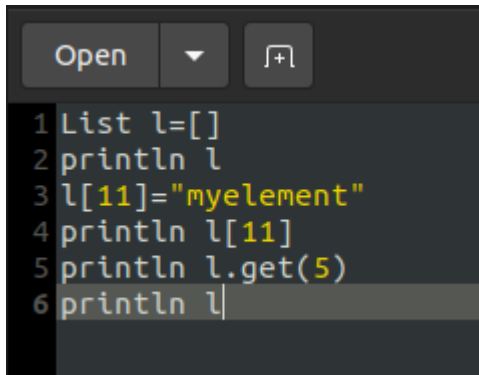


## Groovy Collections Exercise

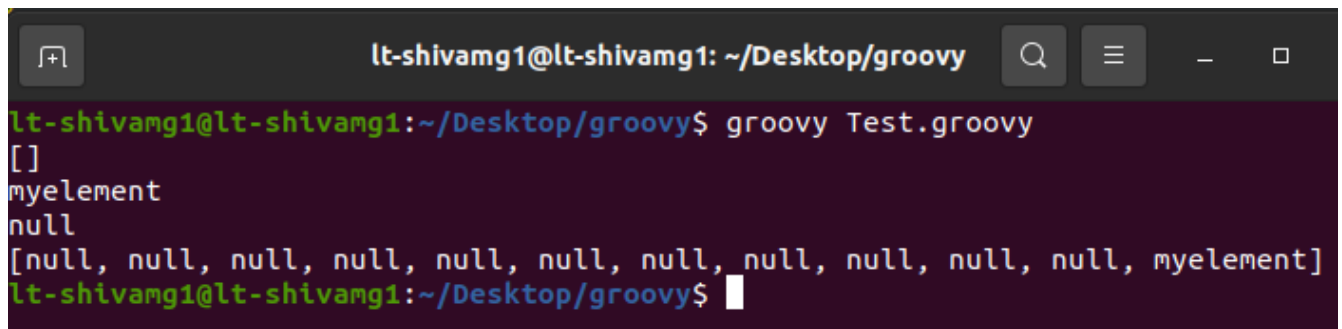
0. Initialize an empty list and give the output of the following code:

```
l[11]= "myelement"  
println l[11]  
println l.get(5)  
println l
```

Ans.



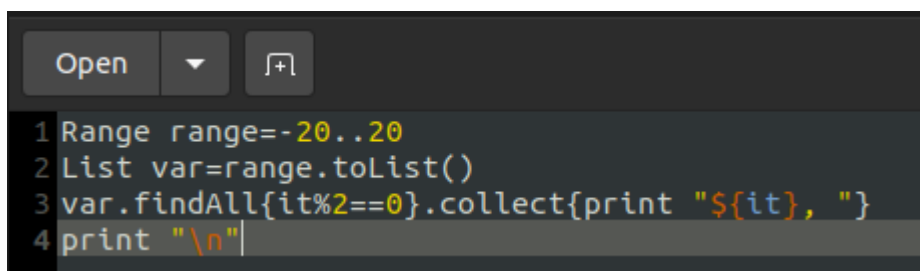
```
1 List l=[]  
2 println l  
3 l[11]="myelement"  
4 println l[11]  
5 println l.get(5)  
6 println l
```



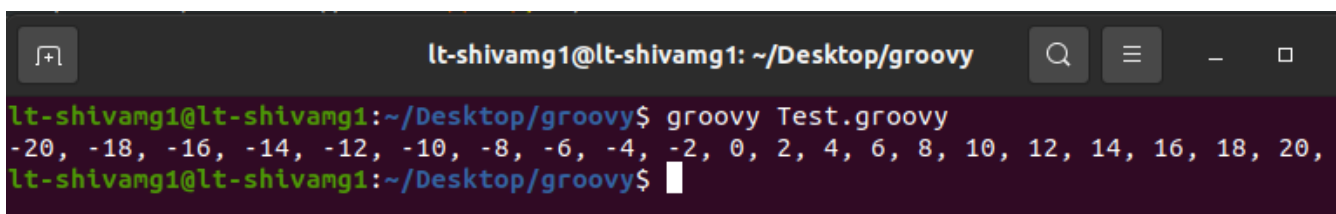
```
lt-shivamg1@lt-shivamg1: ~/Desktop/groovy  
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy  
[]  
myelement  
null  
[null, null, null, null, null, null, null, null, null, null, null, myelement]  
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```

1. Initialize a list using a range and find all elements which are even

Ans.



```
1 Range range=-20..20  
2 List var=range.toList()  
3 var.findAll{it%2==0}.collect{print "${it}, "  
4 print "\n"
```



```
lt-shivamg1@lt-shivamg1: ~/Desktop/groovy  
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy  
-20, -18, -16, -14, -12, -10, -8, -6, -4, -2, 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20,  
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```

2. Create a set from a list containing duplicate elements. What do you observe? How can you achieve the same result without converting a list to a set?

Ans. By creating a set from a list, we observe that the created set does not have any duplicate elements present in the list. This can be achieved without converting the list to a set by using the `.unique()` method.

```
Open ▼ [+]  
1 List list=[1,2,435,34,32,45,45,2,56,5,8,8,6,7,45,56]  
2 println "As set:"  
3 println list as Set  
4 println "Using unique method:"  
5 println list.unique()
```

```
lt-shivamg1@lt-shivamg1: ~/Desktop/groovy  
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy  
As set:  
[1, 2, 435, 34, 32, 45, 56, 5, 8, 6, 7]  
Using unique method:  
[1, 2, 435, 34, 32, 45, 56, 5, 8, 6, 7]  
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```

3. Given two lists [11,12,13,14] and [13,14,15], how would we obtain the list of items from the first that are not in the second?

Ans.

```
Open ▼ [+]  
1 List list1=[11,12,13,14]  
2 List list2=[13,14,15]  
3 println list1  
4 println list2  
5 println list1-list2
```

```
lt-shivamg1@lt-shivamg1: ~/Desktop/groovy  
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy  
[11, 12, 13, 14]  
[13, 14, 15]  
[11, 12]  
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```

4. Find whether two lists have a common element or not.

Ans.

```
Test.groovy
~/Desktop/groovy

1 List list1=[11,12,13,14]
2 List list2=[13,14,15]
3 println list1
4 println list2
5 if(list1.disjoint(list2)) {
6     println "the two lists do not have any common elements"
7 } else {
8     println "the two lists have common elements ${list1.intersect(list2)}"
9 }
```

```
lt-shivamg1@lt-shivamg1: ~/Desktop/groovy

lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy
[11, 12, 13, 14]
[13, 14, 15]
the two lists have common elements [13, 14]
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```

5. Remove all records from a list whose index is odd.

Ans.

```
1 List list=[1,3,6,6,3,3,1,6,98,3,2,5,7,8]
2 println list
3 List list2=[]
4 list.eachWithIndex {value,index->
5     if(index%2==0) {
6         list2.add(value)
7     }
8 }
9 println list2
```

```
lt-shivamg1@lt-shivamg1: ~/Desktop/groovy

lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy
[1, 3, 6, 6, 3, 3, 1, 6, 98, 3, 2, 5, 7, 8]
[1, 6, 3, 1, 98, 2, 7]
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```

6. Consider the following list:

[1,2,3, "element",0.3,[2,4,6],0..10]

Print the class name of each element. What's the output of the following statement?

list.get(6).get(9)

Ans.

```
Open ▼ [+]  
1 List list=[1,2,3,"element",0.3,[2,4,6],0..10]  
2 list.each{println it.class}  
3 println list.get(6)  
4 println list.get(6).get(9)|
```

```
[+] lt-shivamg1@lt-shivamg1: ~/Desktop/groovy  
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy  
class java.lang.Integer  
class java.lang.Integer  
class java.lang.Integer  
class java.lang.String  
class java.math.BigDecimal  
class java.util.ArrayList  
class groovy.lang.IntRange  
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
9  
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```

7. Sort the given list in descending order having distinct elements:

[14,12,11,10,16,15,12,10,99,90,14,16,35]

Ans.

```
Open ▼ [+]  
1 List list=[14,12,11,10,16,15,12,10,99,90,14,16,35]  
2 println list  
3 List list2=list.unique()  
4 println list2.sort().reverse()|
```

```
[+] lt-shivamg1@lt-shivamg1: ~/Desktop/groovy  
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy  
[14, 12, 11, 10, 16, 15, 12, 10, 99, 90, 14, 16, 35]  
[99, 90, 35, 16, 15, 14, 12, 11, 10]  
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```

8. Consider a class Employee with following details:

\* Name

\* Age

\* Salary

Create a list consisting of 10 Employee objects.

- Get a list of employees who earn less than 5000
- Get the name of the youngest employee and the oldest employee
- Get the employee with maximum salary
- Get the list of names of all the employees

Ans.

```
Open ▼ [icon] Test ~/Des

1 class Employee {
2     String name; int age; int salary
3     Employee(String name,int age,int salary) {
4         this.name=name
5         this.age=age
6         this.salary=salary
7     }
8     public String toString() {
9         return "["+this.name+","+this.age+","+this.salary+"]"
10    }
11 }
12
13 Employee e1=new Employee("Emp1",22,5000)
14 Employee e2=new Employee("Emp2",21,10000)
15 Employee e3=new Employee("Emp3",19,3000)
16 Employee e4=new Employee("Emp4",60,50000)
17 Employee e5=new Employee("Emp5",35,20000)
18 Employee e6=new Employee("Emp6",68,45000)
19 Employee e7=new Employee("Emp7",20,3000)
20 Employee e8=new Employee("Emp8",30,15000)
21 Employee e9=new Employee("Emp9",44,25000)
22 Employee e10=new Employee("Emp10",22,5000)
23 List empList=[e1,e2,e3,e4,e5,e6,e7,e8,e9,e10]
24
25 println "Employees who earn less than 5000:"
26 empList.findAll{it.salary<5000}.collect{println it}
27
28 println "\nYoungest and oldest employees"
29 println empList.min{it.age}
30 println empList.max{it.age}
31
32 println "\nEmployee with maximum salary"
33 println empList.max{it.salary}
34
35 println "\nList of names of all employees"
36 empList.each{print "${it.name} "}
37 print "\n"
```

```
lt-shivamg1@lt-shivamg1: ~/Desktop/groovy

lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy
Employees who earn less than 5000:
[Emp3,19,3000]
[Emp7,20,3000]

Youngest and oldest employees
[Emp3,19,3000]
[Emp6,68,45000]

Employee with maximum salary
[Emp4,60,50000]

List of names of all employees
Emp1 Emp2 Emp3 Emp4 Emp5 Emp6 Emp7 Emp8 Emp9 Emp10
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```

9. Consider the following piece of code:

```
String s="this string needs to be split"
println s.tokenize(" ")
println s.tokenize()
```

Compare this with the following code:

```
String s="this string needs to be split"
println s.split(" ")
println s.split(/\s/) (Try Same Parameter with tokenize)
```

Also try the following exercise:

```
String s="are.you.trying.to.split.me.mister?"
s.tokenize(".")
s.split(".")
```

Ans.

```
Open ▼ [+]
```

```
1 String s1="this string needs to be split"
2 println s1.tokenize(" ")
3 println s1.tokenize()
4 println s1.tokenize(/\s/)
5 print "\n"
6
7 String s2="this string needs to be split"
8 println s2.split(" ")
9 println s2.split(/\s/)
10 print "\n"
11
12 String s3="are.you.trying.to.split.me.mister?"
13 println s3.tokenize(".")
14 println s3.split(".")
```

```
lt-shivamg1@lt-shivamg1: ~/Desktop/groovy

lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy
[this, string, needs, to, be, split]
[this, string, needs, to, be, split]
[thi, , tring need, to be , plit]

[this, string, needs, to, be, split]
[this, string, needs, to, be, split]

[are, you, trying, to, split, me, mister?]
[]
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```

10. Get first, second and last element of Range

Ans.

```
Open ▼ [+]
```

```
1 Range range=5..20
2 println range.toList()
3 println "First element: ${range.from}"
4 println "Second element: ${range[1]}"
5 println "Last element: ${range.to}"
```

```
lt-shivamg1@lt-shivamg1: ~/Desktop/groovy

lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
First element: 5
Second element: 6
Last element: 20
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```

11. Print the table of a given number: 2 and 12

Ans.

```
Open ▼ [+]
```

```
1 (1..20).each {println "2x${it}=${2*it}\t12x${it}=${12*it}"}|
```

```
lt-shivamg1@lt-shivamg1: ~/Desktop/groovy

lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy
2x1=2    12x1=12
2x2=4    12x2=24
2x3=6    12x3=36
2x4=8    12x4=48
2x5=10   12x5=60
2x6=12   12x6=72
2x7=14   12x7=84
2x8=16   12x8=96
2x9=18   12x9=108
2x10=20  12x10=120
2x11=22  12x11=132
2x12=24  12x12=144
2x13=26  12x13=156
2x14=28  12x14=168
2x15=30  12x15=180
2x16=32  12x16=192
2x17=34  12x17=204
2x18=36  12x18=216
2x19=38  12x19=228
2x20=40  12x20=240
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```

12. We have a sorted list of alphabets a-z, print all alphabets appearing after j.

Ans.

```
Open ▼

1 Range range="a".. "z"
2 List list=range.toList()
3 println list.subList(10,26)
```

```
lt-shivamg1@lt-shivamg1: ~/Desktop/groovy

lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy
[k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z]
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```



13. Find the number of occurrences of a character in a string.

Ans.

```
Test.groovy
~/Desktop/groovy

1 String str="Hello Welcome to Groovy!!"
2 println str
3 List list=str.toList()
4 list.unique()
5 list.each{println "The character '${it}' appears ${str.count(it)} times in this string"}
```

```
lt-shivamg1@lt-shivamg1: ~/Desktop/groovy

lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy
Hello Welcome to Groovy!!
The character 'H' appears 1 times in this string
The character 'e' appears 3 times in this string
The character 'l' appears 3 times in this string
The character 'o' appears 5 times in this string
The character ' ' appears 3 times in this string
The character 'W' appears 1 times in this string
The character 'c' appears 1 times in this string
The character 'm' appears 1 times in this string
The character 't' appears 1 times in this string
The character 'G' appears 1 times in this string
The character 'r' appears 1 times in this string
The character 'v' appears 1 times in this string
The character 'y' appears 1 times in this string
The character '!' appears 2 times in this string
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```

14. Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.

Ans.

```
Open ▼

1 Range range=1..100
2 List list=range.toList()
3 list.each{
4     if(it%3==0 && it%5==0)
5         print "FizzBuzz, "
6     else if(it%5==0)
7         print "Buzz, "
8     else if(it%3==0)
9         print "Fizz, "
10    else
11        print "${it}, "
12 }
13 print "\n"
```

```
lt-shivamg1@lt-shivamg1: ~/Desktop/groovy
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy
1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14, FizzBuzz, 16, 17, Fizz, 19, Buzz, Fizz, 22, 23, Fizz, Buzz, 26, Fizz, 28, 29,
FizzBuzz, 31, 32, Fizz, 34, Buzz, Fizz, 37, 38, Fizz, Buzz, 41, Fizz, 43, 44, FizzBuzz, 46, 47, Fizz, 49, Buzz, Fizz, 52, 53, Fizz, Buzz, 56,
Fizz, 58, 59, FizzBuzz, 61, 62, Fizz, 64, Buzz, Fizz, 67, 68, Fizz, Buzz, 71, Fizz, 73, 74, FizzBuzz, 76, 77, Fizz, 79, Buzz, Fizz, 82, 83,
Fizz, Buzz, 86, Fizz, 88, 89, FizzBuzz, 91, 92, Fizz, 94, Buzz, Fizz, 97, 98, Fizz, Buzz,
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```

15. Consider a class named “Stack” that holds a list of objects and has the following operations associated:

- a) POP-pops the last element off the stack
- b) PUSH-pushes an element on top of the stack
- c) TOP-returns the element at the top of the list

Implement the aforesaid class

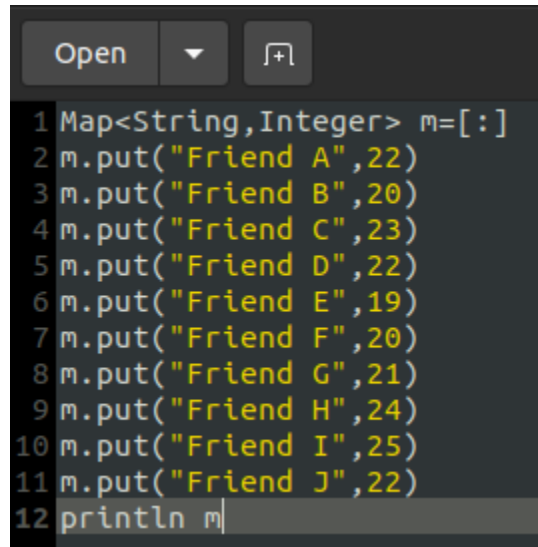
Ans.

```
Open [icon]
1 class Stack {
2     private List list=[]
3     public void POP() {
4         list.pop()
5     }
6     public void PUSH(Object obj) {
7         list.add(obj)
8     }
9     public Object TOP() {
10        return list.last()
11    }
12    public String toString() {
13        return list
14    }
15 }
16
17 Stack stk=new Stack()
18 stk.PUSH("hello")
19 stk.PUSH(10)
20 stk.PUSH(1.4)
21 stk.PUSH("GROOVY")
22 stk.PUSH(7.94)
23 stk.PUSH(100)
24 println stk
25 println stk.TOP()
26 stk.POP()
27 stk.POP()
28 println stk
29 println stk.TOP()
```

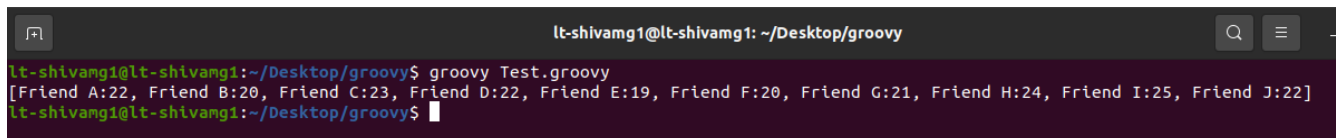
```
lt-shivamg1@lt-shivamg1: ~/Desktop/groovy
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy
[hello, 10, 1.4, GROOVY, 7.94, 100]
100
[hello, 10, 1.4, GROOVY]
GROOVY
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```

16. Create a new map consisting of 10 of your friend's names as keys and their ages as values

Ans.



```
1 Map<String,Integer> m=[:]
2 m.put("Friend A",22)
3 m.put("Friend B",20)
4 m.put("Friend C",23)
5 m.put("Friend D",22)
6 m.put("Friend E",19)
7 m.put("Friend F",20)
8 m.put("Friend G",21)
9 m.put("Friend H",24)
10 m.put("Friend I",25)
11 m.put("Friend J",22)
12 println m
```



```
lt-shivamg1@lt-shivamg1: ~/Desktop/groovy
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy
[Friend A:22, Friend B:20, Friend C:23, Friend D:22, Friend E:19, Friend F:20, Friend G:21, Friend H:24, Friend I:25, Friend J:22]
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```

17. Iterate over the previous map in as many ways as possible.

Ans.



```
1 Map<String,Integer> m=[:]
2 m.put("Friend A",22)
3 m.put("Friend B",20)
4 m.put("Friend C",23)
5 m.put("Friend D",22)
6 m.put("Friend E",19)
7 m.put("Friend F",20)
8 m.put("Friend G",21)
9 m.put("Friend H",24)
10 m.put("Friend I",25)
11 m.put("Friend J",22)
12 println "Using each:"
13 m.each{println it}
14 println "\nUsing eachWithIndex:"
15 m.eachWithIndex{key,value-> println "${key}\t${value}"}
```

```
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy
Using each:
Friend A=22
Friend B=20
Friend C=23
Friend D=22
Friend E=19
Friend F=20
Friend G=21
Friend H=24
Friend I=25
Friend J=22

Using eachWithIndex:
Friend A=22      0
Friend B=20      1
Friend C=23      2
Friend D=22      3
Friend E=19      4
Friend F=20      5
Friend G=21      6
Friend H=24      7
Friend I=25      8
Friend J=22      9
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```

18. Create a new map by adding two existing maps

Ans.

```
Open  ▾  [?]
1 Map m=[:]
2 m.put("Friend A",22)
3 m.put("Friend B",20)
4 m.put("Friend C",23)
5 m.put("Friend D",22)
6 m.put("Friend E",19)
7 m.put("Friend F",20)
8 m.put("Friend G",21)
9 m.put("Friend H",24)
10 m.put("Friend I",25)
11 m.put("Friend J",22)
12 Map m2=[:]
13 m2[1]="A"
14 m2[2]="B"
15 m2[3]="C"
16 m2[4]="D"
17 m2[5]="E"
18 Map m3=m+m2
19 println "Map m:"
20 println m
21 println "\nMap m2:"
22 println m2
23 println "\nMap m3:"
24 println m3
```

```
lt-shivamg1@lt-shivamg1: ~/Desktop/groovy
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy
Map m:
[Friend A:22, Friend B:20, Friend C:23, Friend D:22, Friend E:19, Friend F:20, Friend G:21, Friend H:24, Friend I:25, Friend J:22]

Map m2:
[1:A, 2:B, 3:C, 4:D, 5:E]

Map m3:
[Friend A:22, Friend B:20, Friend C:23, Friend D:22, Friend E:19, Friend F:20, Friend G:21, Friend H:24, Friend I:25, Friend J:22, 1:A, 2:B, 3:C, 4:D, 5:E]
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```

19. Try the following code on a map:

```
println map.class
println map.getClass()
```

What do you observe?

Ans. We observe that `map.class` gives null while `map.getClass()` prints the default class of the map object.

```
Open ▼
1 Map m=[:]
2 println m.class
3 println m.getClass()
```

```
lt-shivamg1@lt-shivamg1: ~/Desktop/groovy
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy
null
class java.util.LinkedHashMap
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```

20. Consider the following map:

```
Map m=['1':2, '2':3, '3':4, '2':5]
```

Is this a valid construction? What is the value of `m['2']`?

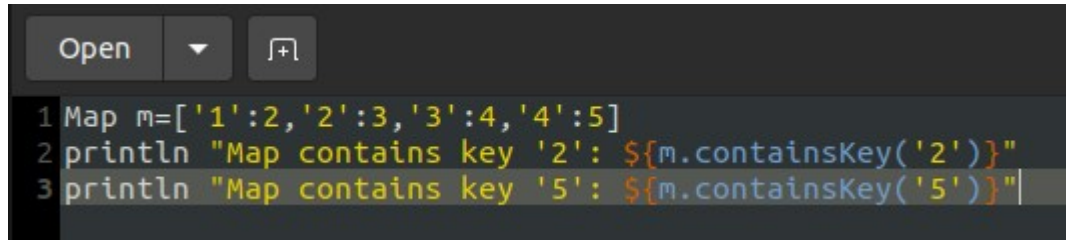
Ans.

```
Open ▼
1 Map m=['1':2, '2':3, '3':4, '2':5]
2 println m
3 println m['2']
```

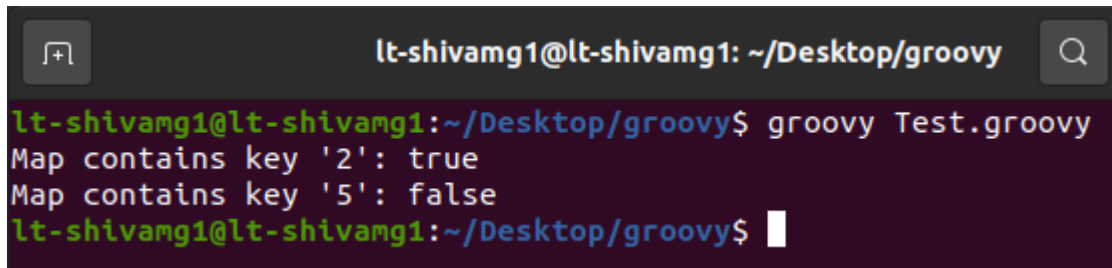
```
lt-shivamg1@lt-shivamg1: ~/Desktop/groovy
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy
[1:2, 2:5, 3:4]
5
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```

21. Find if a map contains a particular key.

Ans.



```
1 Map m=['1':2,'2':3,'3':4,'4':5]
2 println "Map contains key '2': ${m.containsKey('2')}"
3 println "Map contains key '5': ${m.containsKey('5')}"
```



```
lt-shivamg1@lt-shivamg1: ~/Desktop/groovy
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy
Map contains key '2': true
Map contains key '5': false
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```

22. Consider the following map:

Map m=[‘Computing’:[‘Computing’:600, ‘Information Systems’:300], ‘Engineering’:  
[‘Civil’:200, ‘Mechanical’:100], ‘Management’:[‘Management’:800]]

- a) How many university departments are there?
- b) How many programs are delivered by the Computing department?
- c) How many students are enrolled in the Civil Engineering program?

Ans: a) There are 3 university departments: Computing, Engineering, Management  
b) The Computing department delivers 2 programs: Computing and Information Systems  
c) There are 200 students enrolled in Civil Engineering program

23. Consider a class named “Employee” which has the following properties:

\*Name  
\*Age  
\*DepartmentName  
\*EmployeeNumber  
\*Salary

Let’s say that there’s a list of 50 employees available. Perform the following operations on the list of employees:

- a) Group the employees on the basis of the bracket in which their salary falls. The ranges are 0-5000, 5001-10000, and so on
- b) Get a count on the number of employees in each department
- c) Get the list of employees whose age is between 18 and 35
- d) Group the employees according to the alphabet with which their first name starts and display the number of employees in each group whose age is greater than 20
- e) Group the employees according to their department

Ans.



```
Test.groovy
~/Desktop/groovy

1 class Employee {
2     String name; int age; String deptName; int empNo; int sal
3     Employee(String name,int age,String deptName,int empNo,int sal) {
4         this.name=name
5         this.age=age
6         this.deptName=deptName
7         this.empNo=empNo
8         this.sal=sal
9     }
10    public static int getRandom(int min,int max) {
11        return (int)((Math.random()*(max-min))+min)
12    }
13    public String toString() {
14        return "${this.name}:${this.age}:${this.deptName}:${this.empNo}:${this.sal} "
15    }
16 }
17 int varSal=1000
18 List dept=["Dept_1","Dept_2","Dept_3","Dept_4","Dept_5"]
19 int deptSize=dept.size()
20 List letter=("A".."Z").toList()
21 int letterSize=letter.size()
22 Random idx=new Random()
23 List<Employee> em=[]
24 for(i in 1..50) {
25     em.add(new Employee("${letter.get(idx.nextInt(letterSize))}Emp$
26     {i}",Employee.getRandom(18,60),dept.get(idx.nextInt(deptSize)),i+1000,varSal))
27     varSal+=500
28 }
29 Map salGroups=[:].withDefault{[]}
30 em.each {employee->
31     int salary=employee.sal
32     String grp=""
33     if(salary<=5000)
34         grp="0-5000"
35     else if(salary<=10000)
36         grp="5001-10000"
37     else
38         grp="Above 10000"
39     salGroups[grp]<<employee
40 }
41 salGroups.each {grp,emp->
42     println "\nSalary Bracket ${grp}:"
43     emp.each{employee-> print "${employee.name} "}
44     println()
45 }
46
47 Map deptCount=[:].withDefault{0}
48 em.each {employee->
49     String deptmt=employee.deptName
50     deptCount[deptmt]++
51 }
52 deptCount=deptCount.sort{it.key[5]}
53 println()
54 deptCount.each {depart,count-> println "${depart}:${count} employees"}
55
56 List empAge=[]
57 em.findAll {18<=it.age && it.age<=35}.collect{empAge.add(it.name)}
58 println "\nEmployees between 18 and 35 years of age :\n${empAge}"
59
60 Map nameGroup=em.groupBy{it.name[0]}
61 println()
62 int emCount
63 nameGroup.each{
64     it.value.findAll{it.age>20}.collect{emCount++}
65     println "Employees with first alphabet ${it.key} and age>20 :${emCount}"
66 }
67
68 Map deptGroup=em.groupBy{it.deptName}
69 println()
70 deptGroup.each{println it}
```

```
lt-shivangi1@lt-shivangi1: ~/Desktop/groovy
lt-shivangi1@lt-shivangi1:~/Desktop/groovy$ groovy Test.groovy

Salary Bracket 0-5000:
ZEmp1 BEmp2 YEmp3 KEmp4 JEmp5 MEMP6 QEmp7 UEmp8 OEmp9

Salary Bracket 5001-10000:
HEmp10 PEMP11 BEMP12 JEMP13 BEMP14 PEMP15 VEMP16 FEMP17 FEMP18 JEMP19

Salary Bracket Above 10000:
REmp20 SEMP21 XEMP22 OEMP23 REMP24 XEMP25 GEMP26 XEMP27 YEMP28 OEMP29 VEMP30 PEMP31 EEMP32 TEMP33 OEMP34 PEMP35 KEMP36 BEMP37 CEMP38 XEMP39 QE
mp40 QEMP41 VEMP42 OEMP43 WEMP44 QEMP45 FEMP46 IEMP47 TEMP48 OEMP49 WEMP50

Dept_1:8 employees
Dept_2:6 employees
Dept_3:10 employees
Dept_4:8 employees
Dept_5:18 employees

Employees between 18 and 35 years of age :
[YEmp3, KEMP4, JEMP5, MEMP6, QEMP7, UEMP8, OEMP9, PEMP11, JEMP13, BEMP14, PEMP15, FEMP17, JEMP19, SEMP21, OEMP23, REMP24, YEMP28, PEMP31, OEMP
34, CEMP38, XEMP39, OEMP43, TEMP48, OEMP49, WEMP50]

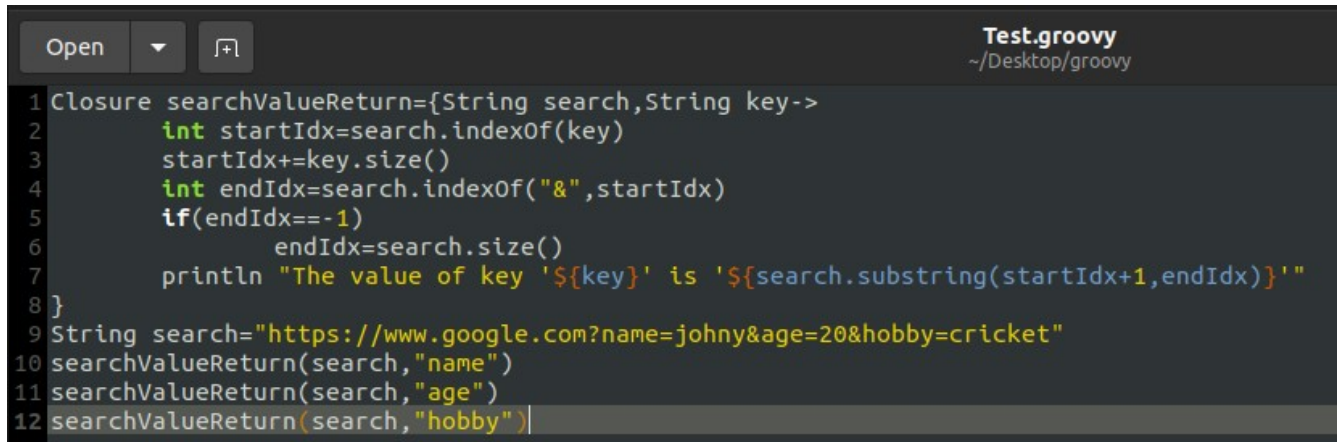
Employees with first alphabet Z and age>20 :1
Employees with first alphabet B and age>20 :5
Employees with first alphabet Y and age>20 :7
Employees with first alphabet K and age>20 :9
Employees with first alphabet J and age>20 :12
Employees with first alphabet M and age>20 :13
Employees with first alphabet Q and age>20 :17
Employees with first alphabet U and age>20 :18
Employees with first alphabet O and age>20 :24
Employees with first alphabet H and age>20 :25
Employees with first alphabet P and age>20 :28
Employees with first alphabet V and age>20 :31
Employees with first alphabet F and age>20 :34
Employees with first alphabet R and age>20 :36
Employees with first alphabet S and age>20 :37
Employees with first alphabet X and age>20 :41
Employees with first alphabet G and age>20 :42
Employees with first alphabet E and age>20 :43
Employees with first alphabet T and age>20 :45
Employees with first alphabet C and age>20 :46
Employees with first alphabet W and age>20 :48
Employees with first alphabet I and age>20 :49

Dept_3=[ZEmp1:52:Dept_3:1001:1000 , JEmp13:24:Dept_3:1013:7000 , PEMP15:19:Dept_3:1015:8000 , YEMP28:33:Dept_3:1028:14500 , VEMP30:59:Dept_3:1
030:15500 , PEMP35:40:Dept_3:1035:18000 , KEMP36:43:Dept_3:1036:18500 , OEMP43:34:Dept_3:1043:22000 , WEMP44:50:Dept_3:1044:22500 , QEMP45:54:
Dept_3:1045:23000 ]
Dept_5=[BEMP2:50:Dept_5:1002:1500 , JEMP5:22:Dept_5:1005:3000 , QEMP7:26:Dept_5:1007:4000 , HEMP10:38:Dept_5:1010:5500 , PEMP11:23:Dept_5:1011
:6000 , BEMP12:50:Dept_5:1012:6500 , VEMP16:43:Dept_5:1016:8500 , JEMP19:23:Dept_5:1019:10000 , REMP20:44:Dept_5:1020:10500 , SEMP21:35:Dept_5
:1021:11000 , OEMP23:35:Dept_5:1023:12000 , REMP24:25:Dept_5:1024:12500 , GEMP26:41:Dept_5:1026:13500 , XEMP27:52:Dept_5:1027:14000 , EEMP32:4
8:Dept_5:1032:16500 , QEMP41:43:Dept_5:1041:21000 , FEMP46:41:Dept_5:1046:23500 , WEMP50:22:Dept_5:1050:25500 ]
Dept_1=[YEMP3:22:Dept_1:1003:2000 , OEMP9:25:Dept_1:1009:5000 , OEMP29:40:Dept_1:1029:15000 , PEMP31:23:Dept_1:1031:16000 , OEMP34:35:Dept_1:1
034:17500 , QEMP40:59:Dept_1:1040:20500 , TEMP48:34:Dept_1:1048:24500 , OEMP49:30:Dept_1:1049:25000 ]
Dept_4=[KEMP4:35:Dept_4:1004:2500 , MEMP6:30:Dept_4:1006:3500 , UEMP8:34:Dept_4:1008:4500 , BEMP14:25:Dept_4:1014:7500 , FEMP17:31:Dept_4:1017
:9000 , XEMP22:57:Dept_4:1022:11500 , XEMP25:45:Dept_4:1025:13000 , VEMP42:41:Dept_4:1042:21500 ]
Dept_2=[FEMP18:59:Dept_2:1018:9500 , TEMP33:44:Dept_2:1033:17000 , BEMP37:46:Dept_2:1037:19000 , CEMP38:34:Dept_2:1038:19500 , XEMP39:28:Dept_
2:1039:20000 , IEMP47:55:Dept_2:1047:24000 ]
lt-shivangi1@lt-shivangi1:~/Desktop/groovy$
```



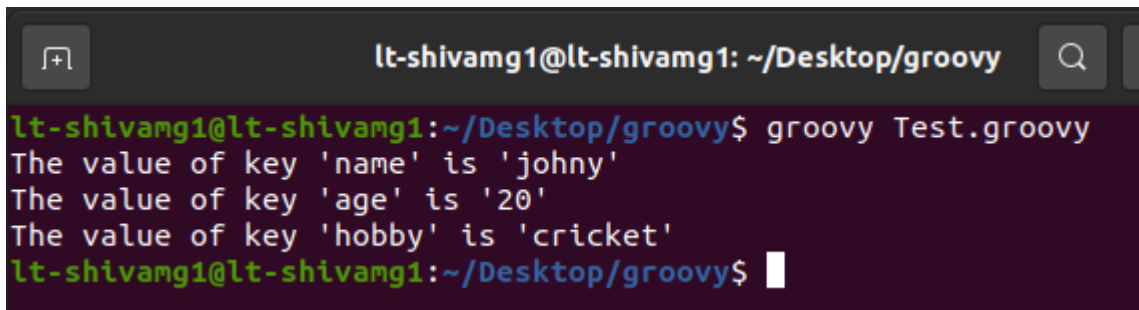
24. Write a method which returns the value of passed key from a search string of the form  
“https://www.google.com?name=johny&age=20&hobby=cricket”

Ans.



```
Test.groovy
~/Desktop/groovy

1 Closure searchValueReturn={String search,String key->
2     int startIdx=search.indexOf(key)
3     startIdx+=key.size()
4     int endIdx=search.indexOf("&",startIdx)
5     if(endIdx==-1)
6         endIdx=search.size()
7     println "The value of key '${key}' is '${search.substring(startIdx+1,endIdx)}'"
8 }
9 String search="https://www.google.com?name=johny&age=20&hobby=cricket"
10 searchValueReturn(search,"name")
11 searchValueReturn(search,"age")
12 searchValueReturn(search,"hobby")
```



```
lt-shivamg1@lt-shivamg1: ~/Desktop/groovy

lt-shivamg1@lt-shivamg1:~/Desktop/groovy$ groovy Test.groovy
The value of key 'name' is 'johny'
The value of key 'age' is '20'
The value of key 'hobby' is 'cricket'
lt-shivamg1@lt-shivamg1:~/Desktop/groovy$
```