Java 9 JShell – REPL Tool

pramodbablad August 27, 2021 0

Java 9 JShell is a REPL tool i.e Read Eval Print Loop tool through that you can evaluate Java code snippet or any business logic without compiling and running the whole Java program. Such tool is already there in other languages like Scala and Python. From Java 9, Java also supports REPL tool called JShell. Let's see how to use Java 9 Jshell to evaluate Java code.

Why JShell?

JShell gives Java developer an oppurtunity to evaluate the business logic or any small piece of Java code like statements or expressions or methods or class definitions without compiling and running the whole Java program.

Creating a Java program involves 4 steps – typing, saving, compiling and running. You have to follow these steps until you get desired output.

Using JShell, one can quickly evaluate the business logic or any statement or any expression or any method definition or any class definition without following the above steps.

For example, to see the output of `2+3` expression, you have to write a Java program like below.

```
1  public class ExpressionD
2  {
3      public static void m
4      {
5          System.out.print
6      }
7  }
```

But, in JShell, you just have to type `2+3` to see its output.

```
jshell> 2+3
$1 ==> 5
| created scratch variable $1 : int
```

JShell is not an alternative to your IDE or code editor. It is just a tool to test your code before actually writing it in an IDE or in an editor. Let's see Java 9 JShell in detail.

How To Start JShell?

To start JShell, you must have installed Java 9 in your PC. It is not available in the earlier versions of Java. To start JShell, path must be set to bin folder of the JDK 9 installation directory. If path is not set, navigate to that folder in the command prompt and type jshell. You can also set path only for current window using `set path` command. More about how to set path.

Open command prompt and type jshell. To open JShell in verbose mode use -v option.

```
C:\Users\HP>set path="C:\Program Files\Java\jdk-9.0.4\bin";

C:\Users\HP>jshell -v
| Welcome to JShell — Version 9.0.4
| For an introduction type: /help intro
```

How To Write Statements In JShell?

Statements in JShell are written normally as we do in any editor or in any IDE. But, semicolon(;) at the end of the statement is optional. i.e statements without semicolon at the end run normally.

```
jshell> System.out.println("Hello Java");
Hello Java

jshell> System.out.println("Hello Java")
Hello Java
```

How To Declare Variables In JShell?

You can declare variables of any type like int, float, long, double, String normally as we do in regular Java programs.

```
jshell> int a=11;
a ==> 11
| created variable a : int

jshell> long l
l ==> 0
| created variable l : long

jshell> double d=1.1
d ==> 1.1
| created variable d : double

jshell> float f;
f ==> 0.0
| created variable f : float
```

What are scratch variables in JShell?

In JShell, you can declare variables without names. These are called scratch variables. JShell implicitly gives names to such variables and they usually starts with $.

```
jshell> 111
$6 ==> 111
| created scratch variable $6 : int

jshell> 11.11
```

```
$7 ==> 11.11
| created scratch variable $7 : double

jshell> "Hi, I am JShell"
$8 ==> "Hi, I am JShell"
| created scratch variable $8 : String
```

How To Define Methods In JShell?

Below code snippet shows how to define methods in JShell and also shows how to call them, how to print their result and how to assign their result to a variable.

```
jshell> int add(int a, int b)
...> {
...> return a+b;
...> }
| created method add(int,int)

jshell> add(10, 20);
$10 ==> 30
| created scratch variable $10 : int

jshell> System.out.println(add(100, 200));
300

jshell> int i = add(11, 22);
i ==> 33
| created variable i : int
```

How To Define Classes In JShell?

Below code snippet shows how to define a class in JShell, how to instantiate it and how to call its methods.

```
jshell> class MathsOps
...> {
...> static int add(int a, int b)
...> {
...> return a+b;
...> }
...> int multiply(int a, int b)
...> {
```

```
...> return a*b;
...> }
...> }
| created class MathsOps

jshell> MathsOps.add(50, 70);
$2 ==> 120
| created scratch variable $2 : int

jshell> new MathsOps().multiply(20, 30);
$3 ==> 600
| created scratch variable $3 : int

jshell> MathsOps mOps = new MathsOps();
mOps ==> MathsOps@1e397ed7
| created variable mOps : MathsOps

jshell> mOps.multiply(10, 10);
$5 ==> 100
| created scratch variable $5 : int
```

How To Modify Variable, Method And Class in JShell?

You can modify definition of a variable or a method or a class in JShell. Just enter a new definition, old definition will be overwritten. (You can use /edit command to edit in JShell edit pad).

```
jshell> int add(int a, int b)
...> {
...> return a+b;
...> }
| created method add(int,int)

jshell> int add(int a, int b)
...> {
...> return a+b+10;
...> }
| modified method add(int,int)
| update overwrote method add(int,int)

jshell> class Person
...> {
...> String firstName;
...> String lastName;
...> String getFirstName()
...> {
...> return firstName;
```

```
...> }
...> String getLastName()
...> {
...> return lastName;
...> }
...> }
| created class Person

jshell> class Person
...> {
...> int ID;
...> String firstName;
...> String lastName;
...> String getDetails()
...> {
...> return "ID : "+ID+"First Name : "+firstName+"Last Name : "+lastName;
...> }
...> }
| replaced class Person
| update overwrote class Person

jshell> String welcomeNote = "Hi...";
welcomeNote ==> "Hi..."
| created variable welcomeNote : String

jshell> welcomeNote = "Hi...I am JShell. You can use me to test your Java code"
welcomeNote ==> "Hi...I am JShell. You can use me to test your Java code"
| assigned to welcomeNote : String

jshell> int salary;
salary ==> 0
| created variable salary : int

jshell> double salary;
salary ==> 0.0
| replaced variable salary : double
| update overwrote variable salary : int
```

Java 9 JShell Commands :

Below is the list of all JShell Commands with their brief description.

1) /list

This command displays all the variables, methods, classes or any other sources you have typed.

```
jshell> /list
1 : int i=10;
2 : long l;
3 : double d = 1.1;
4 : float f;
5 : String s = "Hi…I am JShell";
6 : int add(int a, int b)
{
return a+b;
}
7 : class Person
{
int ID;
String firstName;
String lastName;
String getDetails()
{
return "ID : "+ID+"First Name : "+firstName+"Last Name : "+lastName;
}
}

jshell> /list -all
s1 : import java.io.; s2 : import java.math.;
s3 : import java.net.; s4 : import java.nio.file.;
s5 : import java.util.; s6 : import java.util.concurrent.;
s7 : import java.util.function.; s8 : import java.util.prefs.;
s9 : import java.util.regex.; s10 : import java.util.stream.;
1 : int i=10;
2 : long l;
3 : double d = 1.1;
4 : float f;
5 : String s = "Hi…I am JShell";
6 : int add(int a, int b)
{
return a+b;
}
7 : class Person
{
int ID;
String firstName;
String lastName;
String getDetails()
{
return "ID : "+ID+"First Name : "+firstName+"Last Name : "+lastName;
}
}

jshell> /list -start
s1 : import java.io.; s2 : import java.math.;
s3 : import java.net.; s4 : import java.nio.file.;
```

```
s5 : import java.util.; s6 : import java.util.concurrent.;
s7 : import java.util.function.; s8 : import java.util.prefs.;
s9 : import java.util.regex.; s10 : import java.util.stream.;
s10 : import java.util.stream.*;
```

2) /edit

This command is used to edit a source you have typed in an JShell edit pad.

```
jshell> /edit add
| modified method add(int,int)
| update overwrote method add(int,int)
```

3) /drop

This command is used to delete a varaiable or a method or a class.

```
jshell> /drop add
| dropped method add(int,int)

jshell> /drop s
| dropped variable s
```

4) /save

This command saves the sources you have typed in JShell to a file.

```
jshell> /save -all C:\Users\HP\JshellDemo.txt
```

5) /open

Using this command, you can open a file as a source in JShell.

```
jshell> /open C:\Users\HP\inputFile.txt
```

6) /vars

This command lists all the declared variables.

```
jshell> /vars
| long l = 0
| double d = 1.1
| float f = 0.0
| String s = "Hi...I am JShell"
| int i = 999

jshell> /vars -all
| int i = (not-active)
| long l = (not-active)
| double d = (not-active)
| float f = (not-active)
| String s = (not-active)
| int i = (not-active)
| int i = (not-active)
| long l = 0
| double d = 1.1
| float f = 0.0
| String s = "Hi...I am JShell"
| int i = 999
```

7) /methods

It lists all the declared methods with their signatures.

```
jshell> /methods
| int add(int,int)
```

8) /types

This command lists all the declared types.

```
jshell> /types
| class Person
| class MathsOps
```

9) /imports

It lists all imported classes and libraries.

```
jshell> /imports
| import java.io.*
| import java.math.*
| import java.net.*
| import java.nio.file.*
| import java.util.*
| import java.util.concurrent.*
| import java.util.function.*
| import java.util.prefs.*
| import java.util.regex.*
| import java.util.stream.*
```

10) /history

It lists whatever you have typed in this session in JShell.

11) /reset

This command is used to reset JShell.

12) /reload

This command restarts and restores the JShell.

13) /set

This command is used to configure JShell like mode, editor, format etc.

14) /help

This command gives all information about JShell.

15) /exit

This command is used to exit JShell.

<Tab> For Auto-Complete Suggestions :

You can use <Tab> key for auto-complete suggestions. For example, if you type / and press <Tab>, JShell gives list of all commands.

```
jshell> /
/!          /?          /drop      /edit      /env       /exit
/help      /history      /imports      /list      /methods      /open
/reload      /reset      /save      /set      /types      /vars

<press tab again to see synopsis>

jshell> /
```

Also Read :

- Java 9 Interface Private Methods
- Java 9 JShell Oracle User Guide