Taking input from the user in java:
==========

--In java(desktop application) we can take input from the user inside our application in 3 ways:

1. Scanner class (java 1.5 version)

2. using Command Line argument

3.using BufferedReader class

Note: with the help of CLA, and BR class we can take input only in the form of String.
--using Scanner class we can take the input in almost all the primitive datatype also.

--This Scanner class belongs to java.util package.

in java common classes (String, Object, System, etc....) they belongs to java.lang package.

System

fully qualified name :

java.lang.System

java.util.Scanner
com.masai.Student

--if we want to utilize/use any classes which belongs to other than java.lang package , inside our class, then we need to import that class inside our class.

```
//creating Scanner class object by providing the keyboard address as a source.
Scanner sc = new Scanner(System.in);  // System.in represent the address of the keyboard.
```

--this Scanner class having multiple methods by using which we can read the data in our application in various primitive data types:

```
nextInt();
nextFloat();
```

nextLong();
nextBoolean();
next(); //read the String
----------> these methods will read the next token in their respected data type, they will not read the entire line.

nextLine() // read the String // it will read the entire line.

example:

Demo.java:

```
package com.masai;

import java.util.Scanner;

public class Demo {

	public static void main(String[] args) {

		Scanner sc= new Scanner(System.in);

		System.out.println("Enter a number :");

		int num= sc.nextInt();

		System.out.println("That number is :"+num);


	}

}
```

ex2:

```
		Scanner sc= new Scanner(System.in);
```

```
System.out.println("Enter a roll :");
int roll= sc.nextInt();

System.out.println("Enter marks :");
int marks= sc.nextInt();


System.out.println("The Roll is :"+roll);
System.out.println("The Marks is :"+marks);
```

next() and nextLine():-

==========

Demo.java:

```
Scanner sc= new Scanner(System.in);

System.out.println("Enter a roll :");
int roll= sc.nextInt();


System.out.println("Enter Name :");
String name= sc.next();


System.out.println("Enter marks :");
int marks= sc.nextInt();


System.out.println("The Roll is :"+roll);
System.out.println("The Name is :"+name);
System.out.println("The Marks is :"+marks);
```

explanation: next();

Enter roll
10 "/n"
Enter name:
Ram Kumar singh "/n" // here kumar is consumed as next token as marks value

Enter Marks
20 "/n"

explanation: nextLine();

Enter roll
10 "/n"
Enter name: // here sc.nextLine() will consume the above "/n"
Ram Kumar singh "/n"
Enter Marks
20 "/n"

---so to solve the above problem we have following solutions:

1. never use nextLine() always read the token by using next();
//with this we can not read full space sepetated name.

2. always use nextLine(); //even to read the primitives also.

ex:

```
Scanner sc= new Scanner(System.in);

System.out.println("Enter a roll :");
int roll= Integer.parseInt(sc.nextLine());


System.out.println("Enter Name :");
String name= sc.nextLine();


System.out.println("Enter marks :");
int marks= Integer.parseInt(sc.nextLine());


System.out.println("The Roll is :"+roll);
System.out.println("The Name is :"+name);
System.out.println("The Marks is :"+marks);
```

3. if we need to use nextLine() , then use it at begining.
ex:

```
Scanner sc= new Scanner(System.in);
```

```
System.out.println("Enter Name :");
String name= sc.nextLine();


System.out.println("Enter a roll :");
int roll= sc.nextInt();


System.out.println("Enter marks :");
int marks= sc.nextInt();


System.out.println("The Roll is :"+roll);
System.out.println("The Name is :"+name);
System.out.println("The Marks is :"+marks);
```

4. call the blank nextLine() after reading the token to consume the buffered charecter "/n".

```
Scanner sc= new Scanner(System.in);


System.out.println("Enter a roll :");
int roll= sc.nextInt();

sc.nextLine();


System.out.println("Enter Name :");
String name= sc.nextLine();


System.out.println("Enter marks :");
int marks= sc.nextInt();


System.out.println("The Roll is :"+roll);
System.out.println("The Name is :"+name);
System.out.println("The Marks is :"+marks);
```

reading the data in the form of matrix:

```
Scanner sc= new Scanner(System.in);

System.out.println("Enter number in matrix");

 int n1= sc.nextInt();
 int n2= sc.nextInt();
 int n3= sc.nextInt();

 sc.nextLine();

 int n4= sc.nextInt();
 int n5= sc.nextInt();
 int n6= sc.nextInt();


 System.out.println(n1+" "+n2+" "+n3);
 System.out.println(n4+" "+n5+" "+n6);
```

String class:
==========

--this class belongs to java.lang package.

"java.lang.String"

--in java String is a group of charecter.

String class object we can create in 2 ways:

```
String s1 = new String("Welcome");

String s2 = "Welcome";
```

ex:

Demo.java:

```
package com.masai;

public class Demo {


	Demo(String s){

	}


	public static void main(String[] args) {

		String s1 = new String("Welcome");

		Demo d1 = new Demo("hello");

		System.out.println(d1);// Demo@34232
		System.out.println(s1);// Welcome


	}

}
```

--there are multiple overloaded println() method is defined inside the PrintStream class :

```
1.println(){
//printing line break
}
2. println(primitives){
print the primitive
}
3.println(String s){
// it will print the content
}
4.println(Object obj){
//it will print the address
}
5.println(char[] chr)
```

ex:

```
String s1 = new String("Welcome");

String s2 = "Welcome";
String s3 = "Welcome";

System.out.println(s1.equals(s2)); // true // compare the content
System.out.println(s1 == s2); // false //compare the ref
```

diff bt

```
String s1="Welcome";  // here one object is created
String s2 = new String("Welcome");   // here 2 object is created.
```

example:

Demo.java:

```
package com.masai;

public class Demo {

	public static void main(String[] args) {

		String s1 = "Welcome";

		String s2 = "welcome";

		String s3 = new String("Welcome");

		String s4 = new String("Hello");

	}

}
```

reffer String_scp diagram..

Note: Garbage collector does not have any effect on SCP area,

--SCP area will be sharable in entire application, it is one per JVM.

example:

Demo.java:-

```
package com.masai;

public class Demo {

	public static void main(String[] args) {


		Demo d1 = new Demo();


		d1=null;

		Demo d2 = new Demo();


		String s1 = "Welcome";

		System.out.println(System.identityHashCode(s1));

		s1= null;


		String s2 = "Welcome";

		System.out.println(System.identityHashCode(s2));
```

```
	}

}
```

example2:

A.java:

```
package com.masai;

public class A {

	String msg="Welcome";

}
```

Demo.java:

```
package com.masai;

public class Demo {

	public static void main(String[] args) {

		String s1="Welcome";

		A a1= new A();

		System.out.println(s1 == a1.msg);

	}

}
```

Note: String obj is an immutable object, i.e once a string object is created,we can not modify that obj, if we want to modify it by calling its method, those method will return a new String obj. instead of modifying that exixting string object.

example:

```
String s1="Welcome";

String s2= s1.concat(" to Java");

System.out.println(s1); //Welcome
System.out.println(s2); //Welcome to Java
```

Since String obj is immutable , we can not modify an existing obj, each modification will return a new obj, but if we want to get mutability, then we should use StringBuffer or StringBuilder classes.

--these both classes belongs to java.lang package.

Diff bt StringBuffer and StringBuilder:
==========

--most of the methods of StringBuilder is non-synchronized, i.e not thread-safe it will give fast performance.

--whereas most of the methods of StringBuffer is synchronized i.e thread safe, and give slow performance compare to StringBuilder.

```
synchronized void bookTicket(){
--
--
}
synchronized void getAvailibility(){
--
--
}
```

example:

```
package com.masai;
```

```
public class Demo {


	public static String reverseString(String orignalString) {

		StringBuilder sb = new StringBuilder("");



		char[] chr= orignalString.toCharArray();


		for(int i= chr.length-1; i>=0; i--) {

			sb.append(chr[i]);
		}


		return sb.toString();

	}


	public static void main(String[] args) {


		System.out.println(reverseString("welcome"));




	}

}
```