---

## Question 1: Pattern

**Built-In Classes & Methods Used:**

1. **`StringBuilder`**
    - **What it is:** A mutable sequence of characters that allows efficient modification of strings.
    - **Method Used:**
        - `append(char c)`
            - **Purpose:** Adds a character to the end of the current sequence.
            - **Parameters:** A single character `c`.
            - **Returns:** The `StringBuilder` object itself (method chaining is possible).
        - `setLength(int newLength)`
            - **Purpose:** Resets the length of the sequence. Useful for clearing the content.
            - **Parameters:** An integer `newLength` (e.g., `0` to clear).
            - **Returns:** Void.
    - **How/Where to Use:** Use `StringBuilder` when you need to build or modify strings frequently, as it is more efficient than concatenating strings.
2. **`Character`**
    - **What it is:** A utility class for working with `char` data types.
    - **Methods Used:**
        - `isLetter(char ch)`
            - **Purpose:** Checks if the character is a letter.
            - **Parameters:** A single character `ch`.
            - **Returns:** `true` if `ch` is a letter, `false` otherwise.
        - `isDigit(char ch)`
            - **Purpose:** Checks if the character is a digit.
            - **Parameters:** A single character `ch`.
            - **Returns:** `true` if `ch` is a digit, `false` otherwise.
    - **How/Where to Use:** Use `Character` methods when validating or analyzing individual characters in strings.
3. **`Integer`**
    - **What it is:** A wrapper class for primitive `int` values.
    - **Method Used:**
        - `parseInt(String s)`
            - **Purpose:** Converts a numeric string into an `int`.
            - **Parameters:** A string `s` representing a number.
            - **Returns:** The parsed integer value.

- **How/Where to Use:** Use `Integer.parseInt` to convert strings containing numbers into integers for calculations.

---

## Question 2: ReverseString

**Built-In Classes & Methods Used:**

1. **`String`**
   - **What it is:** A sequence of characters.
   - **Methods Used:**
     - `split(String regex)`
       - **Purpose:** Splits the string into an array of substrings based on the specified delimiter.
       - **Parameters:** A regular expression `regex` (e.g., `" "` for splitting by spaces).
       - **Returns:** An array of strings.
   - **How/Where to Use:** Use `split` to break a string into parts for processing, such as reversing the order of words.
2. **`System.out`**
   - **What it is:** A standard output stream for printing to the console.
   - **Methods Used:**
     - `print(String s)` and `println(String s)`
       - **Purpose:** Prints text to the console.
       - **Parameters:** A string `s`.
       - **Returns:** Void.
   - **How/Where to Use:** Use these methods for debugging or displaying results.

---

## Question 3: Anagram

**Built-In Classes & Methods Used:**

1. **`Arrays`**
   - **What it is:** A utility class for working with arrays.
   - **Methods Used:**
     - `sort(char[] a)`
       - **Purpose:** Sorts the elements of a character array in ascending order.
       - **Parameters:** A character array `a`.
       - **Returns:** Void (modifies the array in-place).
     - `equals(char[] a, char[] b)`
       - **Purpose:** Compares two arrays for equality.
       - **Parameters:** Two character arrays `a` and `b`.
       - **Returns:** `true` if both arrays have the same length and elements, `false` otherwise.

- **How/Where to Use:** Use `Arrays.sort` for sorting arrays before comparison and `Arrays.equals` for checking equality of arrays.

2. **`String`**
    - **What it is:** A sequence of characters.
    - **Methods Used:**
        - `length()`
            - **Purpose:** Returns the length of the string.
            - **Parameters:** None.
            - **Returns:** An integer representing the length.
        - `toCharArray()`
            - **Purpose:** Converts a string into a character array.
            - **Parameters:** None.
            - **Returns:** A character array containing the string's characters.
    - **How/Where to Use:** Use `toCharArray` to convert strings into arrays for sorting or character-level analysis.

---

## Summary of Built-In Classes and Topics

**Classes:**

- `StringBuilder`: For building/modifying strings efficiently.
- `Character`: For analyzing characters (e.g., checking if they are letters or digits).
- `Integer`: For parsing numeric strings into integers.
- `String`: For manipulating strings (e.g., splitting, converting to arrays).
- `Arrays`: For sorting and comparing arrays.

**Key Topics:**

1. **String Manipulation:**
    - Splitting, reversing, and processing strings.
2. **Character Validation:**
    - Checking for letters and digits using `Character`.
3. **Array Operations:**
    - Sorting and comparing arrays with `Arrays`.
4. **Basic I/O:**
    - Printing output with `System.out`.