Assignment-2

Aim: String Handling Functions

1. String Length:

- The java string length() method used for length of the string. It returns count of total number of characters.
- The length() method returns the length of the string.

Example:

Input:

```
import java.lang.String;
class Test{
        public static void main(String[] args){
        String s1 = "Hello_World";
        System.out.print(s1.length());
}}
```

Output:

11

2. String Concatenation :

- The java string concat() method combines specified string at the end of this string. It returns combined string. It is like appending another string.
- public String concat(String anotherString)
- The + operator is used to concatenate two or more strings.
- For string concatenation the Java compiler converts an operand to a String whenever the other operand of the + is a String object.

Example:

Input:

```
import java.lang.String;
public class Test{
public static void main(String[] args){
        String s1 = new String("Parul");
        String s2 = new String("University");
        System.out.println("String s1+s2 = " + s1 + s2 );
        System.out.println("String Concatenation: "+ s1.concat(s2));

}}
```

Output:
String s1+s2 = ParulUniversity
String Concatenation: ParulUniversity

3. Changing Case of String:

- toLowerCase(): Converts all of the characters in a String to lower case.
- toUpperCase(): Converts all of the characters in this String to upper case.
- public String toLowerCase()
- public String toUpperCase()

Example:

Input:

```
import java.lang.String;
class Test{
    public static void main(String[] args){
    String s1 = "Hello WOrld";
    System.out.println("UpperCase:"+s1.toUpperCase()+"\nLowerCase:"+
s1.toLowerCase());
    }}
```

Output:

Upper Case :HELLO WORLD
Lower Case:hello world

4. Character Extraction:

1. **getChar():**
    - Copies more than one characters from string into the destination character array.
    - public void getChars(int srcBeginIndex , int srcEndIndex , char[] destination , int dstBeginIndex)

Example:

Input:

```
import java.lang.String;
class Test{
public static void main(String[] args){
String s1 ="Java is Case Sensitive";
char[] ch = new char[10];
s1.getChars(2,11,ch,0);
System.out.print(ch);
}}
```

Output:

va is Cas

2. **charAt():**
    - Used to extract Characters at a specific index of string.
    - The java string charAt() method returns a char value at the given index number. The index number starts from 0.
    - It returns StringIndexOutOfBoundsException if given index number is greater than this string or negative index number.
    - public char charAt(int index)
    - Returns the character at the specified index. An index ranges from 0 to length() - 1. The first character of the sequence is at index 0, the next at index 1, and so on, as for array indexing.

Example:

Input:

```
import java.lang.String;
class Test{
        public static void main(String[] args){
        char ch = "186380307048".charAt(2);
        System.out.print(ch);
}
}
```

Output:

6

3. **Substring:**
    - The java string substring() method returns a part of the string.
    - We pass begin index and end index number position in the java substring method where start index is inclusive and end index is exclusive. In other words, start index starts from 0 whereas end index starts from 1.
    - There are two types of substring methods in java string.
        1. public String substring(int startIndex)
        2. public String substring(int startIndex, int endIndex)

Example:

Input:

```
import java.lang.String;
class Test{
        public static void main(String[] args){
        String s1 = "Java is Immutable";
        System.out.println(s1.substring(0,4));
        System.out.println(s1.substring(4));
}
}
```

Output:

```
Java
is Immutable
```

5. **String Comparison:**

- We can compare string in java on the basis of content and reference.
- It is used in
  i. authentication (by equals() method),
  ii. sorting (by compareTo() method),
  iii. reference matching (by == operator) etc.

1) Equals()Method:

- The String equals() method compares the original content of the string. It compares values of string for equality. String class provides two methods:
  (i) public boolean equals(Object another) compares this string to the specified object.
  (ii) public boolean equalsIgnoreCase(String another) compares this String to another string, ignoring case.

Example:

Input:

```
import java.lang.String;
class Test{
        public static void main(String[] args){
        String s1="parul university";
        String s2 ="PARUL UNIVERSITY";
        String s3 = "piet-ds";

        System.out.println(s1.equals(s2));
        System.out.println(s1.equals(s3));
        System.out.println(s1.equalsIgnoreCase(s2));

}}
```

Output:

```
False
False
True
```

2) Using == Operator:

- The = = operator compares references not values.

Example:

Input:

```
        import java.lang.String;
class Test{
        public static void main(String[] args){
        String s1="parul university";
```

```
        String s2 ="parul university";
        String s3 = new String("parul university");

        System.out.println(s1 == s2);
        System.out.println(s1 == s3);

    }}
```

Output:

```
True
False
```

3) CompareTo() Method:

- **The String compareTo() method compares values and returns an integer value that describes if first string is less than, equal to or greater than second string.**
- **Suppose string1 and string2 are two string variables.**
  If:
  string1 == string2 : 0
  string1 > string2 : positive value
  string1 < string2 : negative value

Example:

Input:

```
import java.lang.String;
class Test{
    public static void main(String[] args){
    String s1="parul university";
    String s2 ="parul university";
    String s3 = new String("piet-ds");
    System.out.println(s1.compareTo(s2));
    System.out.println(s1.compareTo(s3));
    System.out.println(s3.compareTo(s1))        ;

    }}
```

Output:

```
0
-8
8
```