**[ 2CEIT5PE5:MOBILE APPLICATION DEVELOPMENT]**

Practical: 1



**AIM: Basics of Kotlin**

Submitted By: Chaudhari yogita

Enrollment number: 21012012004



**Department of Computer Engineering/Information Technology**

# Store & Display values in different variable of different type (Integer, Double, Float, Long, Short, Byte, Char, Boolean, String)

**Answer:**

fun main()

{

var int:Int=10

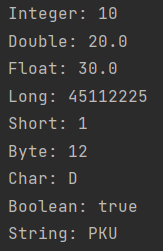
var doub:Double=20.0 var ft:Float=30.0F var lg:Long=45112225 var srt:Short=1

var byte:Byte=12 var char:Char='D'

var bool:Boolean=true var string:String="PKU"

println("Integer: $int") println("Double: $doub") println("Float: $ft") println("Long: $lg") println("Short: $srt") println("Byte: $byte") println("Char: $char") println("Boolean: $bool") println("String: $string")

}

**Output:**

* 1. **Type conversion: Integer to Double, String to Integer, String to Double. Answer:**

fun main()

{

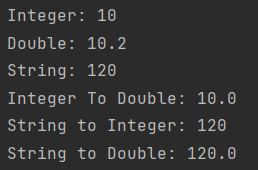
var int:Int=10

var double:Double=10.2 var str:String="120" *println*("Integer: $int") *println*("Double: $double") *println*("String: $str") double = int.toDouble()

*println*("Integer To Double: $double") int = str.*toInt*()

*println*("String to Integer: $int") double = str.*toDouble*() *println*("String to Double: $double")

}

**Output:**

* 1. **Scan student’s information and display all the data. Answer:**

fun main()

{

*print*("Student Enrollment: ")

var enr:Long= *readLine*()!!.*toLong*() *print*("Student Name: ")

var name:String= *readLine*()!!.toString() *print*("Student Branch: ")

var branch:String= *readLine*()!!.toString() *print*("Student Class: ")

var class\_name:String= *readLine*()!!.toString() *print*("Student Batch: ")

var batch:String= *readLine*()!!.toString() *print*("Student College Name: ")

var college\_name:String= *readLine*()!!.toString() *print*("Student uni Name: ")

var uni\_name:String= *readLine*()!!.toString() *print*("Student Age: ")

var age:Int= *readLine*()!!.*toInt*() *println*("=================================")

*println*("Enrollment: $enr") *println*("Name: $name") *println*("Branch: $branch") *println*("Class: $class\_name") *println*("Batch: $batch") *println*("College Name: $college\_name") *println*("Uni Name: $uni\_name") *println*("Age: $age")

}

# Output:

* 1. **Find the number is odd or even by using Control Flow inside println() method**. **Answer:**

fun main()

{

*print*("Enter Number: ")

var num:Int= *readLine*()!!.*toInt*() *println*(if(num%2==0) "Num is Even" else "Num is Odd")

}

**Output:**

* 1. **Display month name using When Answer:**

fun main()

{

*print*("Enter Month Number: ")

var mon:Int= *readLine*()!!.*toInt*() when(mon){

1->*println*("Jan") 2->*println*("Feb")

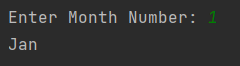
3->*println*("March") 4->*println*("Apr")

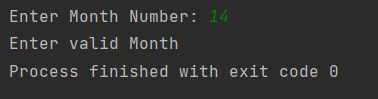
5->*println*("May") 6->*println*("Jun") 7->*println*("Jul") 8->*println*("Aug") 9->*println*("Sep") 10->*println*("Oct") 11->*println*("Nov") 12->*println*("Dec")

else->*print*("Enter valid Month")

}

}

**Output:**



* 1. **By using a user defined function perform all arithmetic operations. Answer:**

fun main()

{

*println*("Addition of: 111,2222,-222 is: ${*operations*('+',111,2222,-222)}") *println*("Subtraction of: 111,2222,-222 is: ${*operations*('-',111,2222,-222)}") *println*("Multiplication of: 111,2222,-222 is: ${*operations*('\*',111,2222,-222)}") *println*("Division of: 2222,111 is: ${*operations*('/',2222,111)}")

}

fun operations(op:Char,vararg numArr:Int):Int{ var result:Int=0

when(op)

{

'+'->{

for(num in numArr)

{

result+=num

}

}

'-'->{

result=numArr[0]

for (num in 1 *until* numArr.size)

{

result-=numArr[num]

}

}

'\*'->{

result=1

for(num in numArr)

{

result\*=num

}

}

'/'->{

result=numArr[0]

for (num in 1 *until* numArr.size)

{

result/=numArr[num]

}

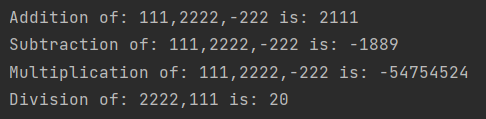
}

else->return -1

}

return result

}

**Output:**

* 1. **Find the factorial of number by recursion. Explain "**tailrec**" keyword. Answer:**

import java.math.BigInteger

fun main()

{

*print*("Enter Number: ")

var num:BigInteger = *readLine*()!!.*toBigInteger*() *println*("Factorial is: ${*fact2*(num)}")

*println*("With using tailrec Factorial is: ${*fact*(num)}")

}

tailrec fun fact(n: BigInteger, temp: BigInteger = BigInteger("1")): BigInteger { return if (n == BigInteger("1")){

temp

} else {

*fact*(n-BigInteger("1"), temp\*n)

}

}

fun fact2(n: BigInteger, temp: BigInteger = BigInteger("1")): BigInteger { return if (n == BigInteger("1")){

temp

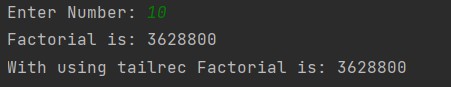
} else {

*fact*(n-BigInteger("1"), temp\*n)

}

}

**Output:**



* 1. **Create different types of Array as shown in image. Explore Arrays.deepToString(), contentDeepToString() methods, IntArray variable .joinToString() and use in program to print Array. Explore range, downTo, until etc. for loop and use in this program.** Sort **Array of Integer data type without using inbuilt function & with using inbuilt function.**

**Answer:**

fun main()

{

*println*("Create Array-1 by using arrayof () method:") var arr1 = *arrayOf*(1,2,3,4,5)

//joinToString

*println*(arr1.*joinToString*())

*println*("Create Array-2 by using Array<>():") var arr2 = *arrayOf*<String>("D","K")

//contentDeepToString

*println*(arr2.*contentDeepToString*())

*println*("Create Array-3 by using Array<>() and lambda function:") var arr3 = Array(8)**{**i:Int**->**i**}**

*println*("\*\*\*\*\*\*\*\*\*\*\*\*\*Before Sorting Without Built-in Function\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

*println*(arr3.*joinToString*()) for(i in 0 *until* arr3.size)

{

for(j in i+1 *until* arr3.size)

{

if(arr3[i]<arr3[j])

{

arr3[j]=arr3[j]+arr3[i] arr3[i]=arr3[j]-arr3[i]

arr3[j]=arr3[j]-arr3[i]

}

}

}

*println*("\*\*\*\*\*\*\*\*\*\*\*\*\*After Sorting Without Built-in Function\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

*println*(arr3.*joinToString*())

*println*("Create Array-4 by using IntArray ():") var arr4 = IntArray(3)

//until

for(i in 0 *until* arr4.size)

{

*print*("a[$i]: ")

arr4[i]= *readLine*()!!.*toInt*()

}

*println*("Create Array-5 by using intArrayof () :") var arr5 = *intArrayOf*(25,90,10,35)

*println*("\*\*\*\*\*\*\*\*\*\*\*\*\*Before Sorting With Built-in Function\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

*println*(arr5.*joinToString*())

//Sort arr5.*sort*()

*println*("\*\*\*\*\*\*\*\*\*\*\*\*\*After Sorting With Built-in Function\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

*println*(arr5.*joinToString*())

*println*("Create 20 Array-6 by using arrayof () and intArrayof() :") var arr6 = *arrayOf*(*intArrayOf*(1,2), *intArrayOf*(3,4,3))

for(i in 0 *until* arr6.size)

{

for(j in 0 *until* arr6[i].size)

{

*print*(arr6[i][j])

}

*println*()

}

//Range

val num = 10

*println*("\*\*\*\*\*\*\*\*\*\*\*\*\*Use In Range\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*") if (num in 5..10) {

*println*("in range")

}

*println*("\*\*\*\*\*\*\*\*\*\*\*\*\*Use Notin Range\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*") if(num !in 5 .. 9)

{

*println*("not in range")

}

//step

*println*("\*\*\*\*\*\*\*\*\*\*\*\*\*Use Step\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*") for (x in 1..10 *step* 2) {

*print*("$x ")

}

*println*()

//downTo

*println*("\*\*\*\*\*\*\*\*\*\*\*\*\*Use downTo\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*") for (x in 9 *downTo* 0 *step* 3) {

*print*("$x ")

}

*println*()

}

# Output:

* 1. **Find the max number from ArrayList. Answer:**

fun main() {

var arrlist = ArrayList<Int>() for(i in 0..5)

{

*print*("arr[$i]: ") arrlist.add(*readLine*()!!.*toInt*())

}

var maxNum:Int maxNum=arrlist[0] for (i in arrlist) {

if(maxNum<i)

{

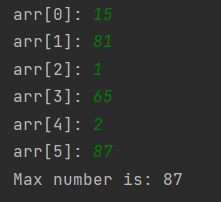
maxNum=i

}

}

*println*("Max number is: $maxNum")

}

**Output:**

* 1. **Write Different types of Class & Constructor. Create a class Car and set various members like type, model, price, owner, milesDrive. add the function getCarPrice in it. Create an object of Car class and access property of it. (getCarInformation(), getOriginalCarPrice(), getCurrentCarPrice(), displayCarInfo() etc.)**

**Constructor:**

A constructor is a concise way to initialize class properties.

In is a special member function that is called when an object is instantiated (created). In Kotlin, there are two type of constructor.

* + 1. Primary constructor
    2. Secondary constructor

**Class:**

Objects are created from classes. A class is a blueprint for an object; it shares common properties and behavior in form of members and member functions.

* simple class
* empty class
* primary constructor
* open class
* data class
* nested class
* inner class
* abstract class
* sealed class

**Answer:**

class Car(info:String,Owner:String,miles:Int,OGPrice:Double,currentPrice:Double){ lateinit var info:String

lateinit var Owner:String var miles:Int

var OGPrice:Double

var currentPrice:Double

init{

*println*("Object of class is created and Init is called") this.info = info

this.Owner = Owner this.miles = miles this.OGPrice = OGPrice

this.currentPrice = currentPrice

}

fun getCarInformation(): String { return info

}

fun getOriginalCarPrice(): Double { return OGPrice

}

fun getCurrentCarPrice(): Double {

return currentPrice

}

fun displayCarInfo(){

*println*(" ")

*println*("Car Information: ${getCarInformation()}")

*println*("Car Owner: $Owner")

*println*("Miles Drive: $miles")

*println*("Original Car Price: ${getOriginalCarPrice()}") *println*("Current Car Price: ${getCurrentCarPrice()}") *println*(" \n")

}

}

fun main() {

*println*("Creating Car Class Object car1 in next line") val car1 = Car("BMW, 2018","Aman",105, 10000.0,98950.0)

car1.displayCarInfo()

*println*("Creating Car Class Object car2 in next line") val car2 = Car("BMW, 2019","Karan",20, 400000.0,399800.0)

car2.displayCarInfo()

*println*("\n\*\*\*\*\*\*\*\*\*\* ArrayList of Car \*\*\*\*\*\*\*\*\*\*") val carlist = ArrayList<Car>()

carlist.add(Car("Toyota, 2017","KJS",100,1080000.0,1079000.0)) carlist.add(Car("Maruti, 2020","NPP",200,4000000.0,3998000.0))

carlist[0].displayCarInfo() carlist[1].displayCarInfo()

}

# Output:

* 1. **Write about Operator Overloading. Perform Matrix Addition, Subtraction & Multiplication using Class Matrix & operator overloading. Overload toString() function in Matrix class.**

**Operator Overloading**

* Operator overloading gives the ability to use the same operator to do various operations**.**

**Answer:**

fun main() {

val firstMatrix = Matrix(*arrayOf*(*intArrayOf*(3,-2,5), *intArrayOf*(3,0,4)),2,3) val secondMatrix = Matrix(*arrayOf*(*intArrayOf*(2,3),*intArrayOf*(-9,0), *intArrayOf*(0,4)),3,2)

val secondMatrix1 = Matrix(*arrayOf*(*intArrayOf*(6,3), *intArrayOf*(9,0), *intArrayOf*(5,4)),3,2)

*println*("\*\*\*\*\*\*\*\*\*\* Addition \*\*\*\*\*\*\*\*\*\*") *println*("Matrix:1 ") *print*(secondMatrix1.toString()) *println*("Matrix:2 ") *print*(secondMatrix.toString())

val thirdMatrix = secondMatrix1 + secondMatrix

*println*("Addition: \n$thirdMatrix")

*println*("\*\*\*\*\*\*\*\*\*\* Subtraction \*\*\*\*\*\*\*\*\*\*")

*println*("Matrix:1 ") *print*(secondMatrix1) *println*("Matrix:2 ") *print*(secondMatrix)

val subtractMatrix = secondMatrix1 - secondMatrix

*println*("Subtraction: \n$subtractMatrix")

*println*("\*\*\*\*\*\*\*\*\*\* Multiplication \*\*\*\*\*\*\*\*\*\*") println("Matrix:1 ") print(secondMatrix1.toString()) println("Matrix:2 ") print(secondMatrix.toString())

val multiplication = firstMatrix \* secondMatrix println("Multiplication: \n$multiplication")

}

class Matrix(matrix: Array<IntArray>,i:Int,j:Int){ var i:Int

var j:Int

lateinit var matrix : Array<IntArray> init{

this.i = i this.j = j

this.matrix = matrix

}

operator fun plus(p: Matrix) : Matrix { val result = Array(i){IntArray(j)} for(row in 0 until i){

for(column in 0 until j){

result[row][column] = matrix[row][column] + p.matrix[row][column]

}

}

return Matrix(result,i,j)

}

operator fun minus(p:Matrix):Matrix{ val result = Array(i){IntArray(j)} for(row in 0 until i){

for(column in 0 until j){

result[row][column] = matrix[row][column] - p.matrix[row][column]

}

}

return Matrix(result,i,j)

}

operator fun times(p:Matrix):Matrix{ val result = Array(i){IntArray(i)} for(row in 0 until i){

for(column in 0 until i){ for(k in 0 until j){

result[row][column] += matrix[row][k] \* p.matrix[k][column]

}

}

}

return Matrix(result,i,i)

}

override fun toString(): String { var msg : String = ""

for(row in matrix.indices){

for(column in 0 until matrix[row].size){ msg += "${matrix[row][column]} "

}

msg += "\n"

}

return msg

}

}

# Output:

