1. ❗ Recursion: `new FlipCoin(...)` Creates a New Object Recursively

You're using recursion (`new FlipCoin(...)`) to simulate each round. This can lead to **stack overflow** if many rounds are needed.

I didn't consider this to have impact on my execution.. I understand I am using recursion here and Stack overflow was not a concern since simulation was not based on a higher level of coin flips.

2. ❗ `executions` is static but gets reset every time

You're using a `static int executions`, which should persist across all instances. However, because you're **creating new instances recursively**, you lose control over its behavior.

> 🔧 **Fix**: Avoid creating new `FlipCoin` objects. Instead, have a single object and manage rounds with a loop method.

This was the most difficult observation...
In my code, I did declare this as a static class level variable (executions)... And hence I would expect it to be available across all instances. My understanding was that on recursion, it still mantains its correct state.

3. ❌ Redundant Parameters

`flipCoinsAgain()` takes `heads`, `tails`, `numberCoins`, and `coins`, but recalculates them internally anyway.

> 🔧 **Fix**: Either pass updated coin values correctly or manage state within the object itself.

int heads and int tails - Again, this was very confusing.. I had introduced variables inside the constructor FlipCoin. It was not suitable to be a class level static variable since it had to be reset to 0.
I discovered chatGPT was correct. There was not a requirement to pass int heads int tails into flipCoinsAgain since this method was effectively calling constructor again (FlipCoin). And the constructor reset the values.....

int numberCoins was declared as class level variable for FlipCoin class since when an instance was created from the main class, it received values via constructor FlipCoin. This has to be passed to FlipCoinAgain since FlipCoinAgain calls the constructor again FlipCoin...
We need to pass this variable around otherwise the whole code would break.... (since FlipCoin is instantiated via the main class and receives the variable).

int coins was confusing again. I only used this variable whilst creating a new instance of FlipCoin from within FlipCoinAgain method. This was a mandatory requirement due to the constructor signature.

4. ❓ Misleading `getRounds()` Method

This method is unused and would return wrong results if `executions` tracking is off.

> 🔧 **Fix**: Use this method from `main()` if you plan to return the count of executions after the simulation ends.

I can relate to this.. This method was redundant because I had a call to this method once it finished with the FlipCoin and FlipCoin again. The method call was part of the main method... But since the code did not exit gracefully (hence using System.exit), I had to comment out method call

5. 📦 Unused or Redundant Imports

You import both `Random` and `*` from `java.util`.

> 🔧 **Fix**: Only import `Random`.

java Copy Edit

```java
import java.util.Random;
```

I recognise this as human error and this can be readily remediated...

I require this to prevent code from looping.

```java
if (heads==1)
{
    System.out.println("One head left: " + numberCoins);

    executions++;

    System.out.println("Number executions:" +  (executions));
    System.exit(0);
}
```

CODE WILL LOOP WITHOUT THIS