```java
package Assign_3;


import java.io.Serializable;


/** This interface provides the specification for a sequence of characters.
  *
  * Note  it extends  CharSequence, provided by java.lang, that's also a supertype
  * of String.
  *
  * @see <a href="https://docs.oracle.com/javase/8/docs/api/java/lang/CharSequence.html">java.lang.
CharSequence</a>
  *
  * @author Earl Foxwell (adapted by D. Hughes)
  * @version 1.0 (Feb, 2016)                                                   */



public interface CharacterSequence extends CharSequence, Serializable {


 /** This method compares this character sequence to the provided sequence, cs.
   * If the provided sequence is null, or if the two sequences differ by at least
   * one character, returns false. Otherwise, returns true.
   *
   * @param  cs  the other character sequence against which to compare
   * @return  boolean  true if cs matches this one; false otherwise              */

 public boolean equals ( CharSequence cs );


 /** This method performs lexicographic comparison of this character sequence
   * against another.
   *
   * Returns a positive value if this character sequence comes after the provided
   * sequence, cs. Returns a negative value if this character sequence comes before
   * the provided sequence. Returns zero if this character sequence has the same
   * lexicographic position as the provided sequence. (i.e. returns 0 if the two
   * sequences are equal.
   *
   * @param  cs  the other character sequence against which to compare
   * @return  int  neagtive if less, 0 if equal positive if greater.            */

 public int compareTo ( CharSequence cs );


 /** This method creates and returns a new version of the character sequence, where
   * all uppercase letters have been replaced with their lowercase counterparts.
   * Non-alphabetic characters (and lowercase letters) are unaffected.
   *
   * @return  CharacterSequence  A copy of this character sequence all lowercase.*/

 public CharacterSequence toLowerCase ( );


 /** This method creates and returns a new version of the character sequence, where
   * all lowercase letters have been replaced with their uppercase counterparts.
   * Non-alphabetic characters (and uppercase letters) are unaffected.
   *
   * @return  CharacterSequence  A copy of this character sequence all uppercase.*/
```

```

public CharacterSequence toUpperCase();


/** This method returns a copy of the character sequence, with leading and
  * trailing whitespace removed. If the original character sequence was empty it
  * returns an empty character sequence. If the original sequence consisted only
  * of whitespace itreturn an empty character sequence. Otherwise it returns a
  * character sequence that represents the largest sub-sequence that doesn't
  * begin or end with whitespace characters.
  *
  * For the sake of identifying whitespace, it is acceptable to either use the
  * Character class's isWhitespace() function.
  *
  * @return  CharacterSequence A copy of this character sequence with no leading
  * or trailing whitespace.                                                    */

public CharacterSequence trim ( );

/** This method returns a copy of the character sequence, with all whitespace removed.
  * If the original character sequence was empty it
  * returns an empty character sequence. If the original sequence consisted only
  * of whitespace it return an empty character sequence. Otherwise it returns a
  * character sequence that represents the largest sub-sequence that doesn't
  * contain any whitespace characters.

  * @return  ConCharacterSequence A copy of this character sequence with no whitespace char */

public ConCharacterSequence trimAll ( );

/** This method returns a new CharacterSequence resulting from replacing all
  * occurrences of oldChar with newChar.
  *
  * @param  oldChar  the old character
  * @param  newChar  the replacement character
  *
  * @return  CharacterSequence  The resulting character sequence.                 */

public CharacterSequence replace ( char oldChar, char newChar );


/** This method returns a new CharacterSequence that represents the concatenation
  * of this sequence followed by the provided sequence. If the provided additional
  * sequence is null, then a copy of this sequence is all that's returned.
  *
  * @param  tail  character sequence to append to end of this sequence
  *
  * @return CharacterSequence  a character sequence consisting of this sequence
  *                            followed by tail.                               */
public CharacterSequence concat ( CharSequence tail );

 /** This method returns a boolean value of true if this sequence is a palindrome
  * and false if not
  *
  * @return boolean value                        */

public boolean isPalindrome ( );



/* Note: The following declarations are already inherited from the CharSequenc
```

```
 * interface. They're included here for offline readability.                        */


/** This method eturns the character at the specified position (zero-based) of the
  * sequence. (Inherited from CharSequence)
  *
  * @param  index  position of requested character in sequence.
  *
  * @return  char  requested character
  * @throws IndexOutOfBoundsException if the index is not within the range
  * [0,length())                                                                     */

public char charAt( int index );


/** This emthod returns the number of characters in this sequence.
  * (Inherited from CharSequence)
  *
  * @return  int  number of characters in this sequence.                            */

public int length ( );


/** This method returns a slice (or substring) from this character sequence.start
  * must be in range [0,length()). end must be in the range [0,length()].
  * (Inherited from CharSequence)
  *
  * @param  start  starting index, inclusive
  * @param  end     end index, exclusive
  *
  * @return  CharSequence  the requested slice
  * @throws  IndexOutOfBoundsException if either index is out of bounds.           */

public CharSequence subSequence ( int start, int end );


/** Converts this character sequence into a java.lang.String
  * (Inherited from CharSequence)
  *
  * @return  String  the equivalent String representation of this character
  * sequence.                                                                       */

public String toString();


}
//Xinan Wang No.5535802
```