selsoft
build better, deliver faster

Java ile Nesne-Merkezli Programlamaya Giriş

Ek 3 - JShell

Eğitmen:

Akın Kaldıroğlu

Çevik Yazılım Geliştirme ve Java Uzmanı

Konular

- **JShell**
  - JShell Komutları
  - Kaynaklar

JShell

selsoft

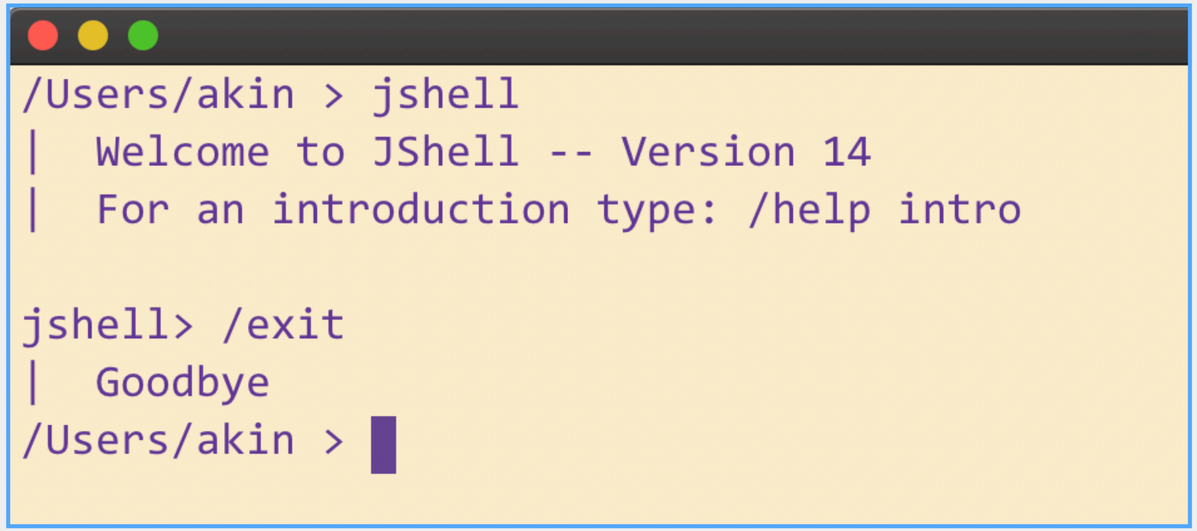
build better, deliver faster

JShell

- JShell, Java'yı öğrenmeyi kolaylaştırmak amacıyla düşünülmüş bir komut satırı arayüzdür.
- JShell bir **Read-Evaluate-Print Loop** (**REPL**) aracıdır, yani adım adım kod yazıp, çalıştırıp, sonucu görmeyi ve bu şekilde devam etmeyi sağlar.
- Java 9 ile birlikte gelmiştir.

Başlatmak ve Kapatmak

- JShell, JDK kurulumunun bin dizinindeki **`jshell`** komutu ile başlatılır.
- JShell başlarken varsayılan bir script çalıştırır, bununla en temel paketler import edilir.
- **`/exit`** komutuyla da sonlandırılır.



Yaz - Çalıştır

- Denemek amacıyla normalde **`main`** metot içine yazılıp derlenerek çalıştırılabilen tek satırlık kod JShell'de hızlıca çalıştırabilir.
- Satır sonuna "`;`" koymaya da gerek yoktur.

```
/Users/akin > jshell
|  Welcome to JShell -- Version 14
|  For an introduction type: /help intro

jshell> System.out.println("Selam");
Selam

jshell> System.out.println("Selam")
Selam

jshell> "Selam"
$3 ==> "Selam"

jshell> System.out.println("Selam)
|  Error:
|  unclosed string literal
|  System.out.println("Selam)
|                     ^

jshell> int i
i ==> 0

jshell> i
i ==> 0

jshell> int j = 6
j ==> 6

jshell>
```

Değişken Tanımlama

- JShell'de değişken tanımlanabilir.
- Değişkenlere doğrudan ulaşıldığında değeri basılır.
- JShell bir değer üreten ve bir değişkene atanmayan her işlemin sonucunu, kendi oluşturduğu ve "**$**" ile başlayan isme sahip bir değişkene atar ve bunu paylaşır.
  - Bu isimle o değişkene ulaşılmak mümkündür.

```
/Users/akin > jshell
|  Welcome to JShell -- Version 14
|  For an introduction type: /help intro

jshell> int i = 5
i ==> 5

jshell> String s = "selam"
s ==> "selam"

jshell> 3 + 2
$3 ==> 5

jshell> "Merhaba"
$4 ==> "Merhaba"

jshell> Math.sqrt(2)
$5 ==> 1.4142135623730951

jshell> $4
$4 ==> "Merhaba"

jshell>
```

Metot Çağırma ve Tanımlama

- JShell’de metot çağrısı yapılabilir, döndürdükleri değerler alınabilir ve yeni metot tanımlanabilir.
- Java API’sinde hali hazırda var olan metotlar yanıda JShell’de tanımlanan metotlar da çağrılabilir.

```
/Users/akin > jshell
|  Welcome to JShell -- Version 14
|  For an introduction type: /help intro

jshell> double d = 5
d ==> 5.0

jshell> double dSquared = Math.sqrt(d)
dSquared ==> 2.23606797749979

jshell> String selamSoyle(String kime){
   ...> return "Selam " + kime + " :)";
   ...> }
|  created method selamSoyle(String)

jshell> selamSoyle("Mihrimah")
$4 ==> "Selam Mihrimah :)"

jshell> String s = selamSoyle("İsmail")
s ==> "Selam İsmail :)"

jshell> $4
$4 ==> "Selam Mihrimah :)"

jshell>
```

Tip Tanımlama

- JShell'de **`class`**, **`interface`** ve **`enum`** cinsinden tipler tanımlanabilir,
- Tanımlanan tiplerin nesneleri oluşturulabilir.
- Nesnelerin değişkenlerine ulaşılabilir, metotları çağrılabilir.

java

```
/Users/akin > jshell
|  Welcome to JShell -- Version 14
|  For an introduction type: /help intro

jshell> class Person{
   ...> String name;
   ...> int age;
   ...>
   ...> public String toString(){
   ...> return "Person info: Name: " + name + " age: " + age;
   ...> }
   ...> }
|  created class Person

jshell> Person p = new Person()
p ==> Person info: Name: null age: 0

jshell> p.name = "Zeynep"
$3 ==> "Zeynep"

jshell> p.age = 20
$4 ==> 20

jshell> p
p ==> Person info: Name: Zeynep age: 20

jshell>
```

Dış Kodlara Erişim

- JShell'i çalıştırırken classpath bilgisini geçerek dış kodlara da ulaşmak mümkündür.

```
jshell --class-path <classpath>
```

- Bu şekilde verilen classpathde ulaşılabilen bütün tipler JShell'e `import` edildikten sonra kullanılabilir hale gelir.

```
java
/Users/akin > jshell --class-path "/Users/akin/Desktop/Selam Ornegi/bin"
|  Welcome to JShell -- Version 14
|  For an introduction type: /help intro

jshell> import a.Selam

jshell> Selam nesne = new Selam()
nesne ==> a.Selam@26a1ab54

jshell> nesne.selamSoyle("Halil")
$3 ==> "Selam Halil :)"

jshell>
```

selsoft

build better, deliver faster

JShell Komutları

Komutlar - I

- JShell'e has komutlar "`/`" ile çalıştırılır.
- Örneğin, **`/vars`**, **`/methods`** ve **`/types`** sırasıyla shellde tanımlanmış değişkenler, metotlar ve tipler hakkında bilgi verir.
- **`/list`** o ana kadar çalıştırılan tüm kodları listeler.
- **`/list -all`** o ana kadar girilen tüm komutları, shell başlarken çalışan varsayılan kodlarla beraber listeler.

Ana Komutlar

- `/<tab>`: Var olan komutları listeler.
- `/!`: Son çalıştırılan kod bir daha çalıştırır.
- `/?` ve `/help`: Yardımı basar.
- `CTRL-L`: Komut satırını temizler.
- Ayrıca JShell `<tab>` ile komut tamamlama özelliğine de sahiptir.

selsoft

build better, deliver faster

Kaynaklar

Kaynaklar

- Kaynaklar
  - https://docs.oracle.com/javase/9/jshell/introduction-jshell.htm
  - http://cr.openjdk.java.net/~rfield/tutorial/JShellTutorial.html

Soru ve Cevap Zamanı!

selsoft
build better, deliver faster

info@selsoft.com.tr
selsoft.com.tr

selsoft

build better, deliver faster

Ödevler

Ödevler - I

1. JShell’de değişkenler tanımlayın,
2. JShell’de `3`, `5` gibi sayısal değerlerle `+`, `*` gibi basit aritmetik işlemler yapın.
3. JShell’de değişkenler tanımlayıp, bunlarla `+`, `*` gibi basit aritmetik işlemler yapın.
4. JShell’de metotlar tanımlayıp çağırın.

Ödevler - II

4. Daha önce yazdığımız **`Selam`** sınıfını (**`Selam.java`** dosyasında) JShell'de yazın ve sonrasında (daha önce yazdığımız **`SelamTest`** sınıfında olduğu gibi)

```
Selam nesne = new Selam();
```

ile **`Selam`** sınıfının bir nesnesini oluşturup, bu nesne üzerinde **`selamSoyle()`** metodunu farklı **`String`** argümanlarla çalıştırın, cevapları gözlemleyin.

Bölüm Sonu

Soru ve Cevap Zamanı!

selsoft
build better, deliver faster

info@selsoft.com.tr
selsoft.com.tr