Rishabh Rajpurohit

Groovy Collections Assignment

A0.

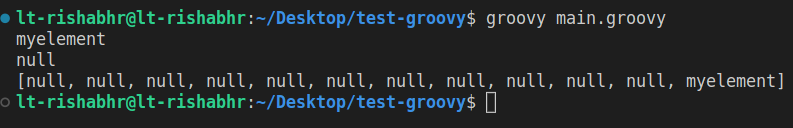
List l = []

l[11] = "rishabh"

println l[11]

println l.get(5)

println l



A1.

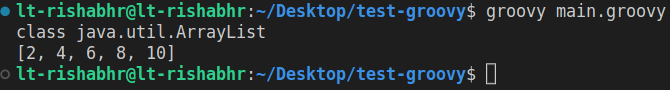
Range r = 1..10

def list = r.toList()

println list.getClass()

def evenList = list.findAll{item->~item%2}

println evenList



A2.

List l = [1,2,3,7,7,7,4,5,6,7,1,2,5,4,3,8,9]

Set s = l.toSet()

println "using toSet(): ${s}"

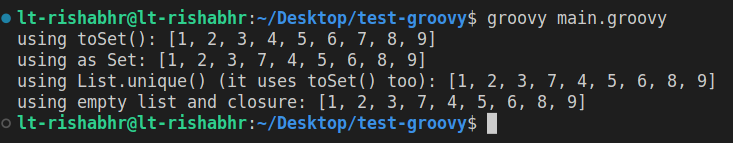
println "using as Set: ${l as Set}"

println "using List.unique() (it uses toSet() too): ${l.unique()}"

def seen = []

def uniqueList = l.findAll { !seen.contains(it) && seen.add(it) }

println "using empty list and closure: ${uniqueList}"



A3.

List l1 = [11,12,13,14]

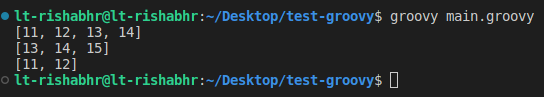
List l2 = [13,14,15]

List result = (l1+l1).unique() - l2

println l1

println l2

println result



A4.

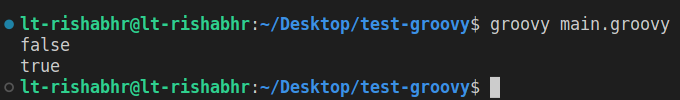
List l1 = [1,2,3,4,5]

List l2 = [7,8,9,4,10]

List l3 = [11,22,33,44,55]

println l1.disjoint(l2)

println l1.disjoint(l3)



A5.

def list = [1, 2, 3, 4, 5, 6, 7, 8, 9]

def result = []

list.eachWithIndex { item, index ->

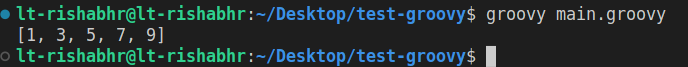
if (index % 2 == 0) {

result.add(item)

}

}

println result

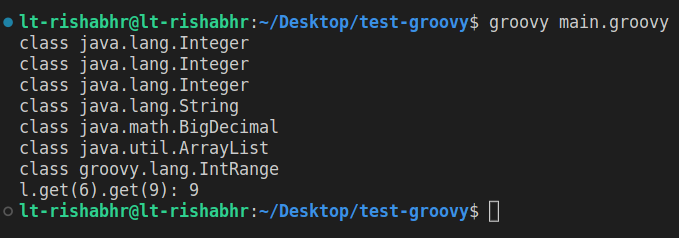


A6.

List l = [1,2,3,"element1",0.3,[2,4,6],0..10]

l.each{element->println element.getClass()}

println "l.get(6).get(9): ${l.get(6).get(9)}"



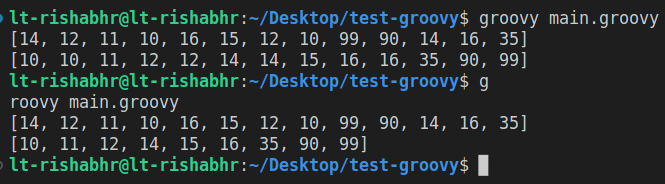
A7.

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

println l

l.sort().unique()

println l



A8.

class Emp {

String name

Integer age

Integer salary

public Emp(String name, Integer age, Integer salary){

this.name = name

this.age = age

this.salary = salary

}

public String toString(){

return "${this.name}+${this.age}Y+${this.salary}\$"

}

}

Emp e1 = new Emp("e1",28,20000)

Emp e2 = new Emp("e2",52,50000)

Emp e3 = new Emp("e3",32,800)

Emp e4 = new Emp("e4",25,200000)

Emp e5 = new Emp("e5",18,10000)

Emp e6 = new Emp("e6",12,20200)

Emp e7 = new Emp("e7",20,403000)

Emp e8 = new Emp("e8",26,2000)

Emp e9 = new Emp("e9",24,49000)

Emp e10 = new Emp("e10",22,93000)

List l = [e1,e2,e3,e4,e5,e6,e7,e8,e9,e10]

def queryA = l.findAll{it->it.salary<5000}

println "queryA: ${queryA}"

def queryB = l.min{it.age}

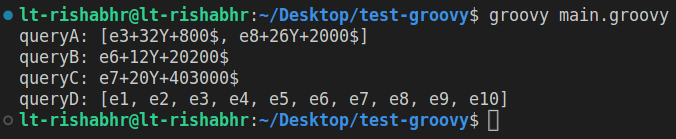
println "queryB: ${queryB}"

def queryC = l.max{it.salary}

println "queryC: ${queryC}"

def queryD = l.collect{it.name}

println "queryD: ${queryD}"



A9.

String s = "this string needs to be split"

println s.tokenize(" ")

println s.tokenize()

println s.tokenize(/\s/)

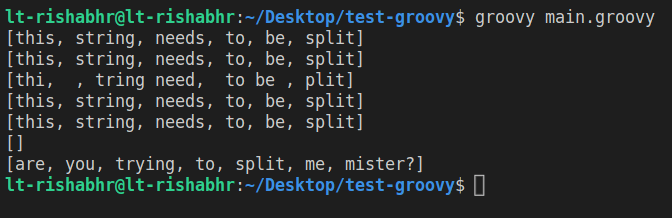
println s.split(" ")

println s.split(/\s/)

s = "are.you.trying.to.split.me.mister?"

println s.split(".")

println s.tokenize(".")



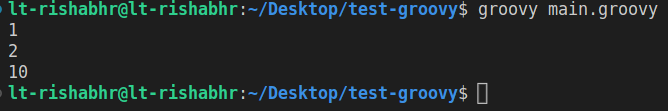
A10.

Range r = 1..10

println r.get(0)

println r.get(1)

println r.get(r.size()-1)

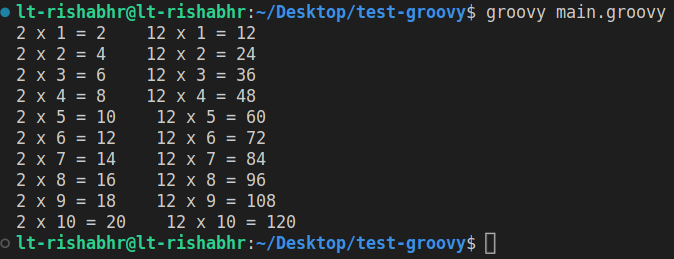


A11.

(1..10).each {

println "2 x $it = ${2 \* it} 12 x $it = ${12 \* it}"

}

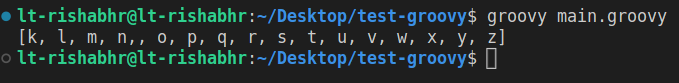


A12.

List l = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n,','o','p','q','r','s','t','u','v','w','x','y','z']

def listAfterJ = l.findAll{it->it>'j'}

println listAfterJ



A13.

String someString = "efbwefbwfiwfwfhwfiwninviwvowifowienfw"

def whatToFind = 'w'

def count = someString.findAll{it->it==whatToFind}.size()

println "count of ${whatToFind}: ${count}"



A14.

(1..100).each{it->

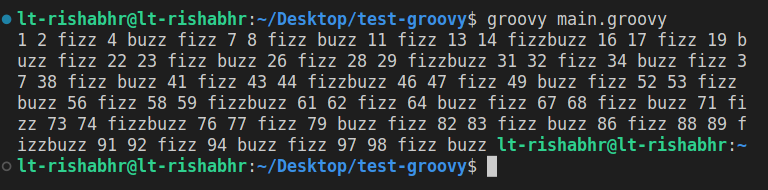
if((it%3==0)&&(it%5==0)) print "fizzbuzz "

else if(it%3==0) print "fizz "

else if(it%5==0) print "buzz "

else print "${it} "

}



A15.

class Stack {

List<Object> stackList = []

def push(Object item) {

stackList.add(item)

}

def pop() {

stackList.removeAt(stackList.size() - 1)

}

def top() {

stackList[stackList.size() - 1]

}

}

def myStack = new Stack()

myStack.push("apple")

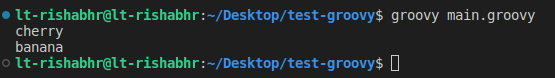
myStack.push("banana")

myStack.push("cherry")

println myStack.top() // Output: cherry

myStack.pop()

println myStack.top() // Output: banana

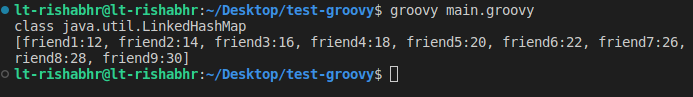


A16.

def friends = ['friend1':12,'friend2':14,'friend3':16,'friend4':18,'friend5':20,'friend6':22,'friend7':24,'friend7':26,'friend8':28,'friend9':30]

println friends.getClass()

println friends



A17.

def friends = ['friend1':12,'friend2':14,'friend3':16,'friend4':18,'friend5':20,'friend6':22,'friend7':24,'friend7':26,'friend8':28,'friend9':30]

println friends.getClass()

println friends

friends.each { key, value ->

println "$key is $value years old"

}

def doubledAges = friends.collect { key, value ->

value \* 2

}

println doubledAges

for (key in friends.keySet()) {

println "Key: $key"

}

for (value in friends.values()) {

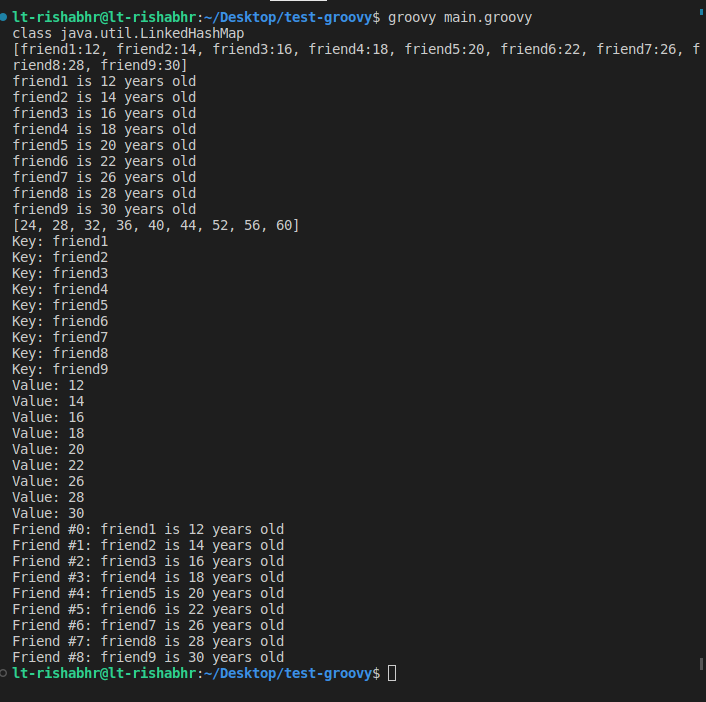
println "Value: $value"

}

friends.eachWithIndex { key, value, index ->

println "Friend #$index: $key is $value years old"

}



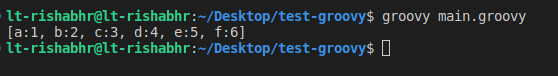
A18.

def map1 = [a: 1, b: 2, c: 3]

def map2 = [d: 4, e: 5, f: 6]

def combinedMap = map1 + map2

println combinedMap



A19.

Map.class shows null

whereas map.getClass() returns Java.util.LinkedHashMap

A20.

This is valid but since map only stores unique keys the value of existing key gets updated to last entry.

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

println m



A21.

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

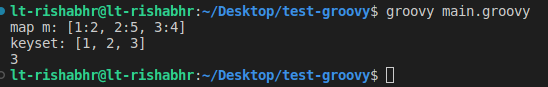
println "map m: ${m}"

def mkeys = m.keySet()

println "keyset: ${mkeys}"

def result = mkeys.find{it->it=="3"}

println result



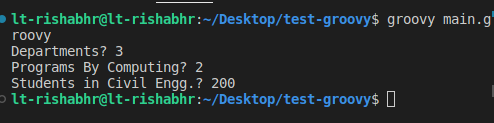
A22.

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

println "Departments? ${m.keySet().size()}"

println "Programs By Computing? ${m['Computing'].keySet().size()}"

println "Students in Civil Engg.? ${m['Engineering']['Civil']}"



A23.

class Employee {

String name

int age

String deptName

int empID

int salary

static Employee createRandomEmployee() {

String[] names = ["John", "Mary", "David", "Lisa", "Peter", "Sarah"]

String[] deptNames = ["Sales", "Marketing", "Engineering", "Finance"]

Random rand = new Random()

Employee emp = new Employee()

emp.name = names[rand.nextInt(names.length)]

emp.age = rand.nextInt(30) + 10 // Random age between 20 and 59

emp.deptName = deptNames[rand.nextInt(deptNames.length)]

emp.empID = rand.nextInt(10000)

emp.salary = rand.nextInt(12000) // Random salary between 50000 and 99999

return emp

}

public String toString(){

return "${this.name}->${this.age}Y"

}

}

List<Employee> employees = (1..10).collect { Employee.createRandomEmployee() }

def SalaryRanges = [

"0-5000":{it.salary<=5000},

"5001-10000":{it.salary>5000 && it.salary <=10000},

"10001 and above":{it.salary>10000}

]

def groupedEmployeesBySalary = employees.groupBy{emp->

SalaryRanges.findResult {

range->range.value(emp)?range.key:null

}}

print "A."

groupedEmployeesBySalary.each { range, emps ->

println "\nEmployees in range $range:"

emps.each { emp ->

println " $emp.name (ID: ${emp.empID}, Age: ${emp.age}, Salary: ${emp.salary})"

}

}

print "\nB."

def groupedEmployeesByDept = employees.groupBy{it.deptName}

groupedEmployeesByDept.each { key,value ->

println "Employees in ${key}: ${value.size()}"

}

print "\nC.\nEmployees b/w age 18 and 35: "

def EmployeesBw18and35 = employees.findAll{

it.age>18 && it.age<35

}

println "Employees of Age b/w 18 and 35: "

EmployeesBw18and35.each{emp->println emp}

println "\nD."

def groupedEmployeesByFirstNameLetter20Above = employees.groupBy { emp -> emp.name[0] }.collectEntries { k, v -> [k, v.count { it.age > 20 }] }

groupedEmployeesByFirstNameLetter20Above.each { letter, count ->

println "Employees with first name starting with '$letter': $count"

}

println "\nE."

def groupedEmployeesbyDepartment = employees.groupBy { emp -> emp.deptName }

groupedEmployeesbyDepartment.each { dept, empList ->

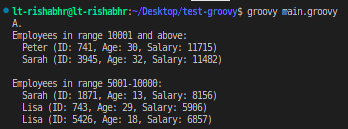
println "Employees in department '$dept':"

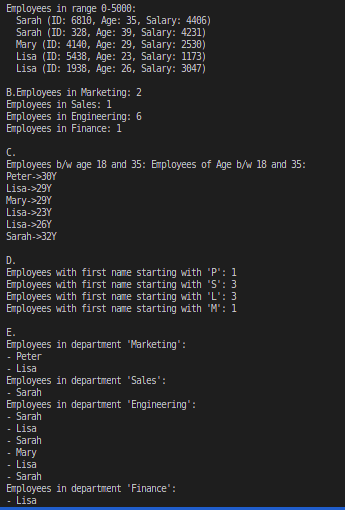
empList.each { emp ->

println "- ${emp.name}"

}

}





A24.

def searchString = "https://www.google.com?name=johny&age=20&hobby=cricket"

Closure findKey = {String keyName->

def key = keyName

def startIndex = searchString.indexOf("${key}=")

def value = ""

if (startIndex != -1) {

def endIndex = searchString.indexOf("&", startIndex)

if (endIndex == -1) {

endIndex = searchString.length()

}

value = searchString.substring(startIndex + key.length() + 1, endIndex)

}

println "Value of ${key}: ${value}"

}

findKey("name")

findKey("age")

findKey("hobby")

