

# 25.6 Discovery with JShell Auto-Completion

[*Note:* This section may be read after studying [Chapter 3](#), Introduction to Classes, Objects, Methods and Strings, and completing [Section 25.5](#).]

JShell can help you write code. When you partially type the name of an existing class, variable or method then press the *Tab* key, JShell does one of the following:

- If no other name matches what you've typed so far, JShell enters the rest of the name for you.
- If there are multiple names that begin with the same letters, JShell displays a list of those names to help you decide what to type next—then you can type the next letter(s) and press *Tab* again to complete the name.
- If no names match what you typed so far, JShell does nothing and your operating system's alert sound plays as feedback.

Auto-completion is normally an IDE feature, but with JShell it's IDE independent.

Let's first list the snippets we've entered since the last `/reset` (from [Section 25.5](#)):

---

```
jshell> /list

1 : public class Account {
    private String name;
```

```
public void setName(String name) {
    this.name = name;
}

public String getName() {
    return name;
}

2 : Account account;
3 : account = new Account()
4 : new Account()
5 : account.setName("Amanda")
6 : account.getName()
7 : System.out.println(account.getName())
8 : String name = account.getName();
```

```
jshell>
```



## 25.6.1 Auto-Completing Identifiers

The only variable declared so far that begins with lowercase "a" is **account**, which was declared in snippet 2. Auto-completion is case sensitive, so "a" does not match the class name **Account**. If you type "a" at the **jshell>** prompt:

---

```
jshell> a
```



then press *Tab*, JShell auto-completes the name:

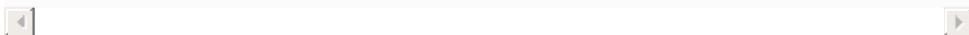
---

```
jshell> account
```



If you then enter a dot:

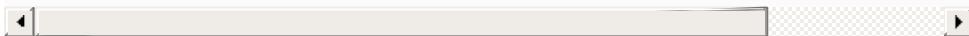
```
jshell> account.
```



then press *Tab*, JShell does not know what method you want to call, so it displays a list of everything—in this case, all the methods—that can appear to the right of the dot:

```
jshell> account.  
equals(      getClass()    getName()    hashCode()  
notifyAll()  setName()    toString()   wait()
```

```
jshell> account.
```



and follows the list with a new `jshell>` prompt that includes what you've typed so far. The list includes the methods we declared in class `Account` (snippet 1) *and* several methods that all Java classes have (as we discuss in [Chapter 9](#)). In the list of method names

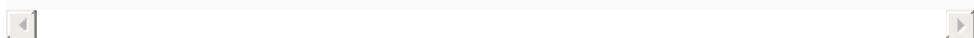
- those followed by "( )" are methods that do not require arguments and
- those followed only by "(" are methods that either require at least one argument or that are so-called *overloaded methods*—multiple methods with the same name, but different parameter lists (discussed in [Section 6.11](#)).

Let's assume you want to use `Account`'s `setName` method

to change the name stored in the `account` object to "John". There's only one method that begins with "s", so you can type `s` then *Tab* to auto-complete `setName`:

---

```
jshell> account.setName(
```

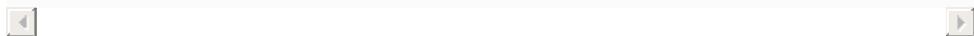


JShell automatically inserts the method call's opening left parenthesis. Now you can complete the snippet as in:

---

```
jshell> account.setName("John")
```

```
jshell>
```



## 25.6.2 Auto-Completing JShell Commands

Auto-completion also works for JShell commands. If you type `/` then press *Tab*, JShell displays the list of JShell commands:

---

```
jshell> /
!          /?          /drop        /edit        /env
/help      /history    /imports     /list        /meth
/reload    /reset      /save       /set         /type

<press tab again to see synopsis>

jshell> /
```



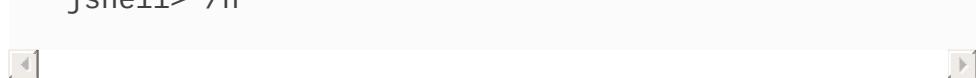
If you then type `h` and press *Tab*, JShell displays only the commands that start with `/h`:

---

```
jshell> /h
/help      /history

<press tab again to see synopsis>

jshell> /h
```



Finally, if you type "`i`" and press *Tab*, JShell auto-completes `/history`. Similarly, if you type `/l` then press *Tab*, JShell auto-completes the command as `/list`, because only that command starts with `/l`.