Java 9 Repl Tutorial

examples.javacodegeeks.com/core-java/java-9-repl-tutorial/

In this example, I would like to show you how to get started with Java 9 REPL (The Java Shell: Read-Eval-Print Loop). Oracle site has excellent details of the features.

Here, I present some examples and details to get started along with some of the important features and commands of this useful feature added in Java 9. `Jshell`is a quick way for developers to test code snippets. More details can be found at JEP 222 and jdk.shell site.

As indicated in JEP 222, the motivation of `jshell`is to interactively test expressions and code within the `jshell`state. The variables and methods that are going to be tested do not need to occur within a method/ class. All inputs to jshell must match the Java Language Specification (JLS). The `jshell`tool is not intended to be an IDE, hence, graphical support and debugger are not supported.

Table Of Contents

1. Introduction

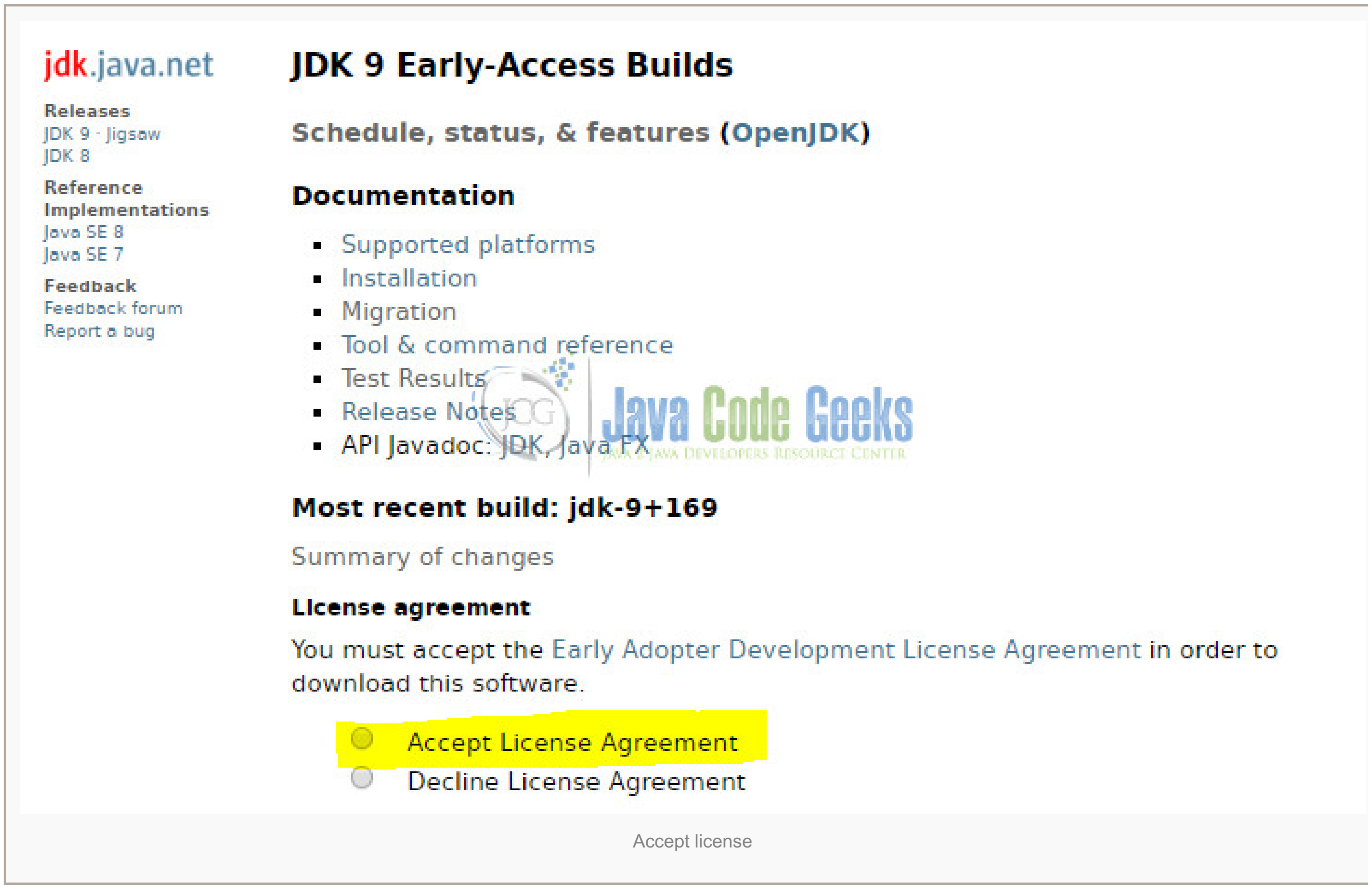
Java 9 is a major release. While writing this article, JDK 9 is currently available for early access download on the

oracle site and is expected to be released on July 27, 2017. This document attempts to summarize details of JDK9 REPL and some of the main features with this new release.
Complete list of features in Java 9 can be viewed at the oracle site.

Tip
You may skip setup sections if JDK 9 is already setup for you and jump directly to the **features section** below.

2. Getting started with Java 9

To download the currently available early access JDK or JRE 9, visit http://jdk.java.net/9/.



jdk.java.net

Releases
JDK 9 · Jigsaw
JDK 8

Reference Implementations
Java SE 8
Java SE 7

Feedback
Feedback forum
Report a bug

JDK 9 Early-Access Builds

Schedule, status, & features (OpenJDK)

Documentation

- Supported platforms
- Installation
- Migration
- Tool & command reference
- Test Results
- Release Notes
- API Javadoc: JDK, JavaFX

Most recent build: jdk-9+169

Summary of changes

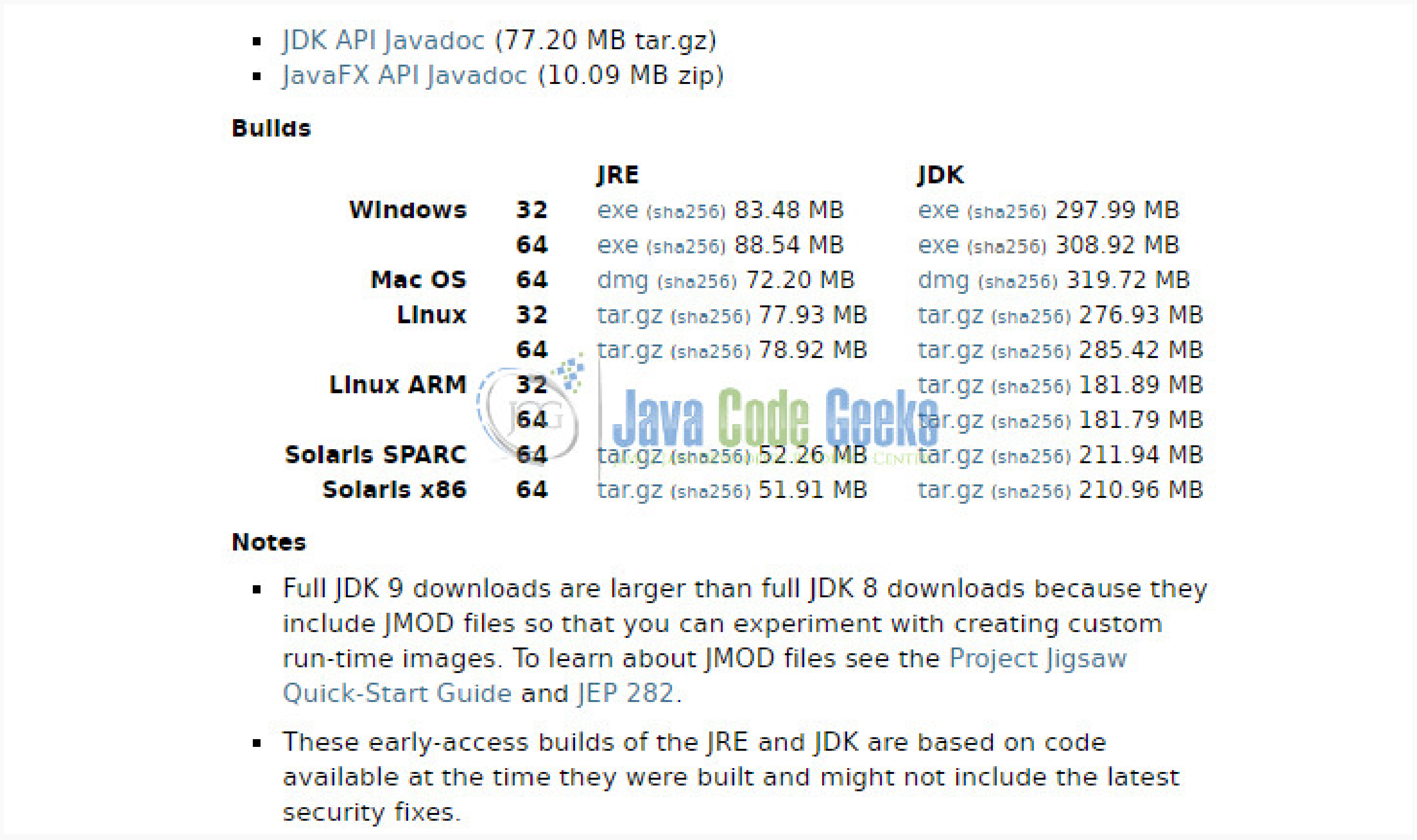
License agreement

You must accept the Early Adopter Development License Agreement in order to download this software.

[✓ Accept License Agreement]
Decline License Agreement

Accept license

As shown in the image above, at the site, accept the license agreement and proceed to the download section as shown below.



- JDK API Javadoc (77.20 MB tar.gz)
- JavaFX API Javadoc (10.09 MB zip)

Builds

		JRE	JDK
Windows	32	exe (sha256) 83.48 MB	exe (sha256) 297.99 MB
	64	exe (sha256) 88.54 MB	exe (sha256) 308.92 MB
Mac OS	64	dmg (sha256) 72.20 MB	dmg (sha256) 319.72 MB
Linux	32	tar.gz (sha256) 77.93 MB	tar.gz (sha256) 276.93 MB
	64	tar.gz (sha256) 78.92 MB	tar.gz (sha256) 285.42 MB
Linux ARM	32		tar.gz (sha256) 181.89 MB
	64		tar.gz (sha256) 181.79 MB
Solaris SPARC	64	tar.gz (sha256) 52.26 MB	tar.gz (sha256) 211.94 MB
Solaris x86	64	tar.gz (sha256) 51.91 MB	tar.gz (sha256) 210.96 MB

Notes

- Full JDK 9 downloads are larger than full JDK 8 downloads because they include JMOD files so that you can experiment with creating custom run-time images. To learn about JMOD files see the Project Jigsaw Quick-Start Guide and JEP 282.
- These early-access builds of the JRE and JDK are based on code available at the time they were built and might not include the latest security fixes.

Download JDK

Please select the appropriate OS and appropriate option for 32/ 64 bits for the OS to download the JDK/ JRE. It is also recommended to download the documentation along with the JDK/ JRE installation.

You may refer to this article to get started with Java 9 by executing a simple hello world program.

3. What is REPL?

REPL stands for read-eval-print-loop and is a shell interface for users to test code snippets. This shell interface reads the input, evaluates and prints the output (and errors if applicable). This is similar to the REPL tool available in Clojure/ Scala. This is a useful tool for testing small code snippets before moving into writing complete code in IDE.

From JEP222, `jshell`aims to provide an interactive tool to evaluate declarations, statements, and expressions of the Java programming language, together with an API so that other applications can leverage this functionality.

Code snippet written in `jshell`must correspond to any one of the below and must adhere to the Java Language Specification (JLS):

- Expression
- Statement
- Class declaration
- Interface declaration
- Method declaration
- Field declaration
- Import declaration

3.1 Jshell /help

The following section Java 9 REPL features has details of the commands on `jshell`. Before we look at the commands, below is the introduction from `jshell`received by running `/help intro` on the `jshell`prompt.

```
jshell> /help intro
|  intro
|
|  The jshell tool allows you to execute Java code, getting immediate results.
|  You can enter a Java definition (variable,          , etc),         x
method,                                        classlike:       int =  8
|  or a Java expression, like:  x + x
|  or a Java statement or                 import.
|  These little chunks of Java code are called 'snippets'.
|
|  There are also jshell commands that allow you to understand and
|  control what you are doing, like: /list
|
|  For a list of commands: /help
```

Here are the shortcuts available in `jshell`:

```
jshell> /help shortcuts
|
|  shortcuts

|

|  Supported shortcuts
include:

|

|

|                 After entering the first few letters of a Java
identifier,

|                 a jshell command, or, in some cases, a jshell command
argument,

|                 press the  key to complete the
input.

|                 If there is more than one completion, then possible completions
will be shown.

|                 Will show                    available and
documentation                              if appropriate.

|

|  Shift-
v

|                 After a complete expression, hold           pressing
down                                                  while ,

|                 then release and        , the expression will be converted
press                                "v"to

|                 a variable declaration whose type is based on the type of the
expression.

|

|  Shift-
i
```

```
|               After an unresolvable identifier, hold          pressing
down                                                        while ,
|               then release and
press                                 "i"
, and jshell will propose possible
imports
|               which will resolve the identifier based on the content of the
specified classpath.
```

Also, we can set an evaluation context to the `jshell` commands.

```
jshell> /help
context
|
|  context
|
|  These options configure the evaluation context, they can be specified
when
|  jshell is started: on the command-line, or restarted with the commands
/env,
|  /reload, or
/reset.
|
|  They are:
|       -
-         class-path
|               A list of directories, JAR
archives,
|               and ZIP archives to
search                                 for class files.
```

```
                  The list is separated with the path separator
                  (a : on unix/linux/mac, and ; on windows).
        --module-path ...
                  A list of directories, each directory
                  is a directory of modules.
                  The list is separated with the path separator
                  (a : on unix/linux/mac, and ; on windows).
        --add-modules [,...]
                  root modules to resolve in addition to the initial module.
                   can also be ALL-DEFAULT, ALL-SYSTEM,
                  ALL-MODULE-PATH.
        --add-exports /=(,)*
                  updates  to export  to ,
                  regardless of module declaration.
                   can be ALL-UNNAMED to export to all
```

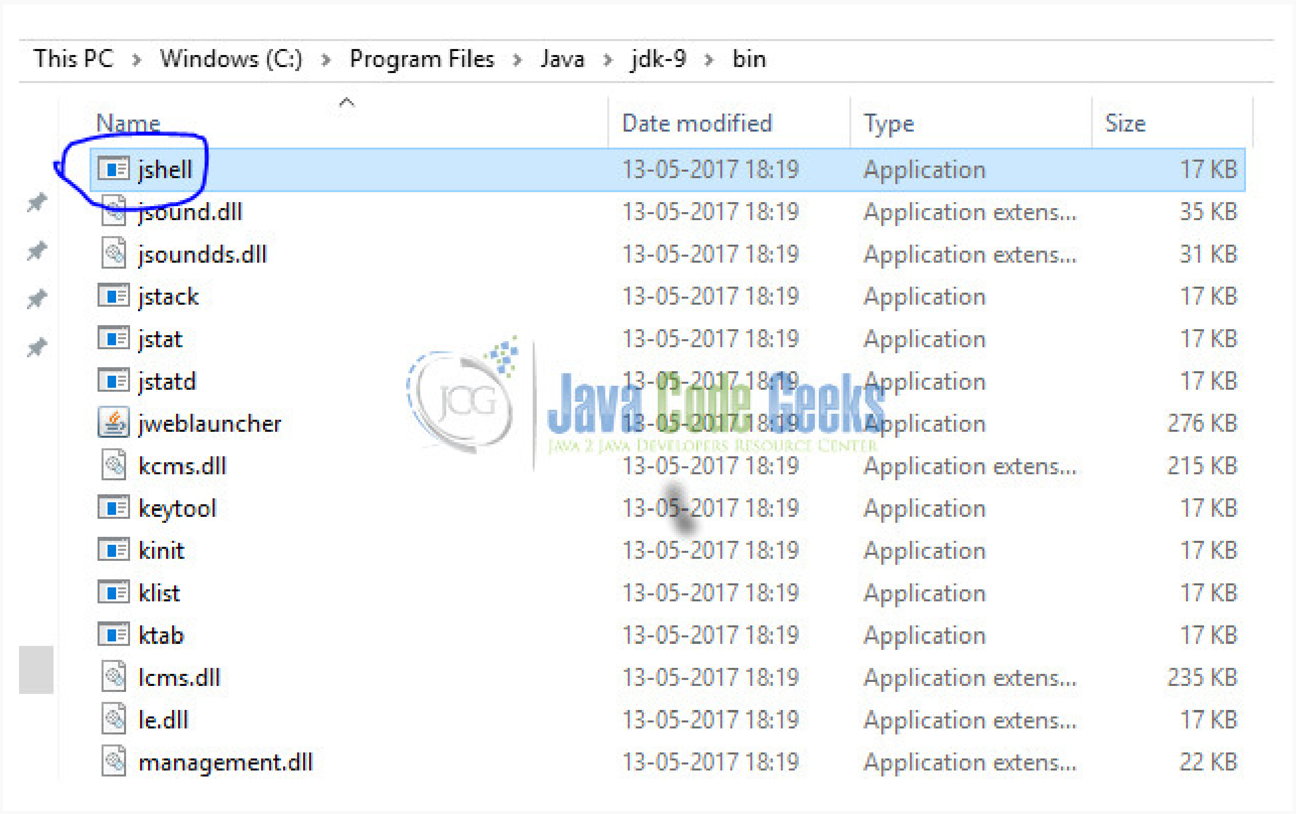
```
|                unnamed modules. In           the   is
jshell,                                         if not
|                specified (no =) then ALL-UNNAMED is
used.
|
|  On the command-line these options must have two dashes, e.g.: --module-
path
|  On jshell commands they can have one or two dashes, e.g.: -module-
path
```

All the above can be used for the commands explained in the section below.

4. Java 9 REPL features

4.1 Getting started

To open `JShell`, go to the JDK installed bin directory and click on `jshell`:

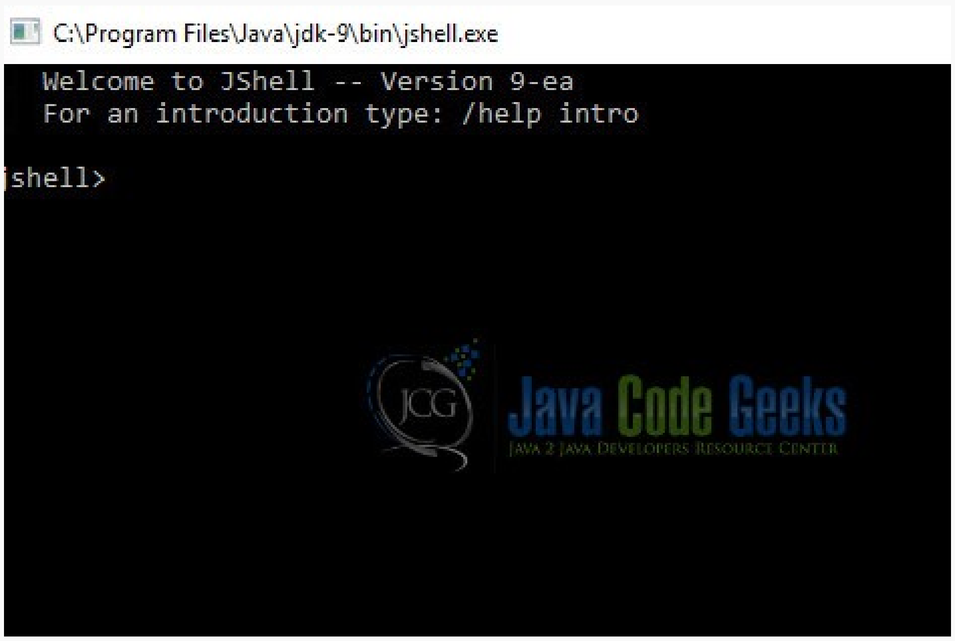


This PC › Windows (C:) › Program Files › Java › jdk-9 › bin

Name	Date modified	Type	Size
jshell	13-05-2017 18:19	Application	17 KB
jsound.dll	13-05-2017 18:19	Application extens...	35 KB
jsoundds.dll	13-05-2017 18:19	Application extens...	31 KB
jstack	13-05-2017 18:19	Application	17 KB
jstat	13-05-2017 18:19	Application	17 KB
jstatd	13-05-2017 18:19	Application	17 KB
jweblauncher	13-05-2017 18:19	Application	276 KB
kcms.dll	13-05-2017 18:19	Application extens...	215 KB
keytool	13-05-2017 18:19	Application	17 KB
kinit	13-05-2017 18:19	Application	17 KB
klist	13-05-2017 18:19	Application	17 KB
ktab	13-05-2017 18:19	Application	17 KB
lcms.dll	13-05-2017 18:19	Application extens...	235 KB
le.dll	13-05-2017 18:19	Application extens...	17 KB
management.dll	13-05-2017 18:19	Application extens...	22 KB

jshell location in JDK directory

This is how `jshell` prompt looks:



```
   Welcome to JShell -- Version 9-ea
   For an introduction type: /help intro

jshell>
```

jshell

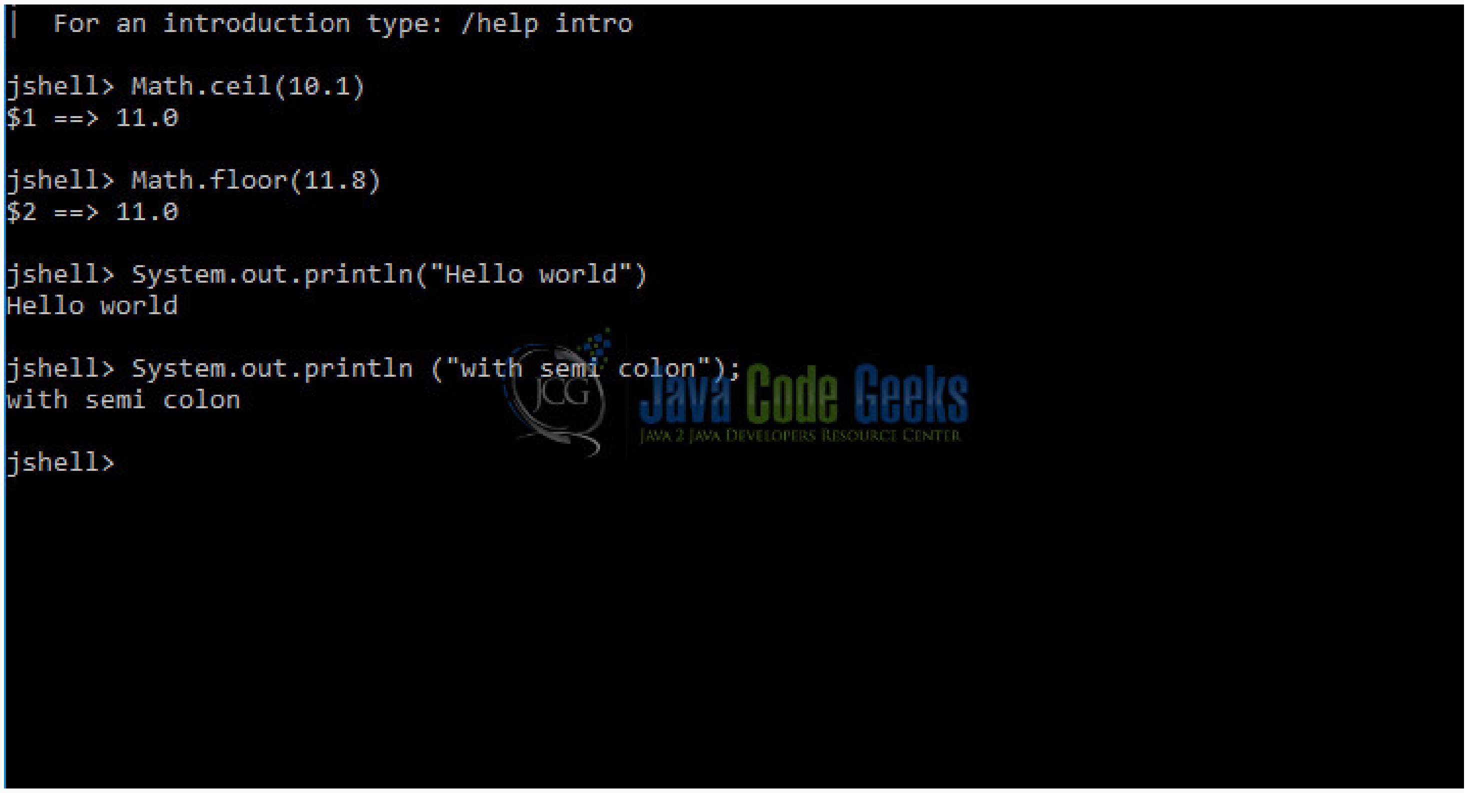
5. REPL examples

Let’s get started with some simple examples to get started with `jshell`.

5.1 Examples with expressions

Let’s start with basic `java.lang.Math` functions and `System.out.println` calls as shown in the snippets below. First we call `Math.ceil` method, followed by `Math.floor` method. These are standard methods in `java.lang.Math` class. `/vars` command lists all the variables set so far. This prints the two variables created when the `Math`methods were executed. `System.out.println`calls print the value being printed.

```
jshell> Math.ceil(10.1)
$1 ==>11.0

jshell> Math.floor(11.6)
$2 ==>11.0

jshell> /vars
double $1 =11.0
double $2 =11.0

jshell> System.out.println("Hello world")
Hello world

jshell> System.out.println ("with semi colon");
with semi colon
```



```
|  For an introduction type: /help intro

jshell> Math.ceil(10.1)
$1 ==> 11.0

jshell> Math.floor(11.8)
$2 ==> 11.0

jshell> System.out.println("Hello world")
Hello world

jshell> System.out.println ("with semi colon");
with semi colon

jshell>
```

Examples with expressions

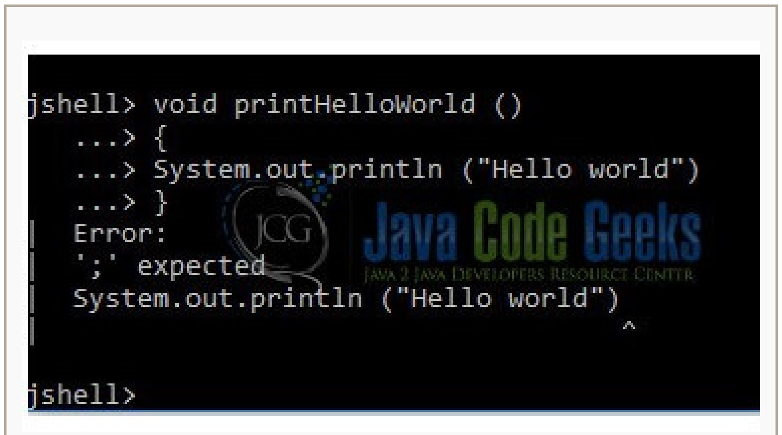
As you can see, we can run expressions on `jshell`and values of variables can be viewed using the `/var` command.

5.2 Examples with method

Let's now move on to a complete method on `jshell`.

We are going to type a simple method that prints "Hello World". The method is named `printHelloWorld`and makes a `System.out.println` call.

```
jshell>void printHelloWorld()
    ...> {
    ...> System.out.println("Hello World")
    ...> }
```



```
jshell> void printHelloWorld ()
   ...> {
   ...> System.out.println ("Hello world")
   ...> }
|  Error:
|  ';' expected
|  System.out.println ("Hello world")
|                                    ^

jshell>
```

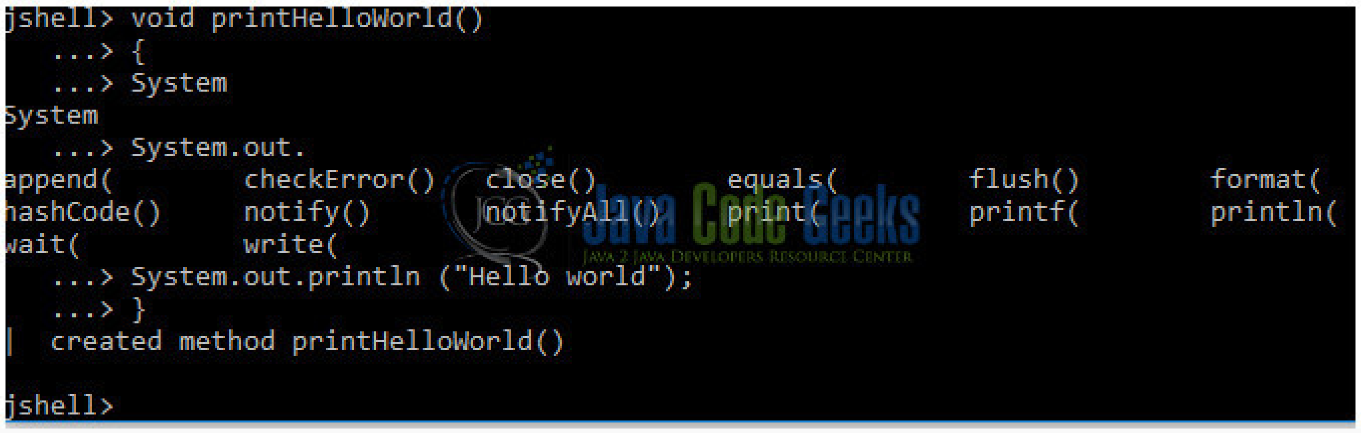
Examples with method

```
| Error:
| ';' expected
| System.out.println("Hello World")
|                                  ^
```

Oops, we forgot a semi-colon! Let’s run it again with the semi-colon in place.

```
jshell>void printHelloWorld()
    ...> {
    ...> System.out.println("Hello World");
    ...>
    }
```

For quick typing, you may also hit tab button to get all possible completions.



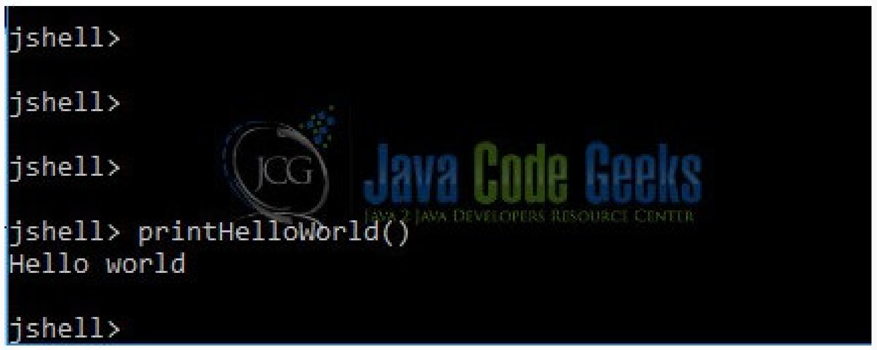
```
jshell> void printHelloWorld()
   ...> {
   ...> System
System
   ...> System.out.
append(        checkError()   close()        equals(        flush()        format(
hashCode()     notify()       notifyAll()    print(         printf(        println(
wait(          write(
   ...> System.out.println ("Hello world");
   ...> }
|  created method printHelloWorld()

jshell>
```

jshell typing completions

Once this new method has been created, we can invoke it to see how it works.

```
jshell> printHelloWorld()
```

```
Hello World
```



```
jshell>

jshell>

jshell>

jshell> printHelloWorld()
Hello world

jshell>
```

jshell method

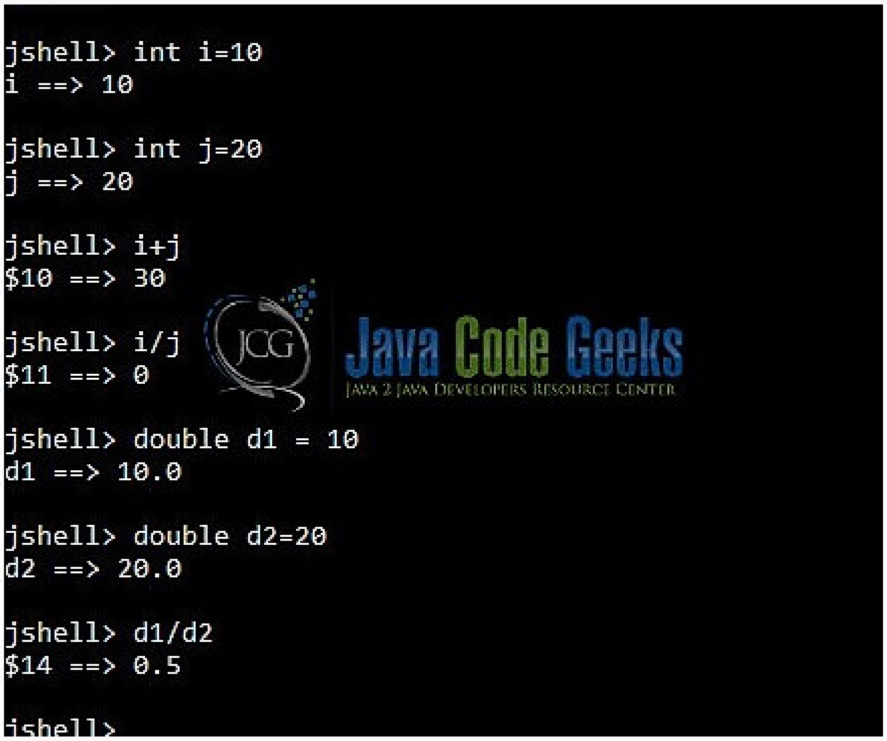
5.3 Examples with variables

To test variables, let’s try the below commands that assign a value to variables `i, j` and then computes their sum `(i+j)`. This is followed by printing `i` divided by `j` `(i/j)` . Then, we assign two double variables `d1`and `d2`and compute `d1 divided by d2` `(d1/d2)` .

```
jshell>int i=10
```

```
i
==>  10


jshell>int j=20
j
==>  20


jshell>
i+j
$10 ==>30


jshell>
i/j
$11 ==>0


                  d1
jshell>double =    10
d1
==>   10.0


jshell>double d2=20
d2
==>   20.0


jshell> d1/d2
```

```
$14 ==>0.5
```



```
jshell> int i=10
i ==> 10

jshell> int j=20
j ==> 20

jshell> i+j
$10 ==> 30

jshell> i/j
$11 ==> 0

jshell> double d1 = 10
d1 ==> 10.0

jshell> double d2=20
d2 ==> 20.0

jshell> d1/d2
$14 ==> 0.5

jshell>
```

Example with variables

As you can see, `jshell`is a simple tool to test out variables and variable assignment.

5.4 Example with class

To test a class on `jshell`, let's try the below code that creates a class `Employee`with attributes `empId, name, salary` . The class has a parameterized `constructor`and overridden `toString`method.

```
jshell>public class Employee
...> {
...> String empId;
...> String name;
...> Integer salary;
...>public Employee (String empId, String name, Integer salary)
```

```
...>
{
```

```
...>this.empId=empId;
```

```
        .name =
...>thisname;
```

```
        .salary =
...>thissalary;
```

```
...>
}
```

```
           String toString
...>public ()
```

```
...>
{
```

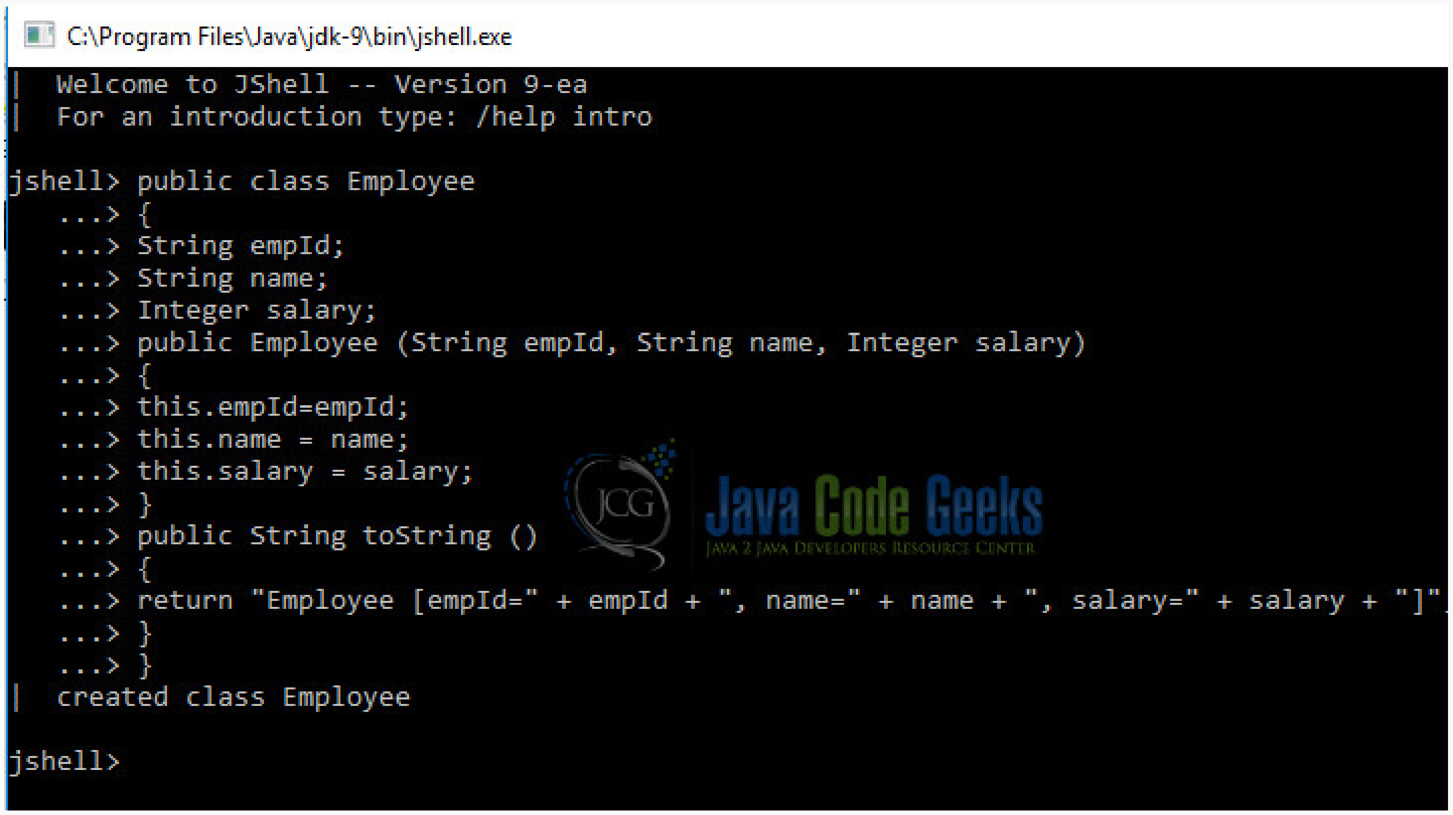
```
           "Employee           + empId  ",       + name  ",          + salary
...>return [empId="           +        name="    +       salary="    +          "]";
```

```
...>
}
```

```
...>
}
```

This gives the below output:

```
| created class
Employee
```



```
C:\Program Files\Java\jdk-9\bin\jshell.exe
|  Welcome to JShell -- Version 9-ea
|  For an introduction type: /help intro

jshell> public class Employee
   ...> {
   ...> String empId;
   ...> String name;
   ...> Integer salary;
   ...> public Employee (String empId, String name, Integer salary)
   ...> {
   ...> this.empId=empId;
   ...> this.name = name;
   ...> this.salary = salary;
   ...> }
   ...> public String toString ()
   ...> {
   ...> return "Employee [empId=" + empId + ", name=" + name + ", salary=" + salary + "]"
   ...> }
   ...> }
|  created class Employee

jshell>
```

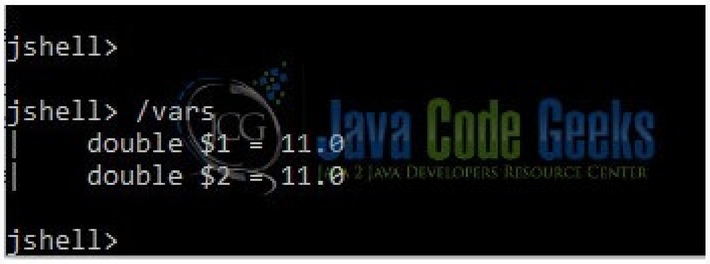
Example with class

In effect, as we have seen in the sections above, we can test any of the below in REPL: expressions, methods, variables, class.

6. Commands

6.1 /var command

To see all the variables created so far, type `/var`

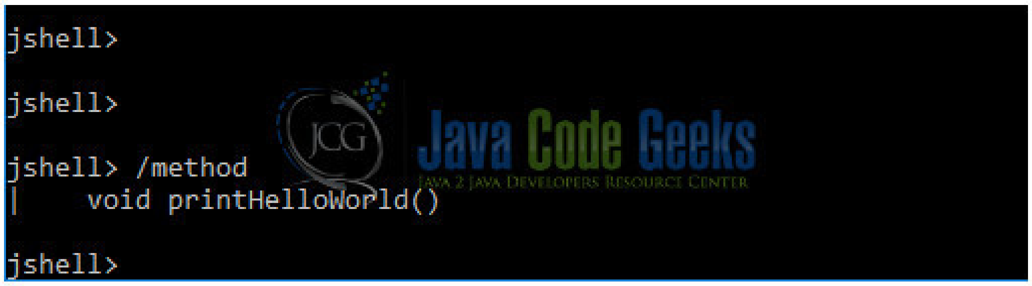


```
jshell>

jshell> /vars
|    double $1 = 11.0
|    double $2 = 11.0

jshell>
```

/var command

6.2 /method command

To see all the methods created so far, type `/method`

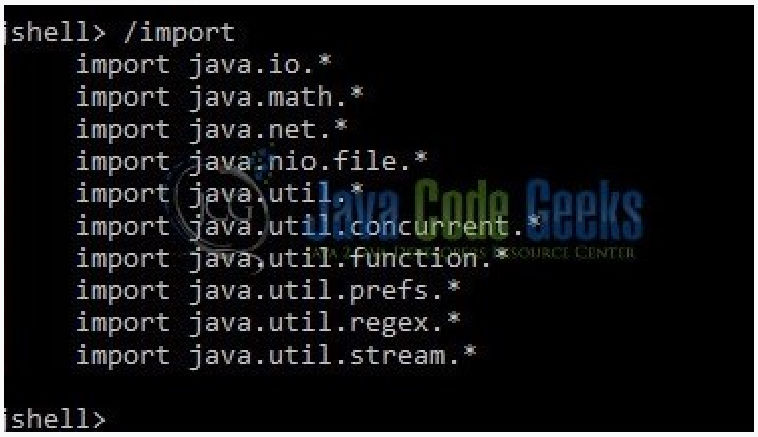


```
jshell>

jshell>

jshell> /method
|    void printHelloWorld()

jshell>
```

/method command

6.3 /import command

To see all imports that are included by default, type `/import`

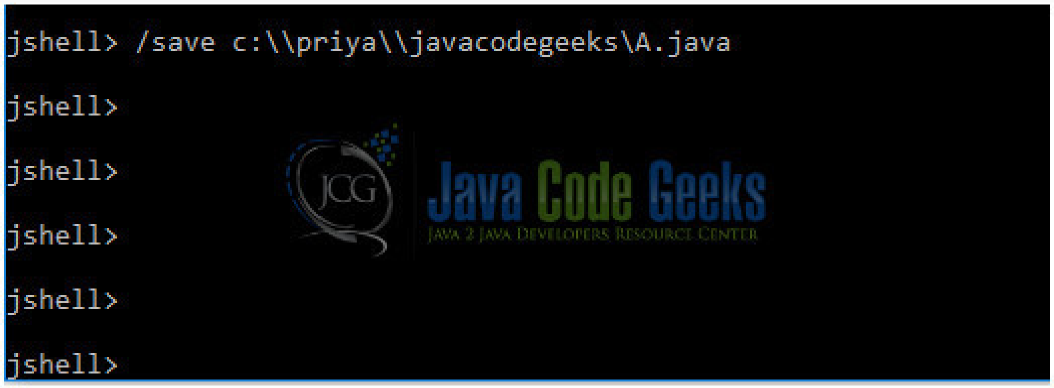


```
jshell> /import
     import java.io.*
     import java.math.*
     import java.net.*
     import java.nio.file.*
     import java.util.*
     import java.util.concurrent.*
     import java.util.function.*
     import java.util.prefs.*
     import java.util.regex.*
     import java.util.stream.*

jshell>
```

/import command

6.4 /save command

To save the history, type `/save filename`

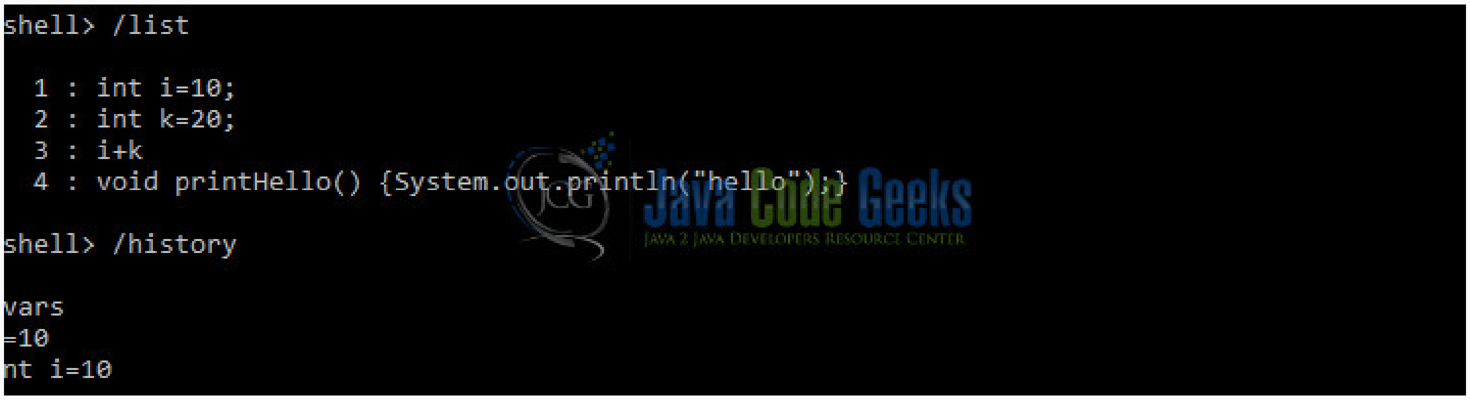


```
jshell> /save c:\\priya\\javacodegeeks\A.java

jshell>

jshell>

jshell>

jshell>

jshell>
```

/save command

6.5 /list and /history commands

To view the list of all snippets created and history of command input try `/list and /history` respectively.

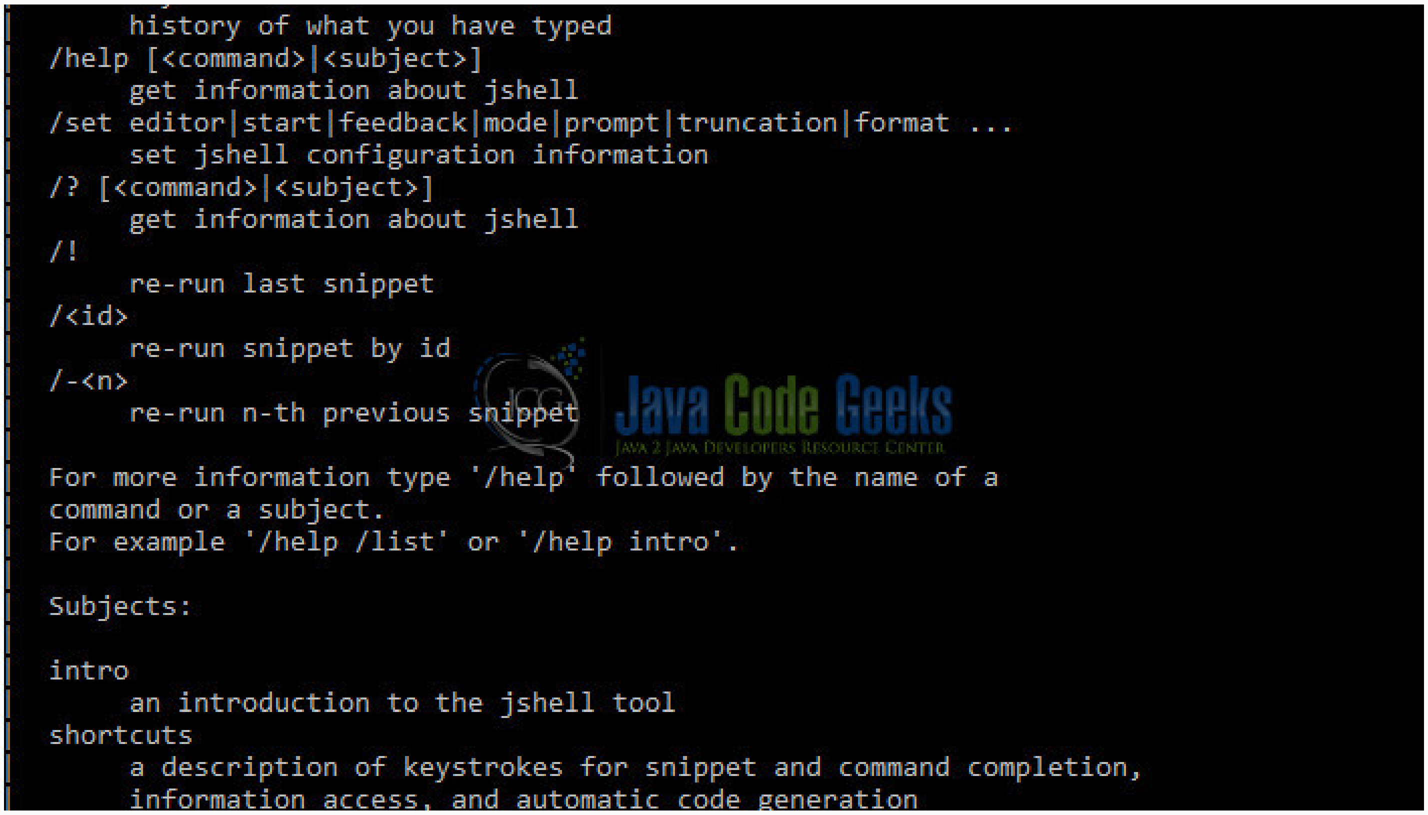


```
shell> /list

  1 : int i=10;
  2 : int k=20;
  3 : i+k
  4 : void printHello() {System.out.println("hello");}

shell> /history

vars
=10
nt i=10
```

/list and /history commands

6.6 /help command

To view all commands type `/help`

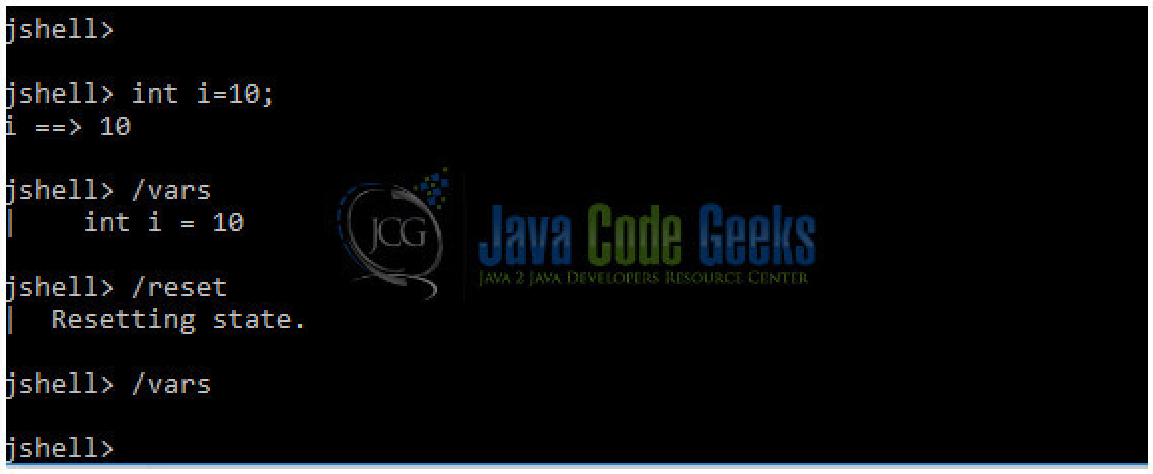


```
|       history of what you have typed
|   /help [<command>|<subject>]
|       get information about jshell
|   /set editor|start|feedback|mode|prompt|truncation|format ...
|       set jshell configuration information
|   /? [<command>|<subject>]
|       get information about jshell
|   /!
|       re-run last snippet
|   /<id>
|       re-run snippet by id
|   /-<n>
|       re-run n-th previous snippet
|
|   For more information type '/help' followed by the name of a
|   command or a subject.
|   For example '/help /list' or '/help intro'.
|
|   Subjects:
|
|   intro
|       an introduction to the jshell tool
|   shortcuts
|       a description of keystrokes for snippet and command completion,
|       information access, and automatic code generation
```

/help command

6.7 /reset command

To reset state, type `/reset`



```
jshell>

jshell> int i=10;
i ==> 10

jshell> /vars
|    int i = 10

jshell> /reset
|  Resetting state.

jshell> /vars

jshell>
```

/reset command

6.8 /exit command

To exit, type `/exit`

7. When to use REPL?

REPL `jshell`is a great way to get started with JDK 9 without needing eclipse or a complete working environment. Simple expressions, methods and classes can be tested on command line. We expect this tool to be very useful for new developers.

However, whether REPL will replace IDEs like IntelliJ or Eclipse seems unlikely. Nevertheless, for new developers who need to try out some language features this could fit their needs well.

8. Summary

This article aims to provide a start to Java 9 REPL features. JDK 9 has some exciting new features and REPL promises to change how we currently write java code by allowing us to test as we go.

9. References

https://docs.oracle.com/javase/9/whatsnew/toc.htm
https://www.infoq.com/news/2016/09/JavaOne-2016-Keynote-JShell
http://openjdk.java.net/jeps/222