

- Register active signals
- Verify the presence of a signal at an intersection
- Deactivate signals
- Display the updated signal list
- Count intersections with names longer than a given length

If an operation references an intersection that does not exist in the system, a **custom exception** `SignalNotFoundException` must be thrown.

🧩 Project Structure and Files

The solution must be structured into the following Java files:

1. `TrafficLightSystem.java`

Maintains private `HashSet<String> activeSignals`.

Provides all core methods for registration, verification, deactivation, display, and length-based counting.

2. `SignalNotFoundException.java`

Custom exception thrown when an intersection is not found in the signal system.

3. `TrafficControlApp.java`

Contains the `main()` method.

Accepts user input via `Scanner`, handles input validation, and invokes methods from `TrafficLightSystem`.

🧱 Class Responsibilities

🟩 TrafficLightSystem

Field:

```java
private HashSet<String> activeSignals;
```

Public methods:

```java
public boolean registerIntersection(String name)
public void verifySignal(String name) throws SignalNotFoundException
public void deactivateSignal(String name) throws SignalNotFoundException
public void displayActiveSignals()
public void countLongNames(int threshold)
```

🟥 SignalNotFoundException

Must extend `Exception`.

Constructor:

```java
public SignalNotFoundException(String message)
```

🟦 TrafficControlApp

- Reads user input
- Performs validation on the number of intersections (`n`) and length threshold (`L`)
- Calls appropriate methods from `TrafficLightSystem`
- Catches and handles exceptions with user-friendly output

⚙️ Functional Requirements

1. Register Intersections

- The system must read and store `n` unique intersection names into a `HashSet`.

2. Verify Intersection

- The system must check if a specific intersection exists in the set of active signals.
- If not found, it must throw a `SignalNotFoundException`.

3. Deactivate Intersection

- The system must remove a specific intersection from the set.
- If the intersection is not found, it must throw a `SignalNotFoundException`.

4. Display Active Signals

- The system must display the current contents of the `HashSet`.

5. Count Long Names

- The system must count how many intersection names in the set are longer than a provided threshold value.

🔎 Input Validation

- If `n` is zero or negative, the process must terminate with a validation message.
- If the length threshold `L` is negative, the process must terminate with a validation message.

📥 Input Format

1. The first input is an integer `n`, representing the number of intersections with active signals.
2. The next `n` lines each contain a **unique intersection name (string)**.
3. The following line contains a string representing an **intersection name to verify**.
4. The next line contains a string representing an **intersection name to deactivate**.
5. The final line contains an integer `L`, representing the **name length threshold**.

📤 Output Format

✅ Verify Intersection Signal Status:

- If found, print:

```csharp
Traffic signal is active at [Intersection].
```

- If not found, throw `SignalNotFoundException` and print:

```pgsql
No active traffic signal found at [Intersection].
```

✅ Deactivate a Traffic Signal:

- If found and removed, print:

```css
Traffic signal at [Intersection] has been deactivated.
```

- If not found, throw `SignalNotFoundException` and print:

```pgsql
No active traffic signal found at [Intersection].
```

✅ Display Updated List:

Print:

```yaml
Updated list of active traffic signals:
Intersection: [Name1]
Intersection: [Name2]
...
```

✅ Count Intersections with Names Longer Than 'L' Characters:

Print:

```javascript
Number of intersections with names longer than [L] characters: [count]
```

⚠️ Validation Errors

- If `n <= 0`, print:

```typescript
The number of intersections must be a positive number.
```

- If `L < 0`, print:

```css
Threshold length must be a non-negative number.
```

🧪 Sample Input 1

```
2
MainStreet
OakAvenue
MainStreet
OakAvenue
5
```

📃 Sample Output 1

vbnet

```
Traffic signal is active at MainStreet.
Traffic signal at OakAvenue has been deactivated.
Updated list of active traffic signals:
Intersection: MainStreet
Number of intersections with names longer than 5 characters: 1
```

🧪 Sample Input 2

```
3
Broadway
HillRoad
ParkLane
Crossroad
BlockLane
12
```

📃 Sample Output 2

yaml

```
No active traffic signal found at Crossroad.
No active traffic signal found at BlockLane.
Updated list of active traffic signals:
Intersection: HillRoad
Intersection: Broadway
Intersection: ParkLane
Number of intersections with names longer than 12 characters: 0
```

🧪 Sample Input 3

```diff
-2
SunsetBoulevard
OceanDrive
SunsetBoulevard
OceanDrive
7
```

📃 Sample Output 3

```typescript
The number of intersections must be a positive number.
```

🧪 Sample Input 4

```diff

2
WoodCreek
Broadway
Broadhill
WillowCreek
-10
```

📃 Sample Output 4

```css

Threshold length must be a non-negative number.
```