1st Question

You are tasked with creating two functional interfaces and implementing them using anonymous inner classes in Java.

---

1. NumberChecker Interface:

- **Functional Interface**: This interface should contain the following method:
  - **Method Name**: `checkNumber`
  - **Parameters**: `int` (a single integer)
  - **Return Type**: `boolean`
  - **Description**: The method `checkNumber` should determine if the provided integer is a **prime number** or not.

---

2. CharChecker Interface:

- **Functional Interface**: This interface should contain the following method:
  - **Method Name**: `checkChar`
  - **Parameters**: `char` (a single character)
  - **Return Type**: `void`
  - **Description**: The method `checkChar` should check whether the **ASCII value** of the provided character is prime or not. It should print the result.

---

3. Main Class:

- **isPrime Static Method**:
  - Implement a static method named `isPrime` in the **Main** class that checks whether an integer (either a number or ASCII value) is a prime number.
  - **Method Signature**: `public static boolean isPrime(int number)`
  - **Description**: The method should return `true` if the given number is prime and `false` otherwise.

---

4. Anonymous Inner Class Implementations:

- **NumberChecker Anonymous Inner Class**:
  - In the **main** method, create an instance of the `NumberChecker` interface using an anonymous inner class.
  - Implement the `checkNumber` method to check whether a sample integer (e.g., `7`) is prime and print the result.
- **CharChecker Anonymous Inner Class**:
  - In the **main** method, create an instance of the `CharChecker` interface using another anonymous inner class.
  - Implement the `checkChar` method to check whether the ASCII value of a sample character (e.g., `'A'`) is prime and print the result.

2nd Question

You are tasked with creating a user-defined class and using predefined functional interfaces to analyze the temperature value.

---

1. SensorData Class:

- **Attributes**:
  - Define a class `SensorData` with two private attributes:
    - `double temperature`
    - `double humidity`
- **Constructor**:
  - Provide a **parameterized constructor** to initialize these attributes.
- **Setters and Getters**:
  - Provide **setter** and **getter** methods for both the `temperature` and `humidity` attributes.

---

2. Functional Interfaces for Temperature Analysis:

In the **SensorData** class, implement the following logic:

- **First Functional Interface (`Function<SensorData, Double>`)**:
  - Define a `Function<SensorData, Double>` interface to extract and return the **temperature** from the `SensorData` instance.
  - **Method Signature**: `Double apply(SensorData Data)`
  - Use an **anonymous inner class** to implement this functional interface and extract the temperature from a `SensorData` object.
  - Store and display the extracted temperature.
- **Second Functional Interface (`Function<Double, Boolean>`)**:
  - Define a `Function<Double, Boolean>` interface that takes a `Double` **(temperature)** as a parameter and returns a `Boolean`.
  - This function should return `true` if the temperature is above a specified **threshold** (e.g., 25 degrees Celsius), and `false` otherwise.
  - Use an **anonymous inner class** to implement this functional interface and check if the extracted temperature is above the threshold.
  - Print the result (`true` or `false`) based on the function's return value.

---

3. Threshold and Main Logic:

- **Local Variable**:
  - Define a local variable `double threshold = 25.0` for the temperature threshold.
- **Temperature Checking**:
  - In the **main method**, apply the first function to the `SensorData` object to extract the temperature.
  - Apply the second function to check if the temperature exceeds the threshold and print `true` or `false` based on the result.
