Tutorial Link https://course.testpad.chitkara.edu.in/tutorials/Java: Scanner Class/5aa2b7e45ead875d7faa99c0

TUTORIAL

Java: Scanner Class

Topics

There are various ways to read input from the keyboard, the java.util.Scanner class is one of them.

The Java Scanner class breaks the input into tokens using a delimiter that is whitespace by default. It provides many methods to read and parse various primitive values.

Java Scanner class is widely used to parse text for string and primitive types using regular expression. Java Scanner class extends Object class and implements Iterator and Closeable interfaces.

Commonly used methods of Scanner class

Following is a list of commonly used Scanner class methods:

```
Method                          Description
public String next()            it returns the next token from
the scanner.
public String nextLine()        it moves the scanner position to
the next line and returns the value as a string.
public byte nextByte()          it scans the next token as a
byte.
public short nextShort()        it scans the next token as a
short value.
public int nextInt()            it scans the next token as an
int value.
public long nextLong()          it scans the next token as a
long value.
public float nextFloat()        it scans the next token as a
float value.
```

```
public double nextDouble()       it scans the next token as a
double value.
```

The tokens may then be converted into primitive values of different types using the various `nextXxx()` methods (`nextInt()`, `nextByte()`, `nextShort()`, `nextLong()`, `nextFloat()`, `nextDouble()`, `nextBoolean()`, `next()` for `String`, and `nextLine()` for an input line). You can also use the `hasNextXxx()` methods to check for the availability of a desired input.The commonly-used constructors are as follows. You can construct a `Scanner` to parse a byte-based `InputStream` (e.g., `System.in`), a disk file, or a given `String`.

```
// Scanner piped from a disk File
public Scanner(File source) throws FileNotFoundException
public Scanner(File source, String charsetName) throws
FileNotFoundException
// Scanner piped from a byte-based InputStream, e.g., System.in
public Scanner(InputStream source)
public Scanner(InputStream source, String charsetName)
// Scanner piped from the given source string (NOT filename
string)
public Scanner(String source)
```

For examples,

```
// Construct a Scanner to parse an int from keyboard
Scanner in1 = new Scanner(System.in);
int i = in1.nextInt();

// Construct a Scanner to parse all doubles from a disk file
Scanner in2 = new Scanner(new File("in.txt"));  // need to handle
FileNotFoundException
while (in2.hasNextDouble()) {
   double d = in.nextDouble();
}

// Construct a Scanner to parse a given text string
Scanner in3 = new Scanner("This is the input text String");
while (in3.hasNext()) {
   String s = in.next();
}
```

Following is an example use of Scanner class to read input from stdin. It reads the int, string and double value as an input:

```
import java.util.Scanner;
class ScannerTest
{
 public static void main(String args[])
 {
   Scanner sc=new Scanner(System.in);  // Create the object of
Scanner class

   System.out.println("Enter your rollno");
   int rollno=sc.nextInt();          // Read an integer

   System.out.println("Enter your name");
   String name=sc.next();            // Read a String

   System.out.println("Enter your marks");
   double marks=sc.nextDouble();       // Read a double

   System.out.println("Roll Number: " + rollno + " Name: " + name
+ " Marks: " + marks);

   sc.close();
 }
}
```

Delimiter

Instead of the default white spaces as the delimiter, you can set the delimiter to a chosen regular expression via these methods:

```
public Pattern delimiter()    // Returns the current delimiter
Regexe Pattern
public Scanner useDelimiter(Pattern pattern)   // Sets the
delimiter Regexe Pattern
public Scanner useDelimiter(String pattern)
```

For example,

```java
import java.util.Scanner;
class Main
{
  public static void main(String[] args)
  {
    Scanner in = new Scanner("CODE delim QUOTIENT delim CODING delim BETTER delim 34");
    // Zero or more whitespace, followed by 'delim', followed by zero or more whitespace.
    in.useDelimiter("\\s*delim\\s*");
    // The delimiter breaks the input into tokens {"CODE", "QUOTIENT", "CODING", "BETTER", 34}.
    System.out.println(in.next());
    System.out.println(in.next());
    System.out.println(in.next());
    System.out.println(in.next());
    System.out.println(in.nextInt());
  }
}
```

The regular expression `\s*delim\s*` matches zero or more white spaces (`\s*`) followed by "delim" followed by zero or more white spaces (`\s*`). An additional backslash (\) is needed to use a backslash (\) in Java `String`'s literal.