Artificial Intelligence
and Data Science Department

OOPM / Odd Sem 2021-22 / Experiment 1

YASH SARANG
DIV: D6AD
ROLL NO: 47

Program 1.A
Aim: To implement a program to print the roots of quadratic equations.

Theory:
Instance Variable: A variable declared inside the class but outside the body of the method is called an instance variable.

Local Variable: A local variable in Java is a variable that's declared within the body of a method

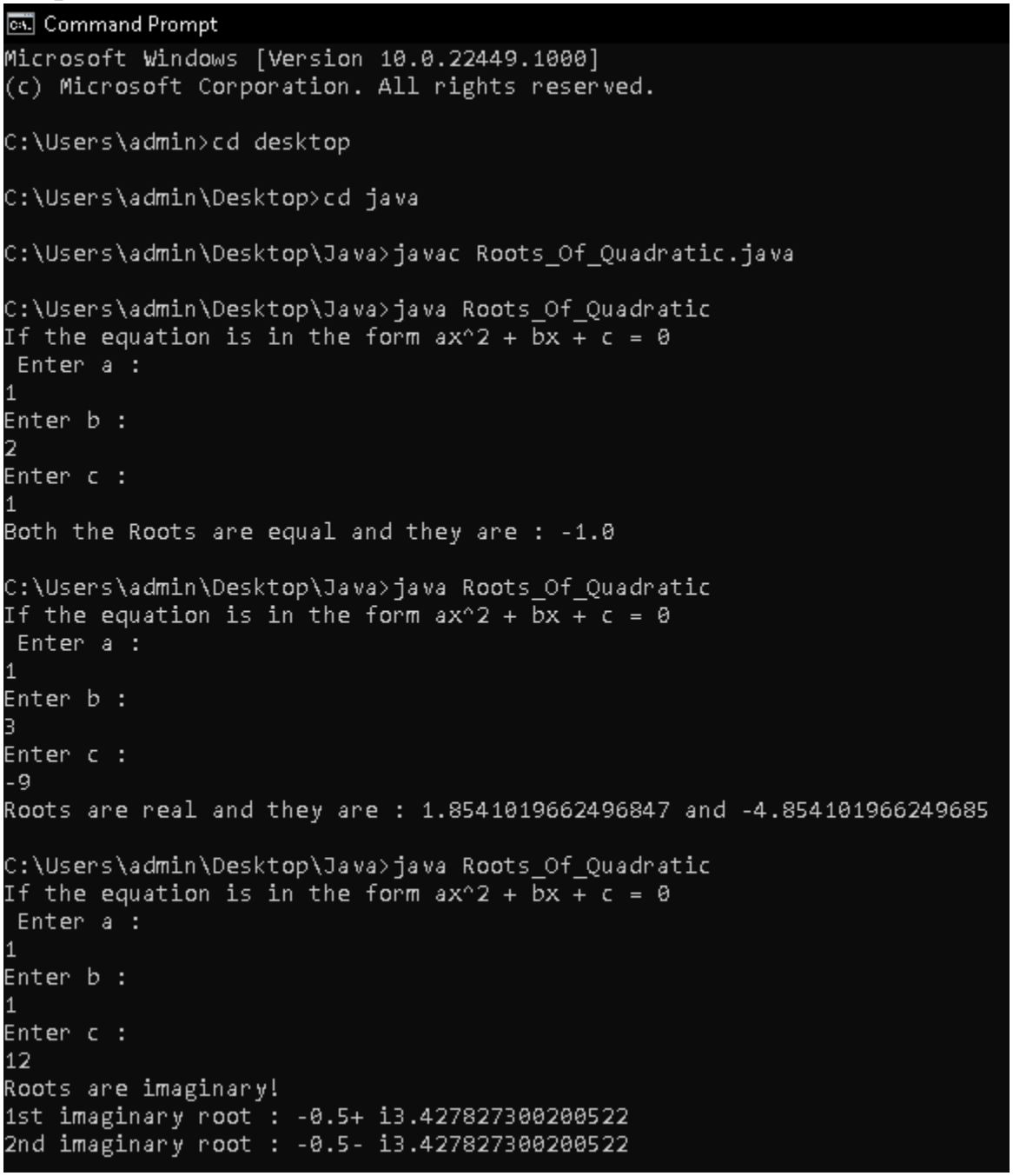
Static Variable: Static variables are stored in the static memory. Variables are created when the program starts and destroyed when the program stops.

Data Types: int: 4 byte, float: 4 byte, long: 8 byte, double:8 byte.

Looping constructs:
```
while(true){                // code to be executed
}
```
CODE:
```
import java.lang.*;
import java.lang.Math;
import java.util.*;
class prime_check
{
	public static void main(String args[])
	{
		Scanner input = new Scanner(System.in); //System.in is a standard input stream.
		System.out.print("Enter the integer number to check - ");
		int a= input.nextInt();
		int x=0;
		if (a<2)
		{
			System.out.println(a+" is not a Prime Number");
		}
		else if (a==2)
		{
			System.out.println(a + " is a Prime Number");
		}
		else
		{
			for(int i=2;i<=(int) a/2;i++)
			{
				if (a%i==0)
					x++;
			}
			if (x>0)
			{
```

```
                        System.out.println(a+" is not a Prime Number with "+x+" factors");
                }
                else
                {
                        System.out.println(a+" is a Prime Number");
                }
        }
    }
}
```

Output Screenshot of the Code:



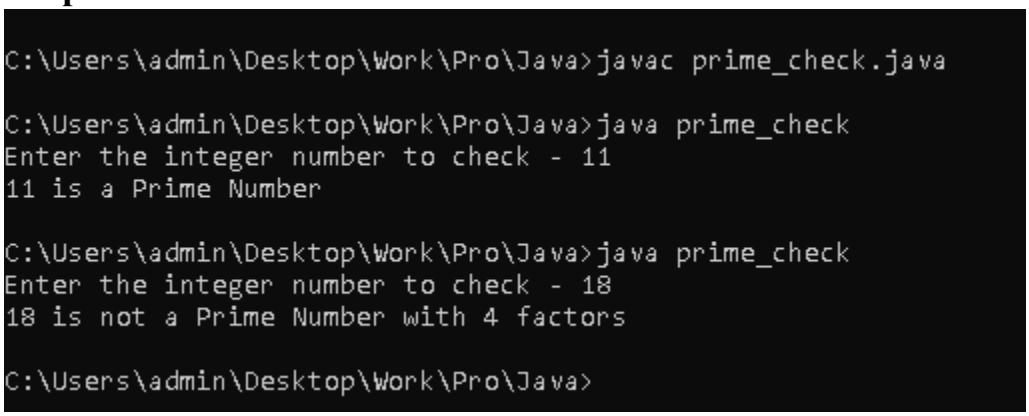
Program 1.b

Aim: To implement a program to check if the entered no. is a prime no. or not.

Code:

```
import java.lang.*;
import java.util.*;
import java.lang.Math;
class Roots_Of_Quadratic
{
	public static void main(String args[])
	{
		byte a,b,c;
		int D;
		Scanner input = new Scanner(System.in);
		System.out.println("If the equation is in the form ax^2 + bx + c = 0 \n Enter a : ");
		a = input.nextByte();
		System.out.println("Enter b : ");
		b = input.nextByte();
		System.out.println("Enter c : ");
		c = input.nextByte();
		double x1,x2;
		D = (b*b)-(4*a*c);
		if (D<0)
		{
			System.out.println("Roots are imaginary!");
			x1 = (double) ((-1.0*b)/(2*a));
			x2 = (Math.sqrt(-D)/(2.0*a));
			System.out.println("1st imaginary root : " + x1 + "+ i" + x2);
			System.out.println("2nd imaginary root : " + x1 + "- i" + x2);
		}
		else if(D==0)
		{
			x1=(-b)/(2*a);
			System.out.println("Both the Roots are equal and they are : " + x1);
		}
		else
		{
			x1=((-b)+Math.sqrt(D))/(2*a);
			x2=((-b)-Math.sqrt(D))/(2*a);
			System.out.println("Roots are real and they are : " + x1 + " and " + x2);
		}
	}
}
```

Output Screenshot of the Code:



Program 1.c

Aim: To implement a program to demonstrate the working of types of operators(Bitwise, Logical and relational) using switch case.

Theory:

Relational Operators:

- == (Equal to)– This operator is used to check if both operands are equal.
- != (Not equal to)– Can check if both operands are not equal.
- >(Greater than)– Can check if the first operand is greater than the second.
- < (Less than)- Can check if the first operand is lesser than the second.
- >= (Greater than equal to)– Check if the first operand is greater than or equal to the second.
- <= (Less than equal to)– Check if the first operand is lesser than or equal to the second

Logical Operators:

- && (AND) – It is used to check if both the operands are true.
- || (OR) – These operators are used to check if at least one of the operands is true.
- ! (NOT) – Used to check if the operand is false

Bitwise Operators:

- & (Bitwise AND) – Converts the value of both the operands into binary form and performs AND operation bit by bit.
- | (Bitwise OR) – Converts the value of both the operands into binary form and performs OR operation bit by bit.
- ^ (Bitwise exclusive OR) – Converts the value of both the operands into binary form and performs EXCLUSIVE OR operation bit by bit.

Code:

```
class Operators
{
	public static void main(String args[])
	{
		int option =1;
		int a, b;
		a = 10;
		b = 5;
		switch(option)
		{
			case 1:
				System.out.println("The value of a & b is "+ (a&b));
				System.out.println("The value of a | b is "+ (a|b));
				System.out.println("The value of a ^ b is "+ (a^b));
				System.out.println("The value of ~a is "+ ~a);
				System.out.println("The value of a<<2 is "+ (a<<2));
				System.out.println("The value of a>>2 is "+ (a>>2));
				break;
			case 2:
				System.out.println("The value of ((a>15) && (b<5)) is "+ ((a>15) &&
	(b<5)));
				System.out.println("The value of ((a>5) || (b<15)) is "+ ((a>5) ||
	(b<15)));
				System.out.println("The value of !((a>25) == (b<5)) is "+ !((a>25) ==
	(b<5)));
				break;
			case 3:
				System.out.println("The value of a == b is " + (a == b));
				System.out.println("The value of a != b is " + (a != b));
				System.out.println("The value of a > b is " + (a > b));
				System.out.println("The value of a < b is " + (a < b));
				System.out.println("The value of a >= b is " + (a >= b));
				System.out.println("The value of a <= b is " + (a <= b));
				break;
			default:
				System.out.println("Invalid Input");
				break;
		}
	}
}
```

```
C:\Users\admin\Desktop\Java\Assignment 1.3 - Bitwise, Logical, Relational Operators>java Operators
The value of a & b is 0
The value of a | b is 15
The value of a ^ b is 15
The value of ~a is -11
The value of a<<2 is 40
The value of a>>2 is 2
```