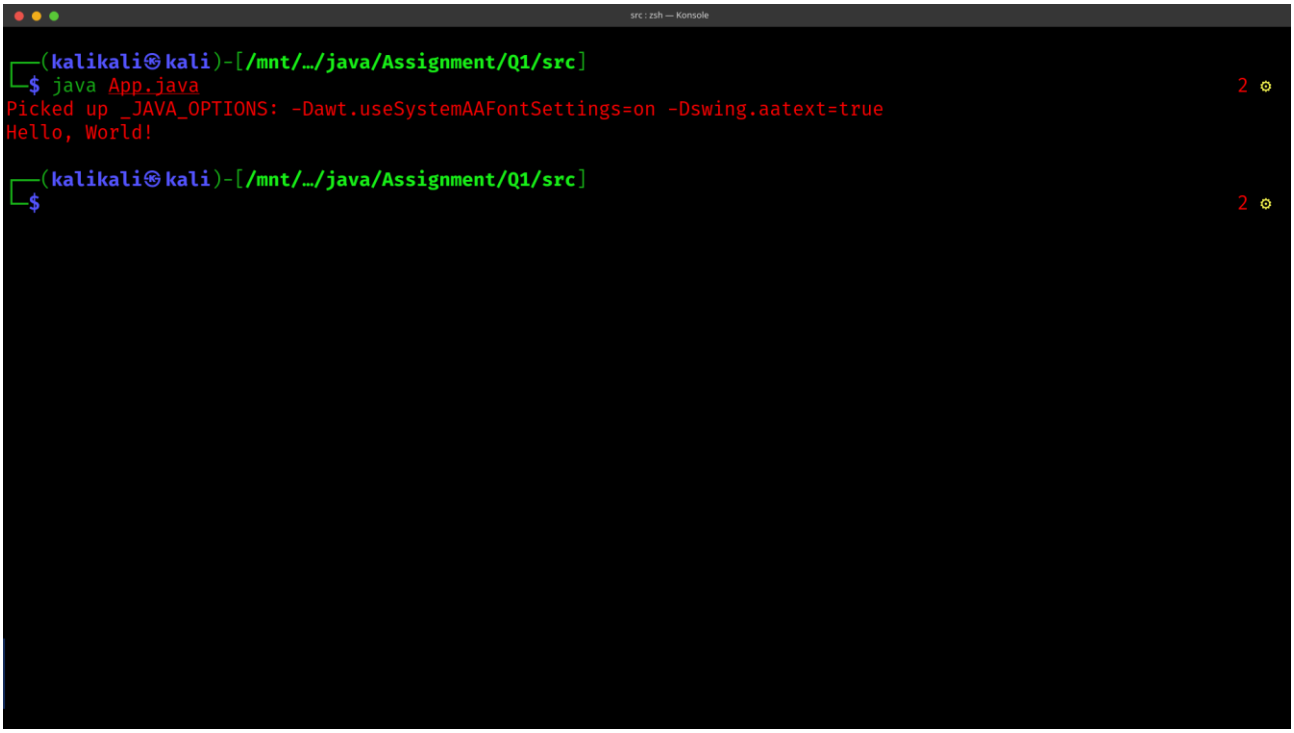
## Q1. WAP to print some statements like “Hello World!”.

**Code:**


**Output:**



## Q2. WAP to calculate room area using multiple classes.

**Code:**

```
src : vim — Konsole
import java.util.Scanner;

public class App {
    public static void main(String[] args) throws Exception {
        Scanner n = new Scanner(System.in);
        System.out.print("Enter the length of a Room :: ");
        double len =n.nextDouble();
        System.out.print("Enter the width of a Room :: ");
        double width =n.nextDouble();
        new Area(len, width);
    }
}
~
```

```
src : vim — Konsole
public class Area {
    Area(double length,double width){
        System.out.println("The area of a room is :: "+length*width);
    }
}
~
```

**output:**

```
bin : zsh — Konsole
┌──(kalikali㉿kali)-[/mnt/…/java/Assignment/Q2/bin]
└─$ java App                                                    5 ⚙
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Enter the length of a Room :: 300
Enter the width of a Room :: 1500
The area of a room is :: 450000.0
```

## Q3. WAP to demonstrate the use of command line arguments.

**Code:**

```
bin : vim — Konsole
public class App {
    public static void main(String[] args) throws Exception {
        for(int i=0;i<args.length;i++){
            System.out.println(args[i]);
        }
    }
}
~
```

**Output:**

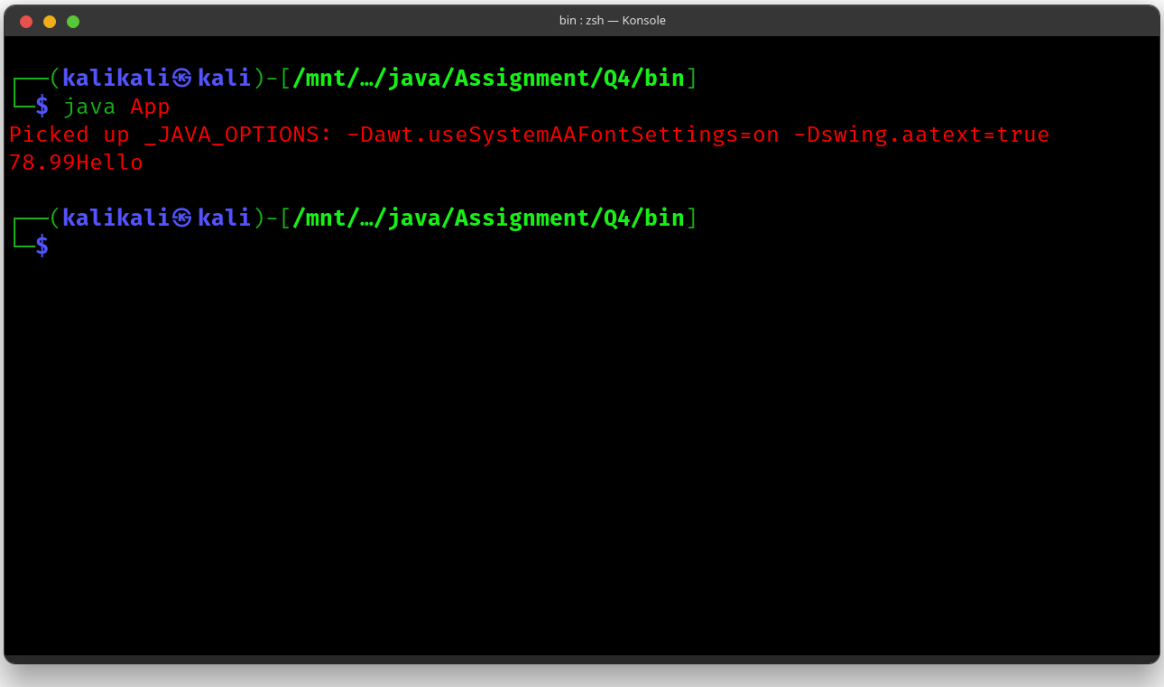
```
bin : zsh — Konsole

┌──(kalikali㉿kali)-[/mnt/…/java/Assignment/Q3/bin]
└─$ java App sahil_khan 21                                              6 ⚙
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
sahil_khan
21
```

## Q4. WAP to explain the basic data types used in java.

**Code:**

```
src : vim — Konsole

public class App {
    public static void main(String[] args) throws Exception {
        int myNum = 5; // Integer (whole number)
        float myFloatNum = 5.99f; // Floating point number
        char myLetter = 'D'; // Character
        boolean myBool = true; // Boolean
        String myText = "Hello"; // String
        if(myBool==true){
            System.out.println(myNum+myFloatNum+myLetter+myText);
        }
        else{
            System.out.println("Mybool is false");
        }
    }
}
~
~
~
~
~
"App.java" 15L, 501B                                          1,1           All
```

**Output:**




## Q5. WAP to explain the type casting in java.

**Code:**

src : vim — Konsole

```
public class App {
    public static void main(String[] args) throws Exception {
        System.out.println("There are Two types of Casting \n1. Implicit Casting\n2.
 Explicit Casting");
        new implicitCasting();
        new explicitCasting();
    }
}
~
~
~
~
~
~
~
~
~
~
~
~
"App.java" 7L, 256B                                          1,1           All
```

src : vim — Konsole

```
public class implicitCasting {
    implicitCasting(){
        System.out.println("\n\n1.Implicit Casting ");
        byte n=100;
        int m =n;
        System.out.println(n +" is of type "+((Object)n).getClass().getSimpleName());
        System.out.println("Casting byte to int!!");
        System.out.println(m +" is of type "+((Object)m).getClass().getSimpleName());
    }
}
~
~
~
~
~
~
~
~
~
~
~
                                                             1,1           All
```

src : vim — Konsole

```
public class explicitCasting {
    explicitCasting(){
        System.out.println("\n2. Explicit Casting");
        long longNum = 100;
        int intNum = (int) longNum;
        System.out.println(longNum +" is of type "+((Object)longNum).getClass().getSimpleName());
        System.out.println("Casting long to int!!");
        System.out.println(intNum +" is of type "+((Object)intNum).getClass().getSimpleName());
    }
}
~
~
~
~
~
~
~
~
~
~
~
~
~
                                                                1,1           All
```

**Output:**

src : zsh — Konsole

```
┌──(kalikali㉿kali)-[/mnt/…/java/Assignment/Q5/src]
└─$ java App                                                              3 ⚙
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
There are Two types of Casting
1. Implicit Casting
2. Explicit Casting


1.Implicit Casting
100 is of type Byte
Casting byte to int!!
100 is of type Integer
2. Explicit Casting
100 is of type Long
Casting long to int!!
100 is of type Integer

┌──(kalikali㉿kali)-[/mnt/…/java/Assignment/Q5/src]
└─$                                                                       3 ⚙
```

## Q6. WAP to demonstrate java expressions using different operators in java like relational, logical, bitwise operators etc.

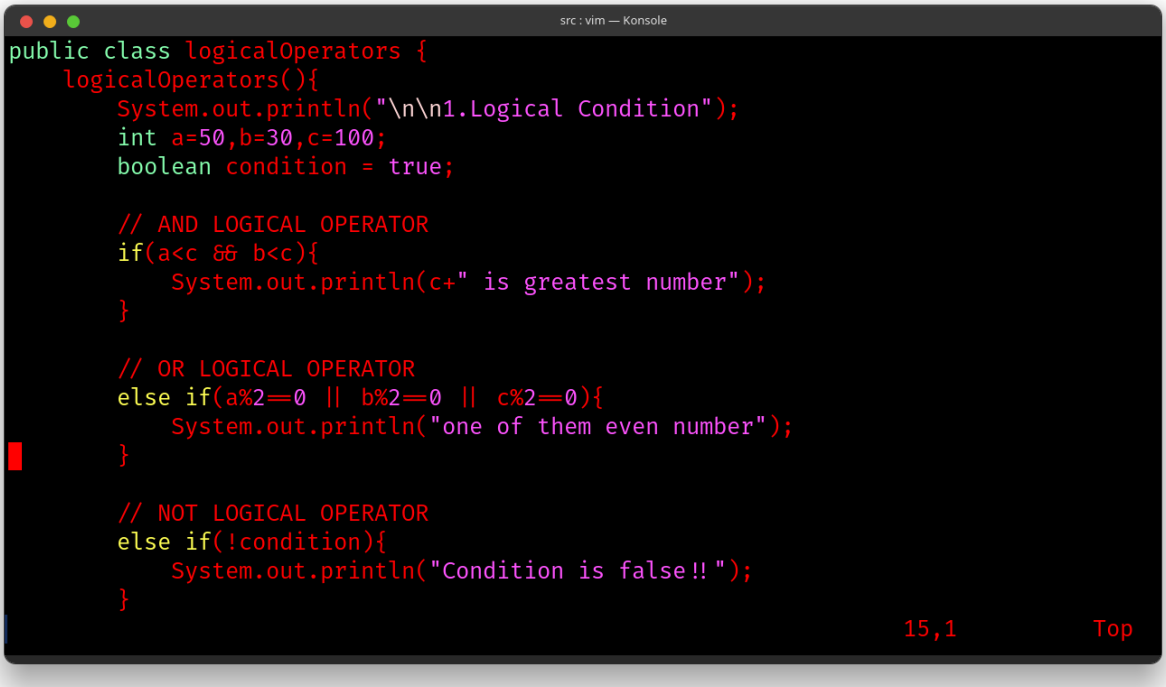
**Code:**

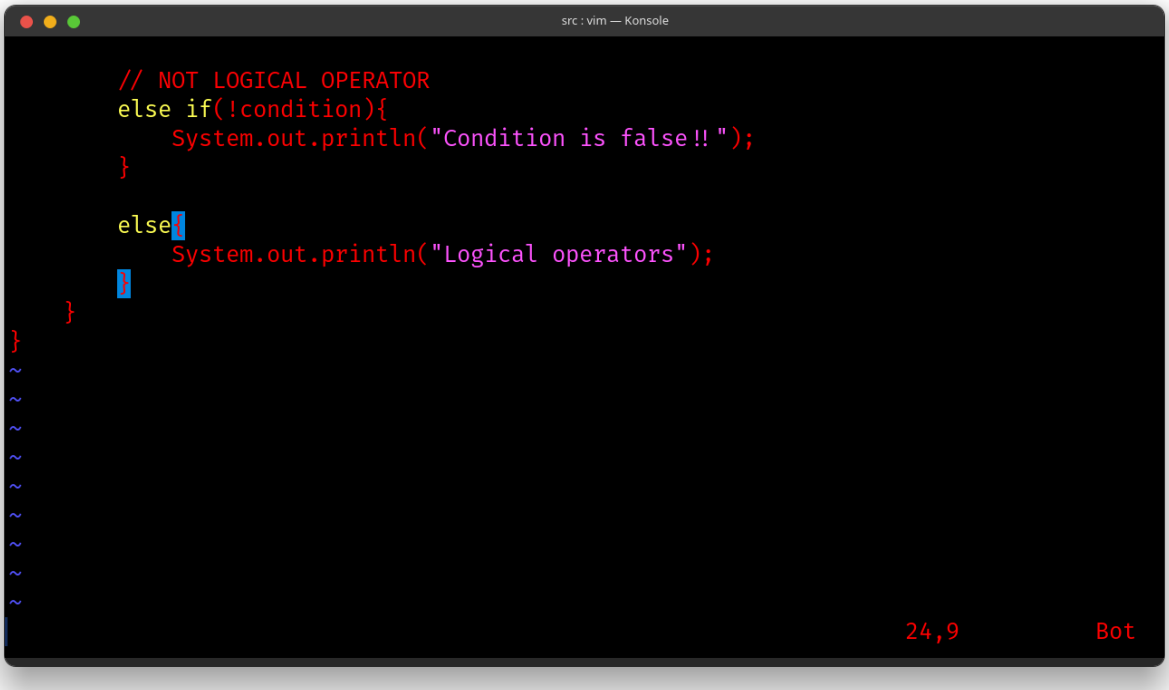
src : vim — Konsole

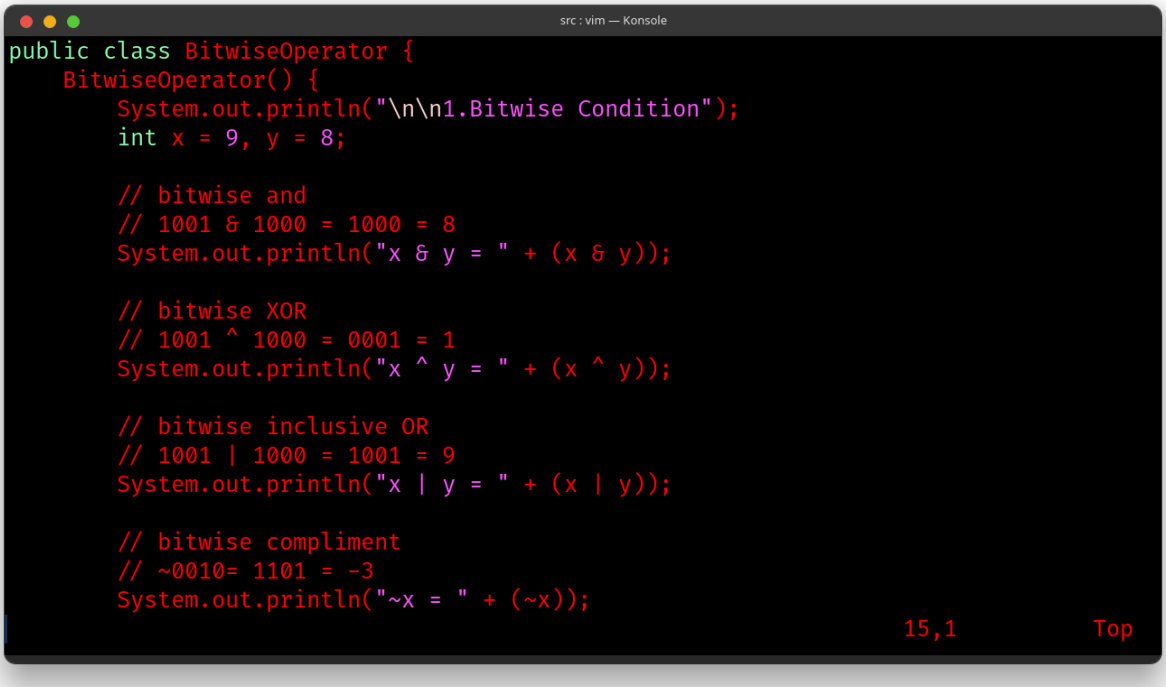
```
public class App {
    public static void main(String[] args) throws Exception {
        System.out.println("WAP to demonstrate java expressions using different oper
ators in java like relational, logical, bitwise operators etc.");
        new RelationalOperators();
        new logicalOperators();
        new BitwiseOperator();
    }
}
~
~
~
~
~
~
~
~
~
~
~
"App.java" 8L, 337B                                           1,1           All
```

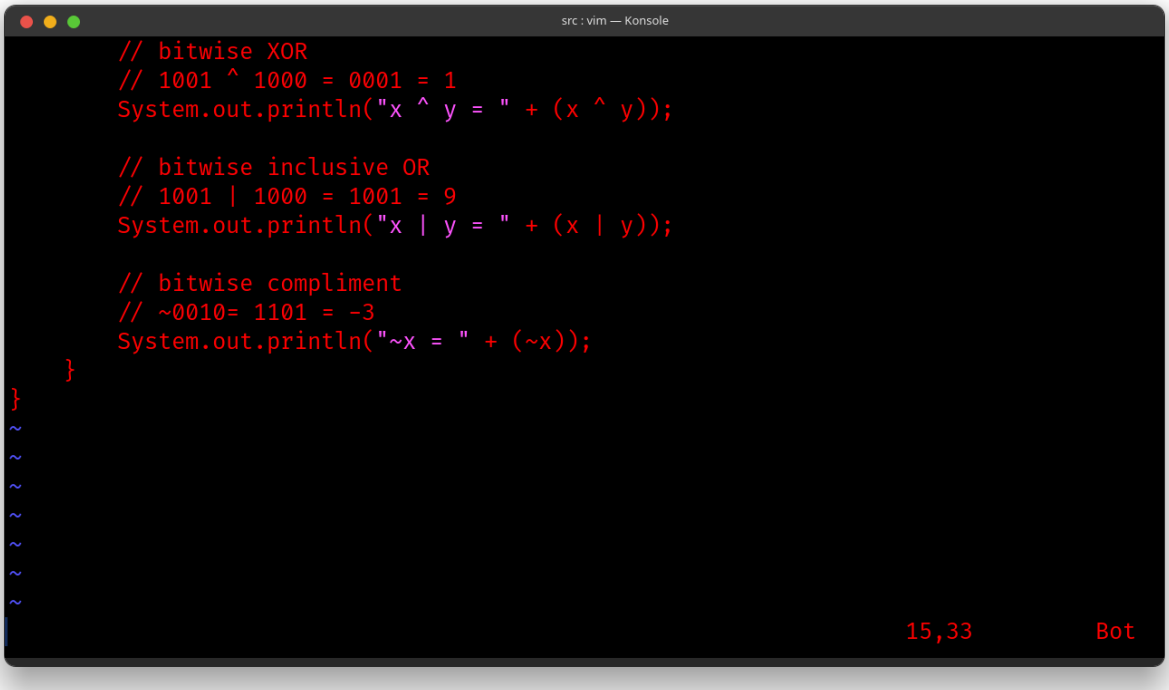
src : vim — Konsole

```
import java.util.Scanner;
public class RelationalOperators {
    RelationalOperators(){
        System.out.println("\n\n1.Rational Condition");
        System.out.println("Enter the value of i :: ");
        Scanner n = new Scanner(System.in);
        int i=n.nextInt();
        if( i==0){
            System.out.println("i is 0");
        }
        else if(i==1){
            System.out.println("i is 1");
        }
        else{
            System.out.println("All of above rational condition are false");
        }
    }
}
~
~
"RelationalOperators.java" 18L, 526B                          1,1           All
```

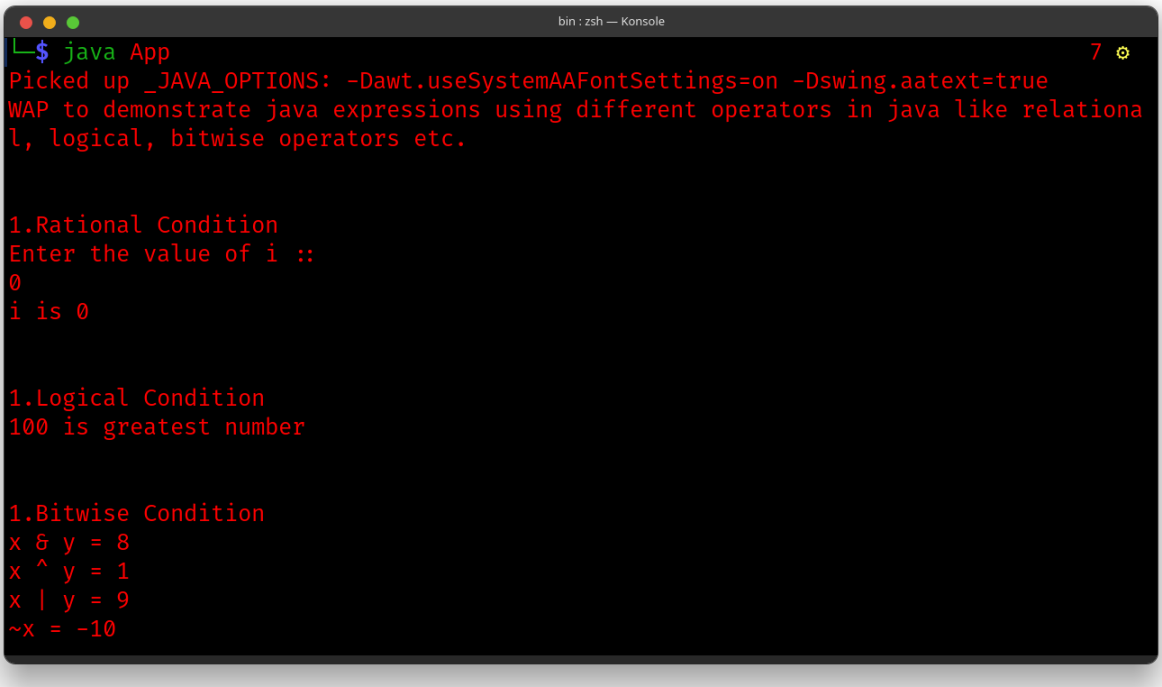

src : vim — Konsole
public class logicalOperators {
logicalOperators(){
System.out.println("\n\n1.Logical Condition");
int a=50,b=30,c=100;
boolean condition = true;
// AND LOGICAL OPERATOR
if(a<c && b<c){
System.out.println(c+" is greatest number");
}
// OR LOGICAL OPERATOR
else if(a%2==0 || b%2==0 || c%2==0){
System.out.println("one of them even number");
}
// NOT LOGICAL OPERATOR
else if(!condition){
System.out.println("Condition is false!!");
}
15,1 Top


src : vim — Konsole
// NOT LOGICAL OPERATOR
else if(!condition){
System.out.println("Condition is false!!");
}
else{
System.out.println("Logical operators");
}
}
}
24,9 Bot


src : vim — Konsole
public class BitwiseOperator {
BitwiseOperator() {
System.out.println("\n\n1.Bitwise Condition");
int x = 9, y = 8;
// bitwise and
// 1001 & 1000 = 1000 = 8
System.out.println("x & y = " + (x & y));
// bitwise XOR
// 1001 ^ 1000 = 0001 = 1
System.out.println("x ^ y = " + (x ^ y));
// bitwise inclusive OR
// 1001 | 1000 = 1001 = 9
System.out.println("x | y = " + (x | y));
// bitwise compliment
// ~0010= 1101 = -3
System.out.println("~x = " + (~x));
15,1 Top


src : vim — Konsole
// bitwise XOR
// 1001 ^ 1000 = 0001 = 1
System.out.println("x ^ y = " + (x ^ y));
// bitwise inclusive OR
// 1001 | 1000 = 1001 = 9
System.out.println("x | y = " + (x | y));
// bitwise compliment
// ~0010= 1101 = -3
System.out.println("~x = " + (~x));
}
}
15,33 Bot

**Output:**




## Q7. WAP to demonstrate if-else, nested if-else, if-else ladder.

**Code:**

src : vim — Konsole

```
public class App {
    public static void main(String[] args) throws Exception {
        System.out.println("WAP to demonstrate if-else, nested if-else, if-else ladd
er.");
        new ifElse();
        new nestedIfelse();
        new ladderIfelse();
    }
}
~
~
~
~
~
~
~
~
~
~
~
"App.java" 8L, 258B                                    1,1           All
```

src : vim — Konsole

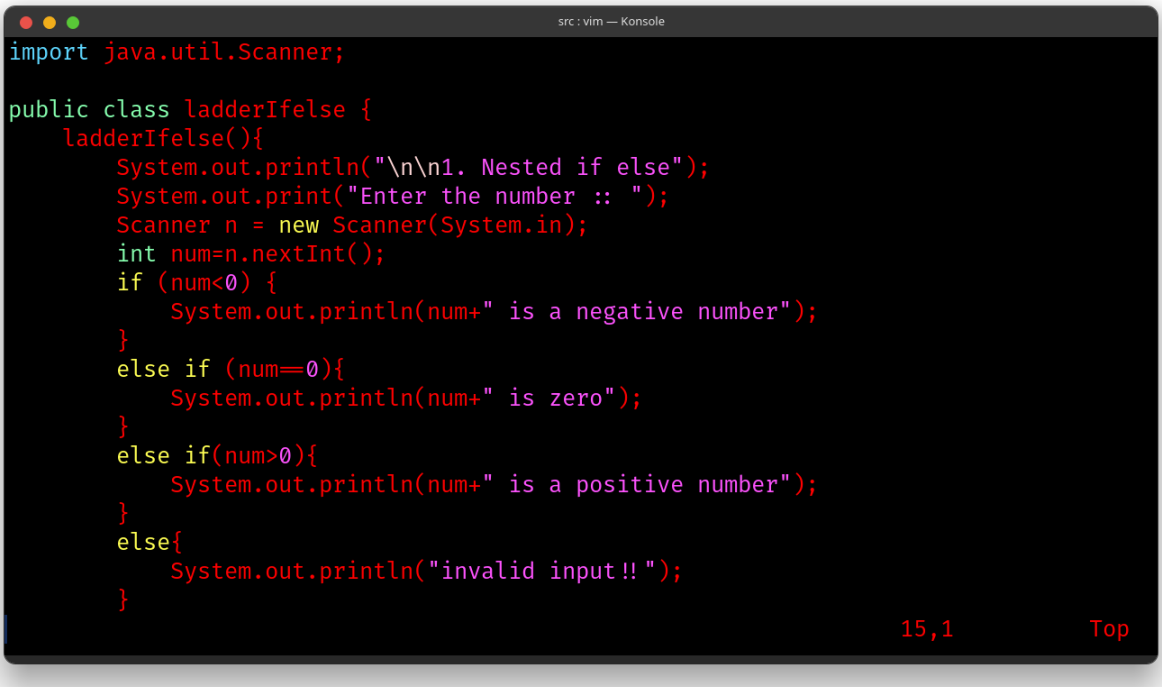
```
import java.util.Scanner;

public class ifElse {
    ifElse(){
        System.out.println("\n\n1. if else");
        System.out.print("Enter the number :: ");
        Scanner n = new Scanner(System.in);
        int num=n.nextInt();
        if(num%2==0){
            System.out.println(num+" is a even number");
        }
        else{
            System.out.println(num+" is a odd number");
        }
    }
}
~
~
~
~
"ifElse.java" 16L, 409B                                1,1           All
```

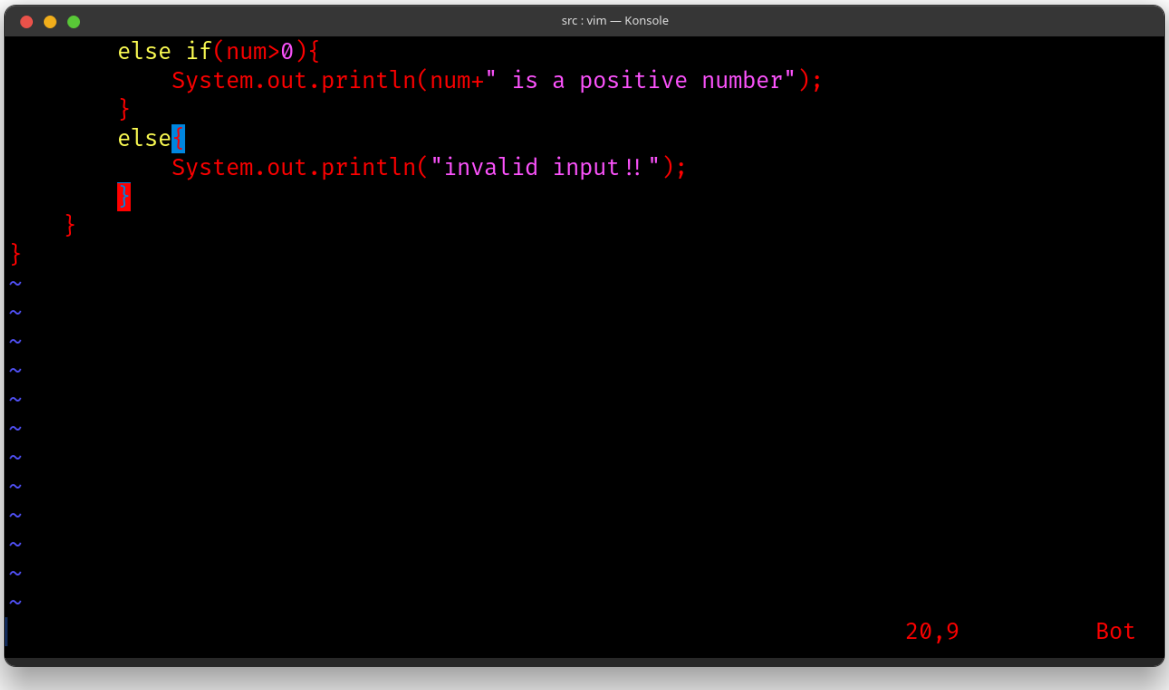
src : vim — Konsole

```
import java.util.Scanner;

public class nestedIfelse {
    nestedIfelse(){
        System.out.println("\n\n1. Nested if else");
        System.out.print("Enter the number :: ");
        Scanner n = new Scanner(System.in);
        int num=n.nextInt();
        if(num<100){
            if(num%2==0){
                System.out.println(num+" is smaller than 100 and even number");
            }
            else{
                System.out.println(num+" is smaller than 100 but not even number (od
d number)");
            }
        }
        else{
            System.out.println(num+" is greater than 100");
        }
                                                              14,1          Top
```

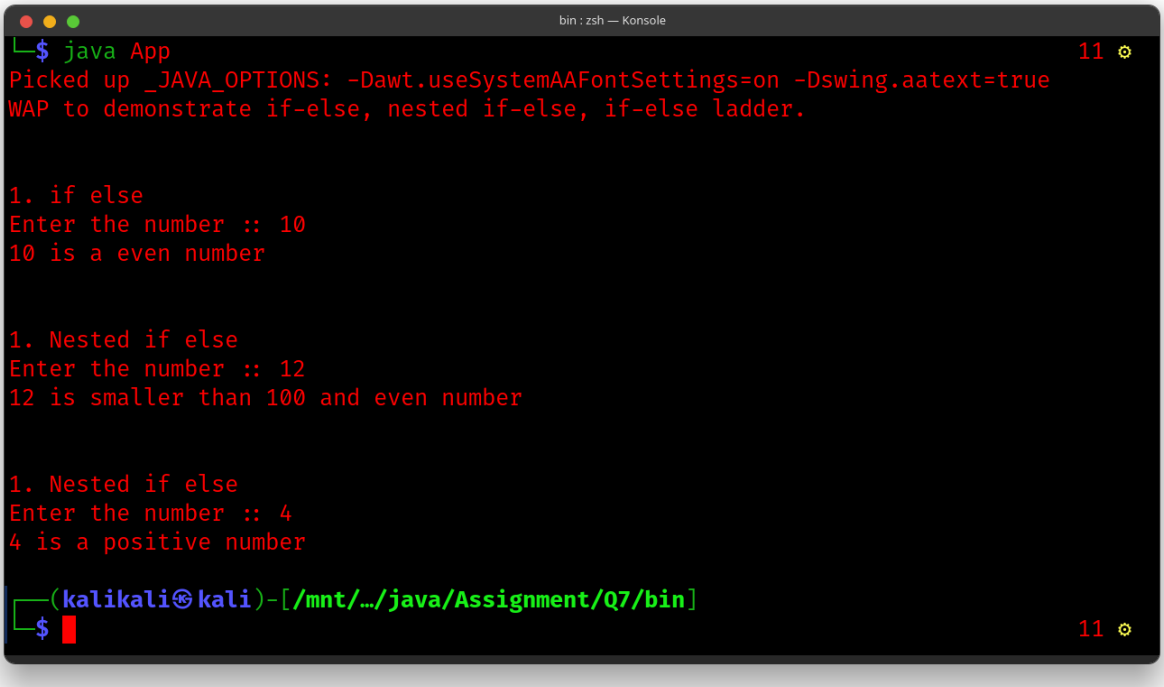
src : vim — Konsole

```
            if(num%2==0){
                System.out.println(num+" is smaller than 100 and even number");
            }
            else{
                System.out.println(num+" is smaller than 100 but not even number (od
d number)");
            }
        }
        else{
            System.out.println(num+" is greater than 100");
        }
    }
}
~
~
~
~
~
~
~
~
                                                              15,13         Bot
```


src : vim — Konsole
import java.util.Scanner;
public class ladderIfelse {
ladderIfelse(){
System.out.println("\n\n1. Nested if else");
System.out.print("Enter the number :: ");
Scanner n = new Scanner(System.in);
int num=n.nextInt();
if (num<0) {
System.out.println(num+" is a negative number");
}
else if (num==0){
System.out.println(num+" is zero");
}
else if(num>0){
System.out.println(num+" is a positive number");
}
else{
System.out.println("invalid input!!");
}
15,1 Top


src : vim — Konsole
else if(num>0){
System.out.println(num+" is a positive number");
}
else{
System.out.println("invalid input!!");
}
}
}
20,9 Bot

**Output:**




## Q8. WAP to demonstrate switch statements.

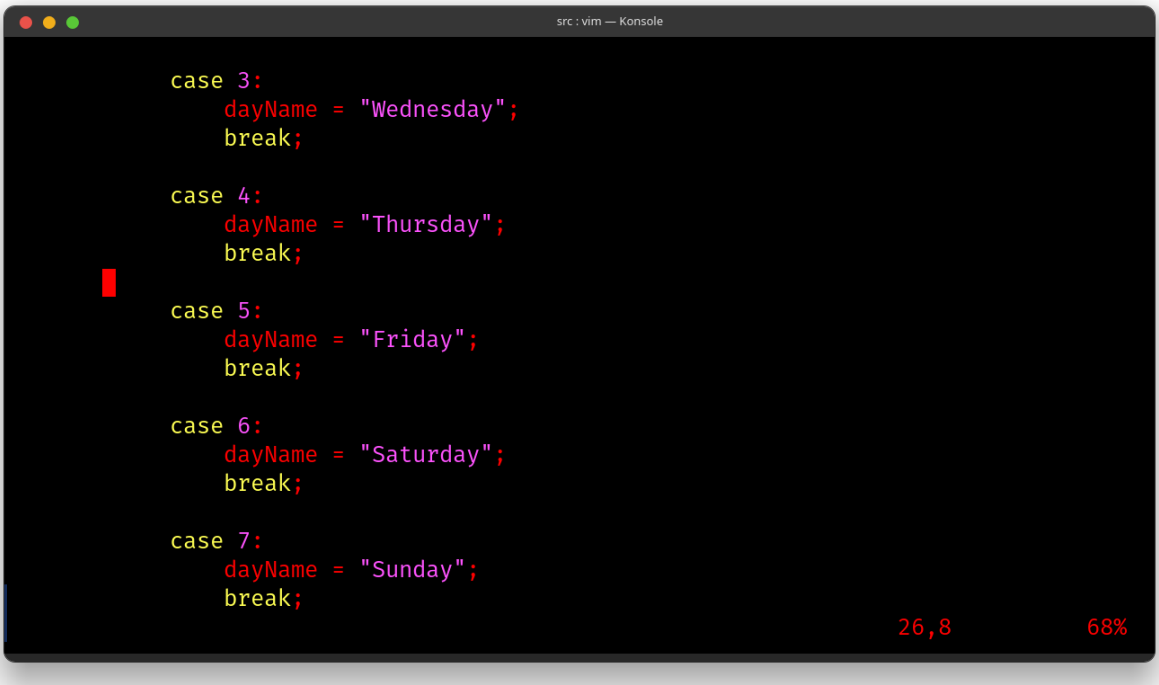
**Code:**

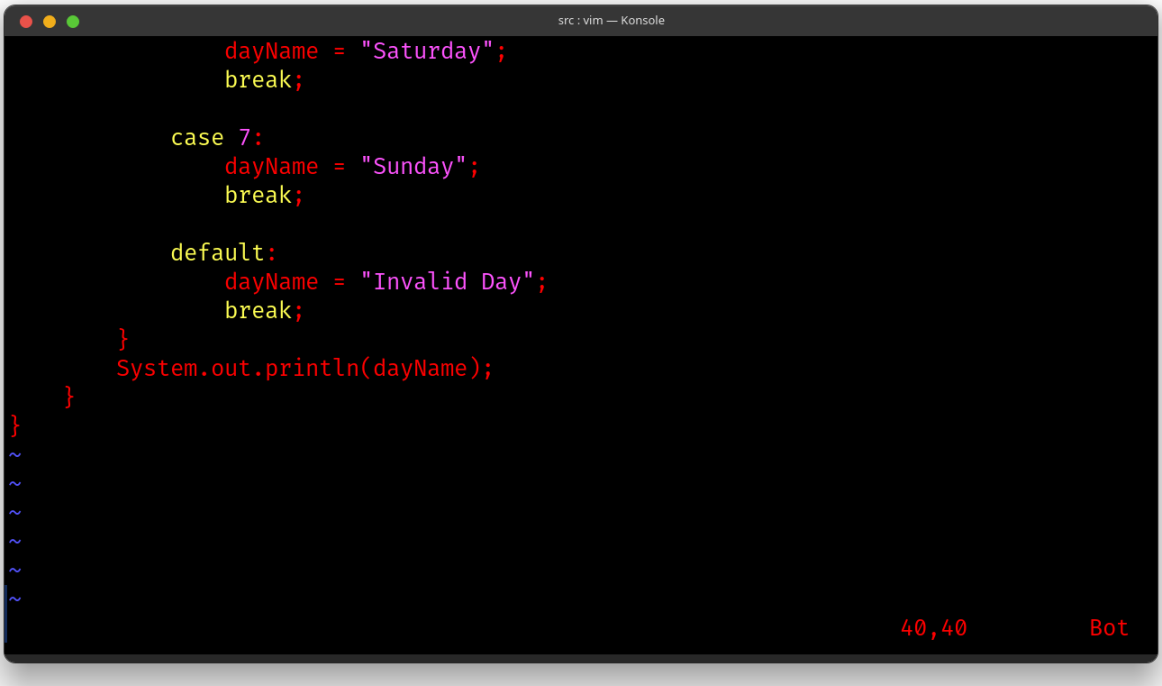
src : vim — Konsole

```
public class App {
    public static void main(String[] args) throws Exception {
        System.out.println("Switch case!!");
        new switchCase();
    }
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"App.java" 6L, 160B                                    1,1           All
```

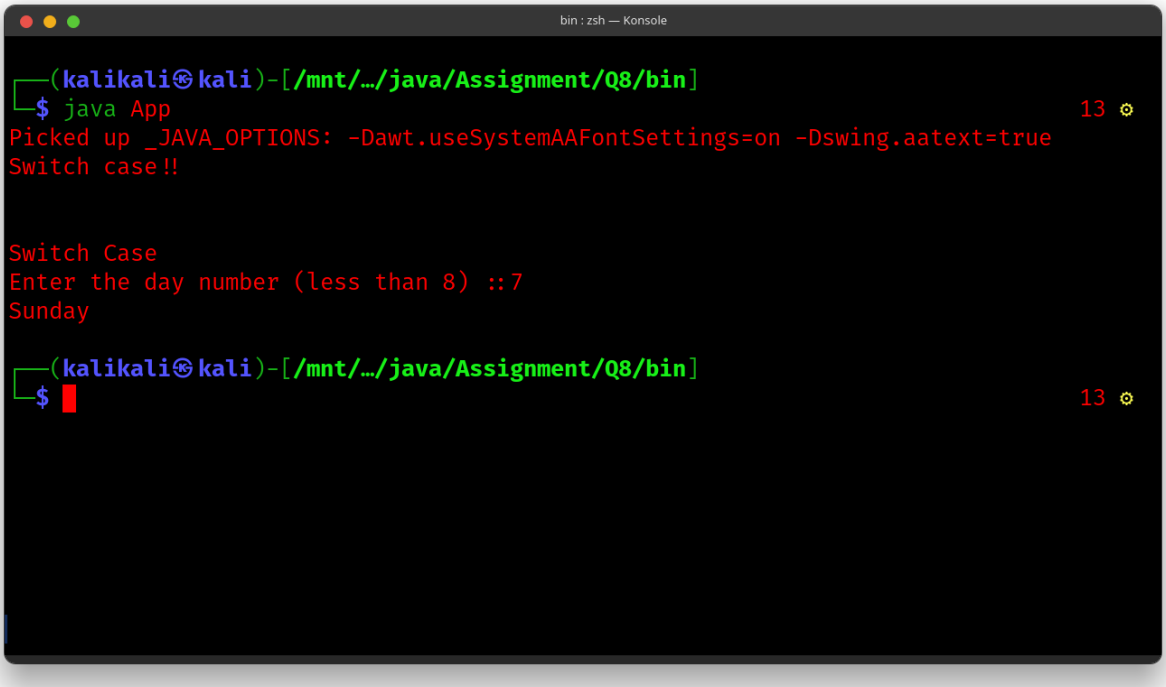
src : vim — Konsole

```
import java.util.Scanner;

public class switchCase {
    switchCase(){
        System.out.println("\n\nSwitch Case");
        System.out.print("Enter the day number (less than 8) ::");
        Scanner n = new Scanner(System.in);
        int dayNum = n.nextInt();
        String dayName;
        switch (dayNum) {
            case 1:
                dayName = "Monday";
                break;

            case 2:
                dayName = "Tuesday";
                break;

            case 3:
                dayName = "Wednesday";
                                                       12,1          Top
```


src : vim — Konsole
case 3:
dayName = "Wednesday";
break;
case 4:
dayName = "Thursday";
break;
case 5:
dayName = "Friday";
break;
case 6:
dayName = "Saturday";
break;
case 7:
dayName = "Sunday";
break;
26,8 68%

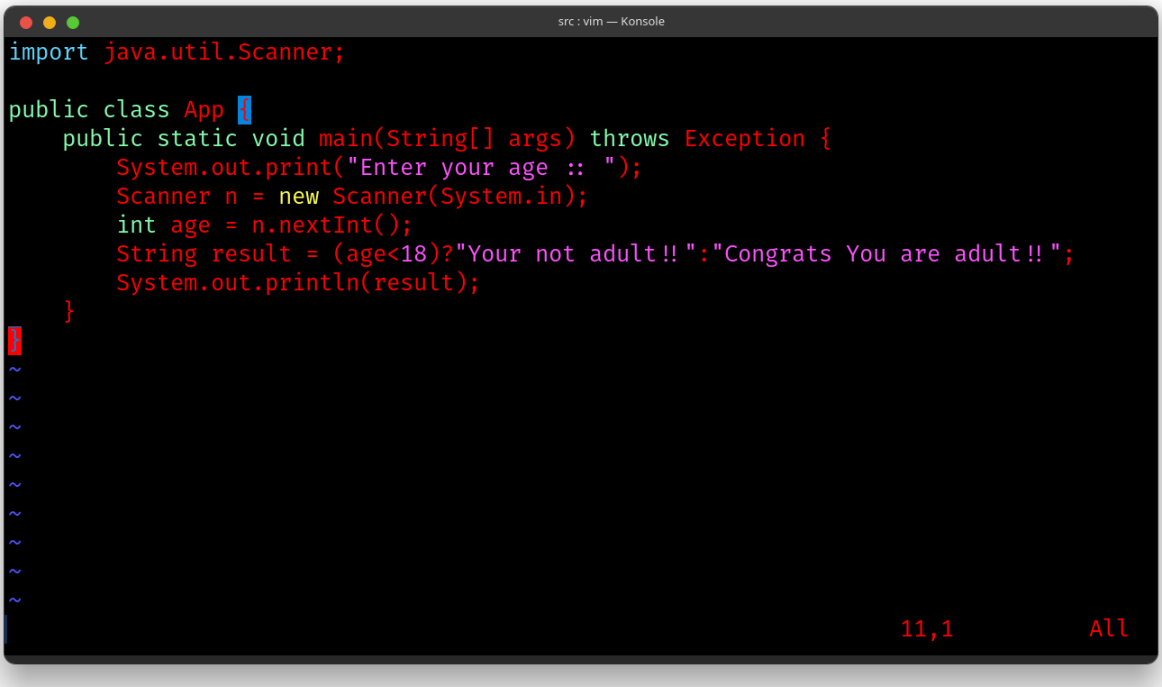

src : vim — Konsole
dayName = "Saturday";
break;
case 7:
dayName = "Sunday";
break;
default:
dayName = "Invalid Day";
break;
}
System.out.println(dayName);
}
}
40,40 Bot

**Output:**

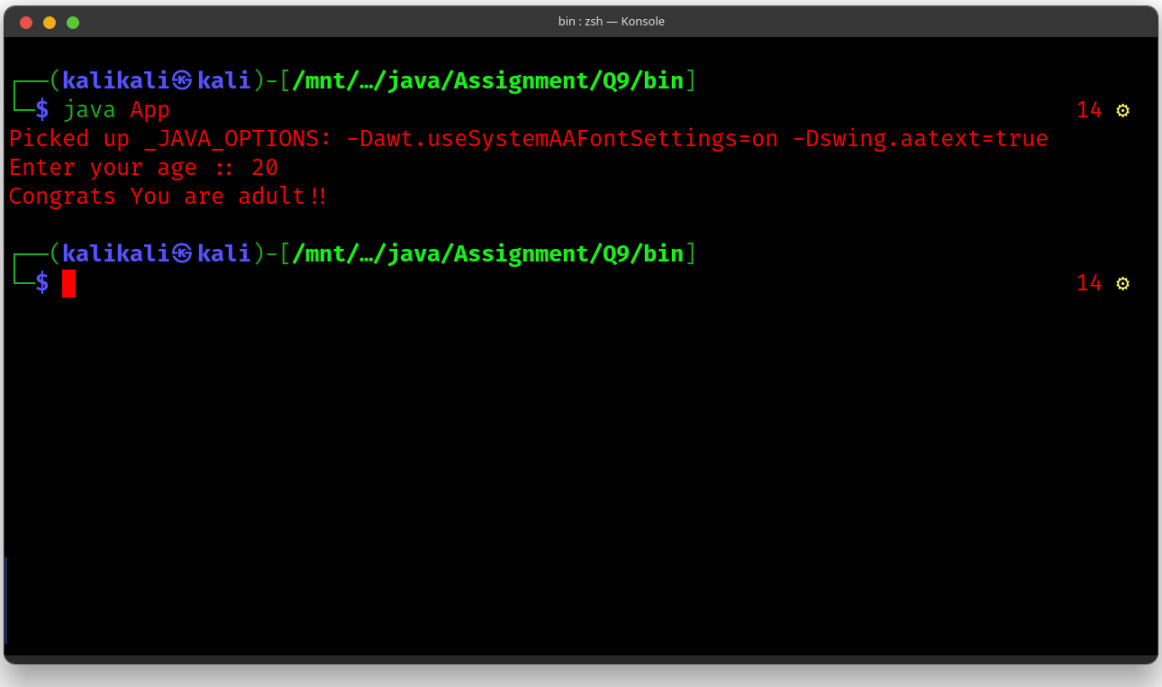



## Q9. WAP to demonstrate "? :" operator.

## Code:



## Output:




## Q10. WAP to explain the working of while, for and do-while loops.

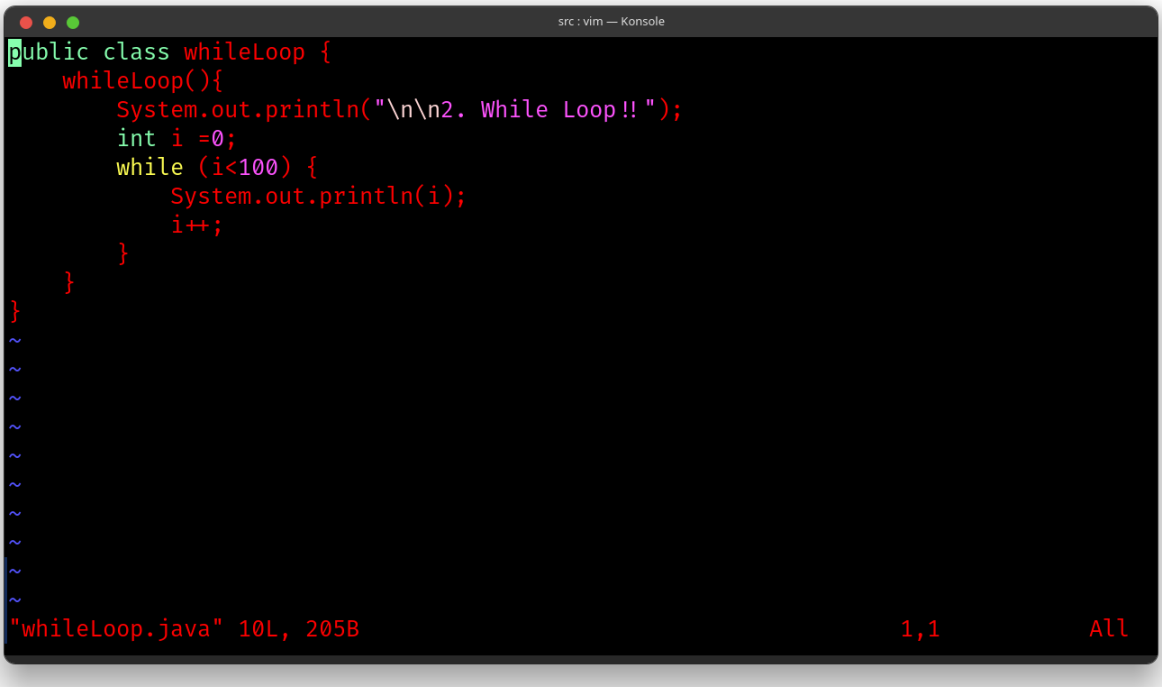
## Code:

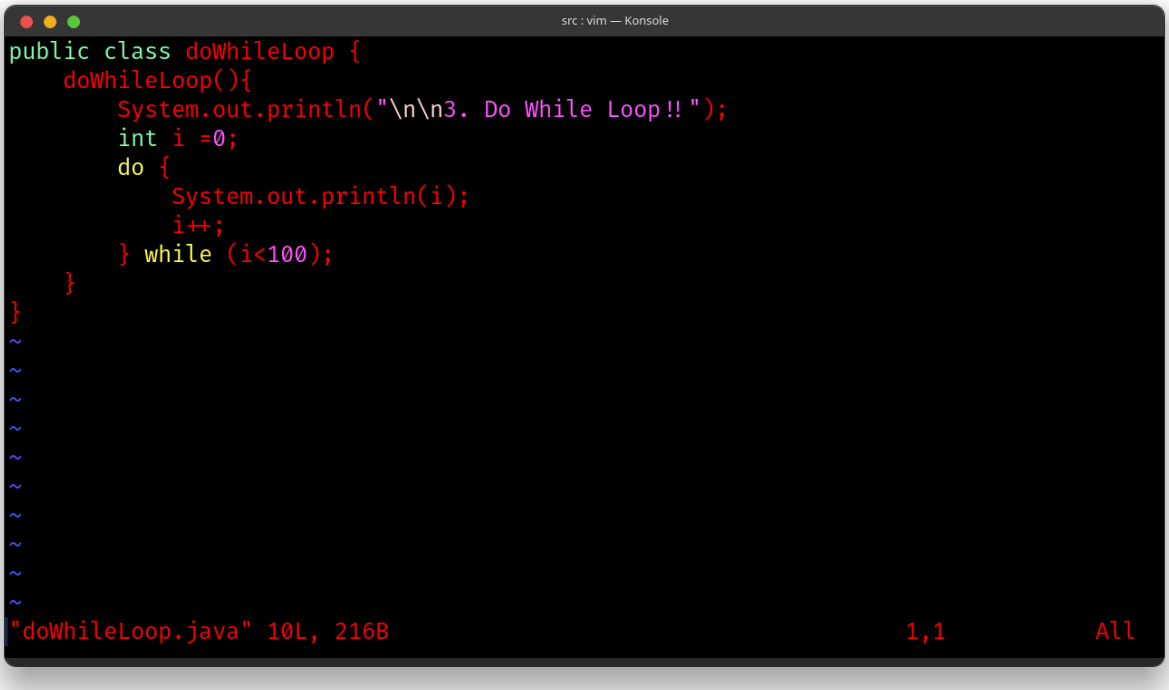
src : vim — Konsole

```
public class App {
    public static void main(String[] args) throws Exception {
        new forLoop();
        new whileLoop();
        new doWhileLoop();
    }
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"App.java" 7L, 166B                                    1,1           All
```

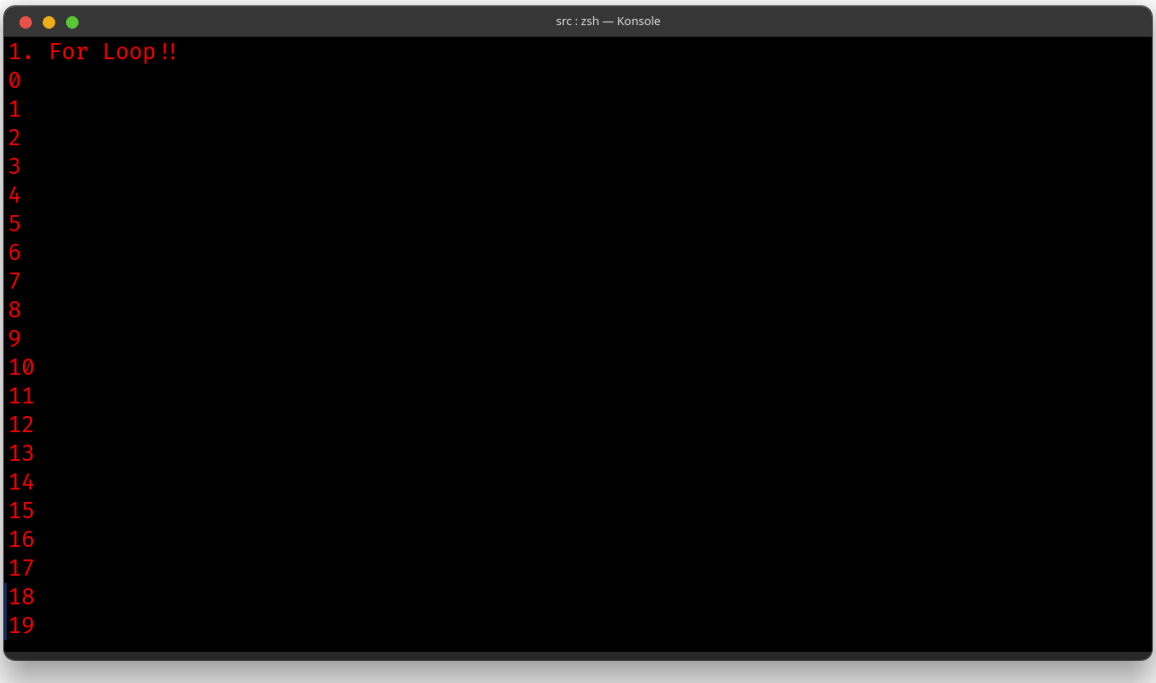
src : vim — Konsole

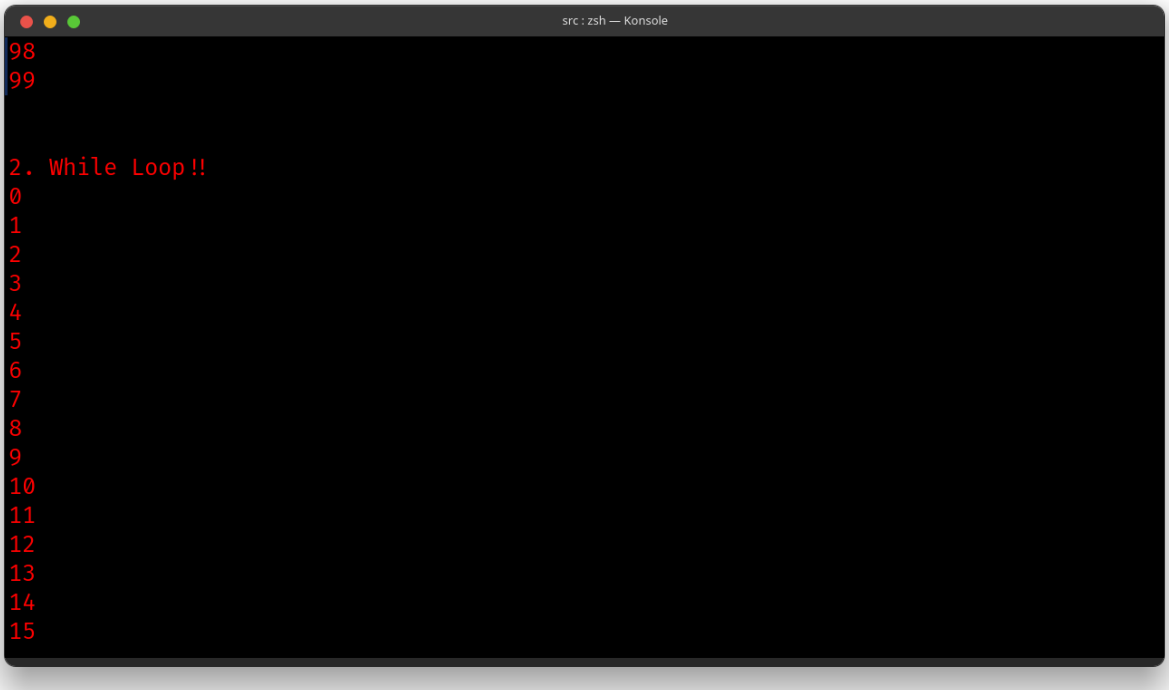
```
public class forLoop {
    forLoop(){
        int i;
        System.out.println("\n1. For Loop!!");
        for(i=0;i<100;i++){
            System.out.println(i);
        }
    }
}
~
~
~
~
~
~
~
~
~
~
~
~
"forLoop.java" 9L, 181B                                1,1           All
```

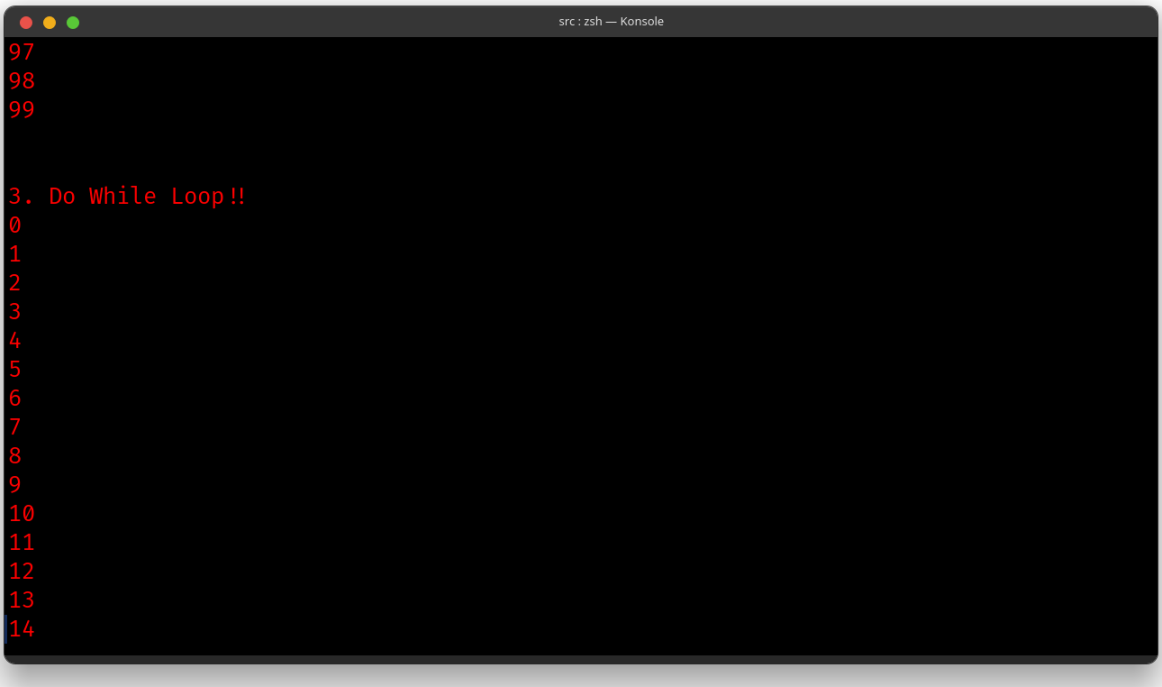

src : vim — Konsole
public class whileLoop {
whileLoop(){
System.out.println("\n\n2. While Loop!!");
int i =0;
while (i<100) {
System.out.println(i);
i++;
}
}
}
"whileLoop.java" 10L, 205B 1,1 All


src : vim — Konsole
public class doWhileLoop {
doWhileLoop(){
System.out.println("\n\n3. Do While Loop!!");
int i =0;
do {
System.out.println(i);
i++;
} while (i<100);
}
}
"doWhileLoop.java" 10L, 216B 1,1 All

## Output:


src : zsh — Konsole
1. For Loop !!
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

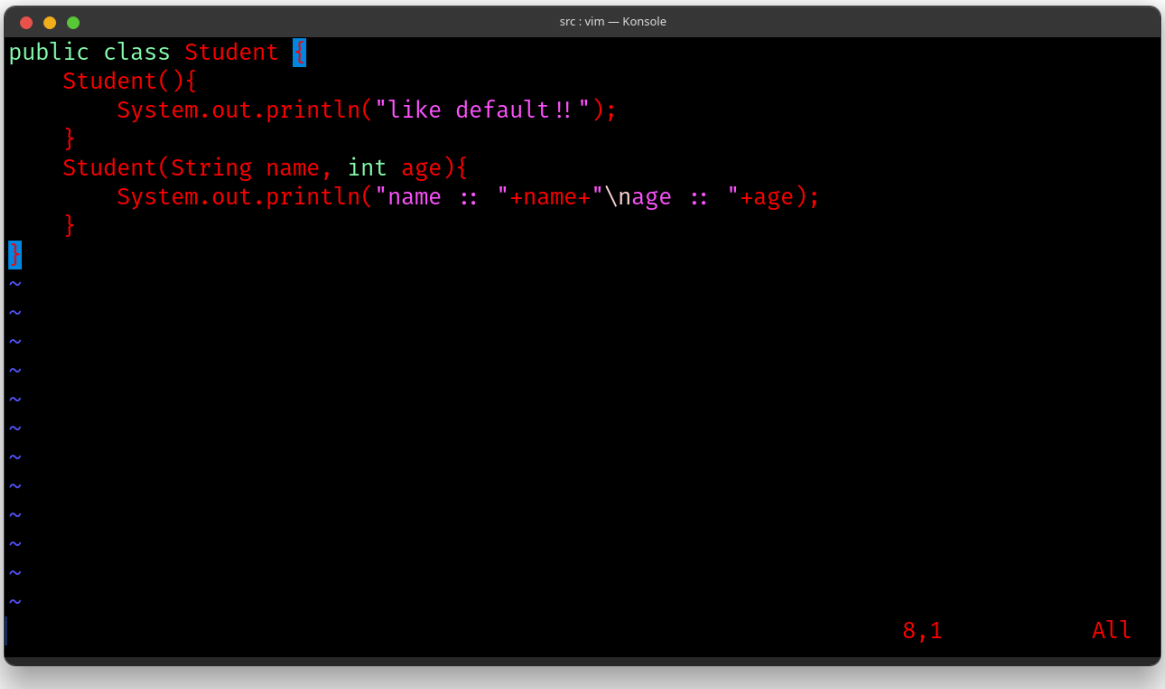

src : zsh — Konsole
98
99
2. While Loop !!
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15



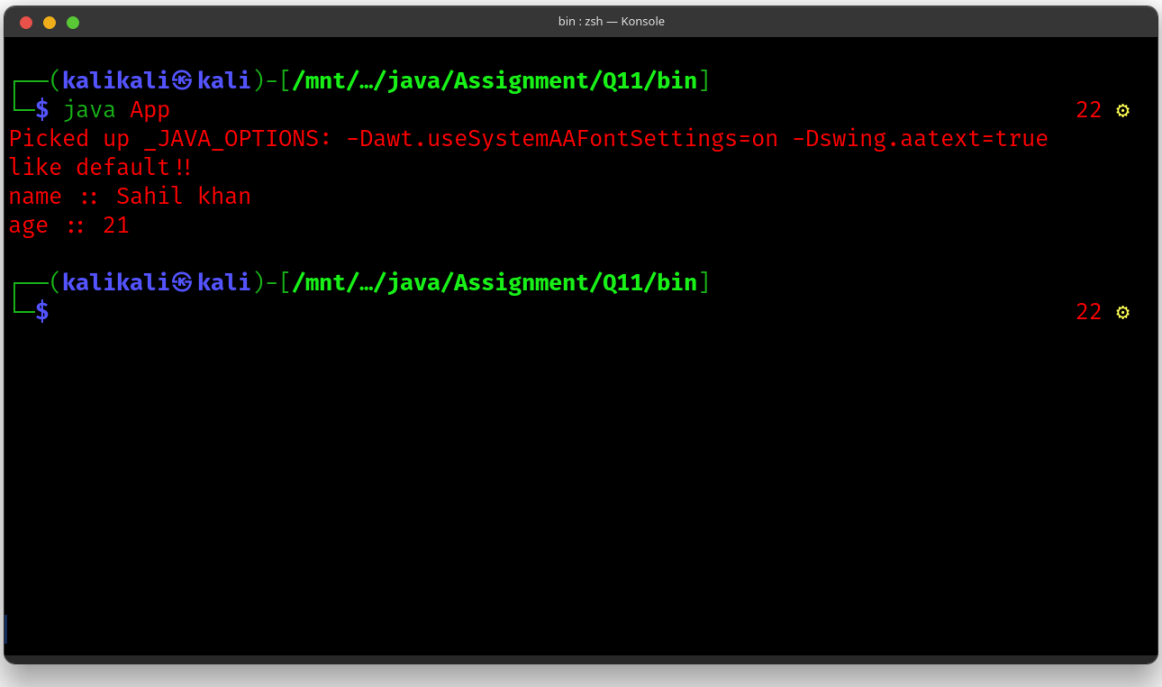

**Q11. WAP to demonstrate different types of constructors in java like default and parameterized.**

**Code:**


src : vim — Konsole
public class App {
public static void main(String[] args) throws Exception {
new Student();
new Student("Sahil khan", 21);
}
}
"App.java" 6L, 151B
1,1
All


src : vim — Konsole
public class Student {
Student(){
System.out.println("like default!!");
}
Student(String name, int age){
System.out.println("name :: "+name+"\nage :: "+age);
}
}
8,1
All

## Output:




## Q12. WAP to explain method overloading and constructor overloading.

## Code:

src : vim — Konsole

```
public class App {
    public static void main(String[] args) throws Exception {
        System.out.println("Method overloading and constructor overloading!!");
        new Add();
        new Add(5, 10);
        new Add(5.5, 10.5);
        Sub num = new Sub();
        num.subtract(10, 5);
        num.subtract(100.5, 200.4);
    }
}
~
~
~
~
~
~
~
~
~
"App.java" 11L, 334B                                    1,1           All
```

src : vim — Konsole

```
public class Add {
    Add(){
        System.out.println("\n\n1. Consrutor Overloading!!");
    }
    Add(int a, int b){
        System.out.println("The sum of "+a+" and "+b+" is :: "+(a+b));
    }
    Add(double a, double b){
        System.out.println("The sum of "+a+" and "+b+" is :: "+(a+b));
    }
}
~
~
~
~
~
~
~
~
~
"Add.java" 11L, 306B                                    1,1           All
```

src : vim — Konsole

```
public class Sub {
    Sub(){
        System.out.println("\n\n2. Method Overloading!!");
    }
    void subtract(int a,int b){
        System.out.println("The subtraction :: "+(a-b));
    }
    void subtract(double a, double b){
        System.out.println("The subtraction :: "+(a-b));
    }
}
~
~
~
~
~
~
~
~
~
"Sub.java" 11L, 294B                                    1,1           All
```

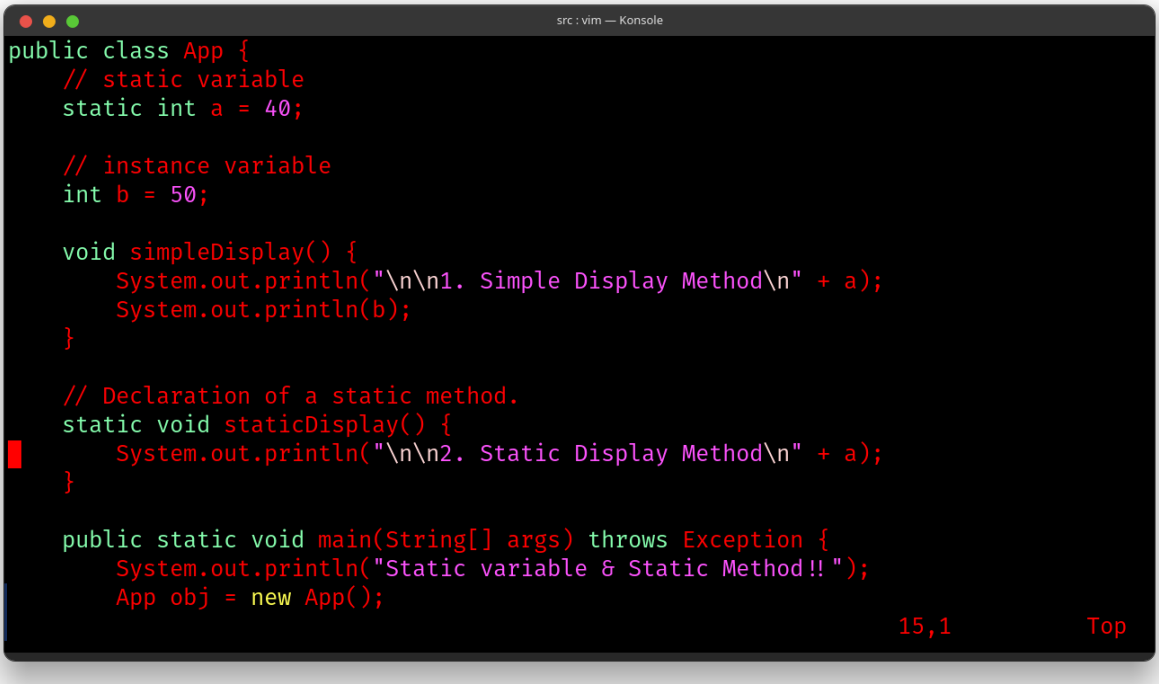
## Output:

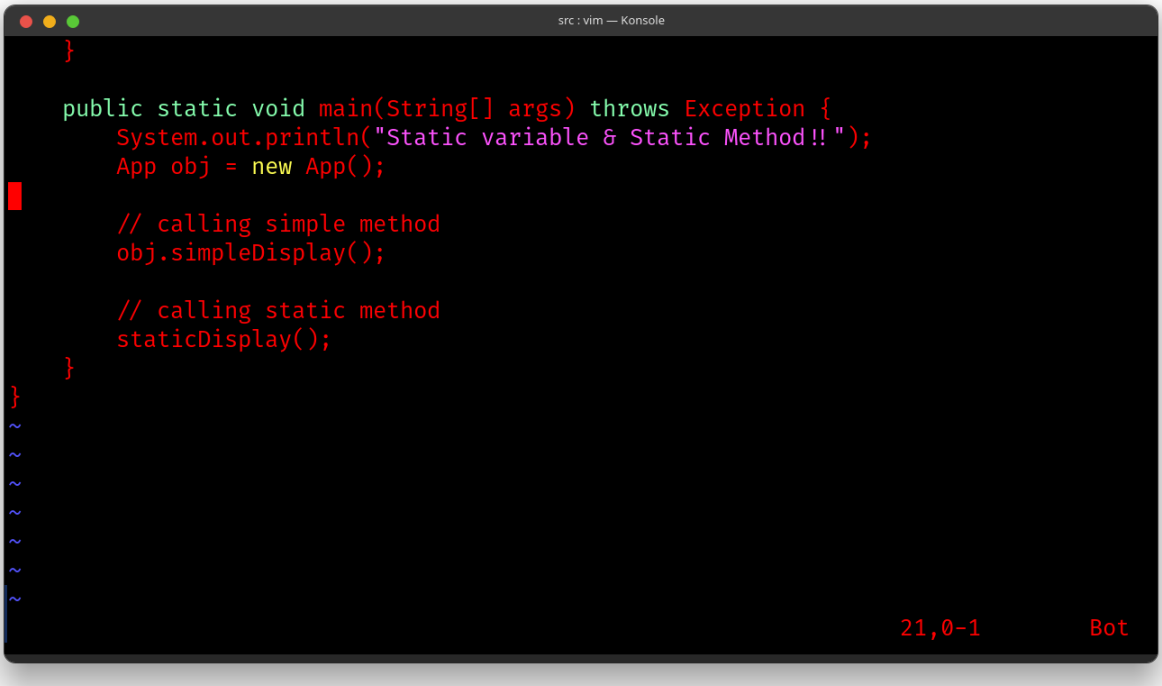
bin : zsh — Konsole

```
┌──(kalikali㉿kali)-[/mnt/…/java/Assignment/Q12/bin]
└─$ java App                                                          25 ⚙
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Method overloading and constructor overloading!!


1. Consrutor Overloading!!
The sum of 5 and 10 is :: 15
The sum of 5.5 and 10.5 is :: 16.0


2. Method Overloading!!
The subtraction :: 5
The subtraction :: -99.9

┌──(kalikali㉿kali)-[/mnt/…/java/Assignment/Q12/bin]
└─$                                                                   25 ⚙
```

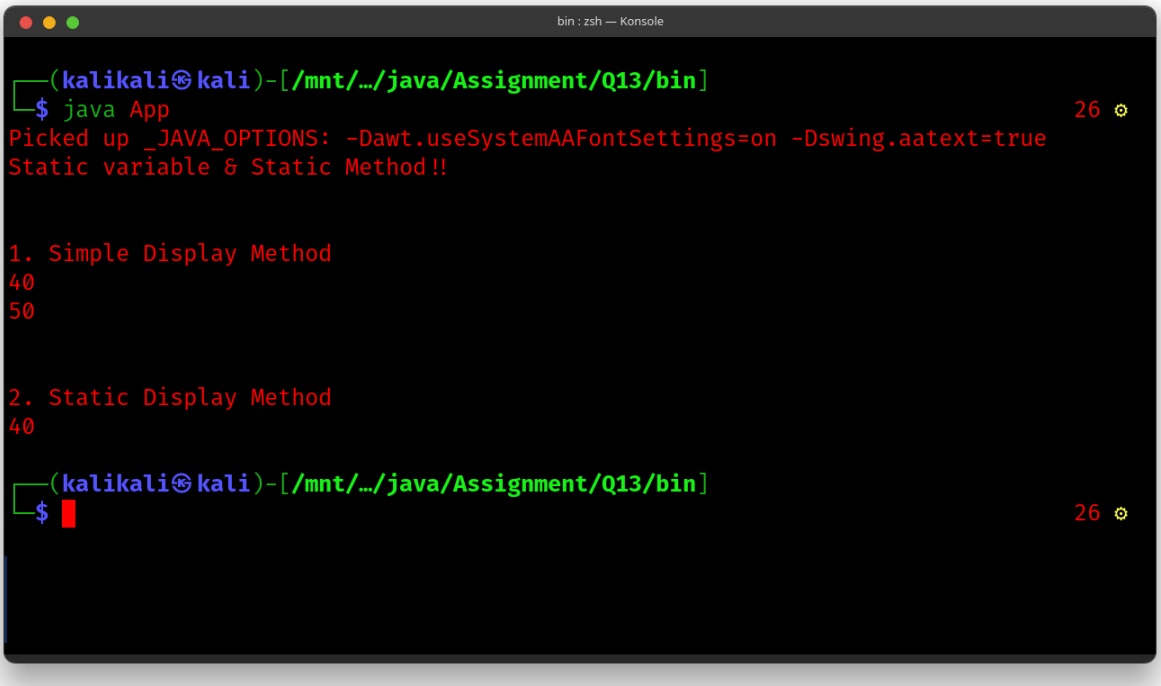
## Q13. WAP to explain static methods and static members.

## Code:


src : vim — Konsole
public class App {
// static variable
static int a = 40;
// instance variable
int b = 50;
void simpleDisplay() {
System.out.println("\n\n1. Simple Display Method\n" + a);
System.out.println(b);
}
// Declaration of a static method.
static void staticDisplay() {
System.out.println("\n\n2. Static Display Method\n" + a);
}
public static void main(String[] args) throws Exception {
System.out.println("Static variable & Static Method!!");
App obj = new App();
15,1 Top


src : vim — Konsole
}
public static void main(String[] args) throws Exception {
System.out.println("Static variable & Static Method!!");
App obj = new App();
// calling simple method
obj.simpleDisplay();
// calling static method
staticDisplay();
}
}
21,0-1 Bot

## Output:




## Q14. WAP to demonstrate multilevel inheritance and super keyword.

## Code:

src : vim — Konsole

```
public class App {
    public static void main(String[] args) throws Exception {
        Maruti800 car1 = new Maruti800();
        car1.vehicleType();
        car1.warranty();
        car1.speed();
    }
}
~
~
~
~
~
~
~
~
~
~
~
~
"App.java" 8L, 206B                                    1,1           All
```

src : vim — Konsole

```
class Car {
    String warranty ="10 yr";
    Car(){
        System.out.println("Class Car");
    }
    void vehicleType(){
        System.out.println("Vehicle Type : 4 wheeler");
    }
}
class Maruti extends Car{
    Maruti(){
        System.out.println("Class Maruti");
    }
    void brand(){
        System.out.println("Brand : Maruti");
    }
    void speed(){
        System.out.println("Max : 200kmph");
    }
    void warranty(){
                                                       15,1          Top
```

```
src : vim — Konsole
    }
    void warranty(){
        System.out.println("Warranty : "+super.warranty);
    }
}
class Maruti800 extends Maruti{
    Maruti800(){
        System.out.println("Maruti Model : 800");
    }
    void speed(){
        System.out.println("Max : 100kmph");
    }
    void warranty(){
        System.out.println("Warranty : "+super.warranty);
    }
}
~
~
~
~
                                                    24,31         Bot
```

## Output:

```
bin : zsh — Konsole
┌──(kalikali㉿kali)-[/mnt/…/java/Assignment/Q14/bin]
└─$ java App                                                    28 ⚙
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Class Car
Class Maruti
Maruti Model : 800
Vehicle Type : 4 wheeler
Warranty : 10 yr
Max : 100kmph

┌──(kalikali㉿kali)-[/mnt/…/java/Assignment/Q14/bin]
└─$                                                             28 ⚙
```

## Q15. WAP to demonstrate method overriding in hierarchical inheritance.

## Code:

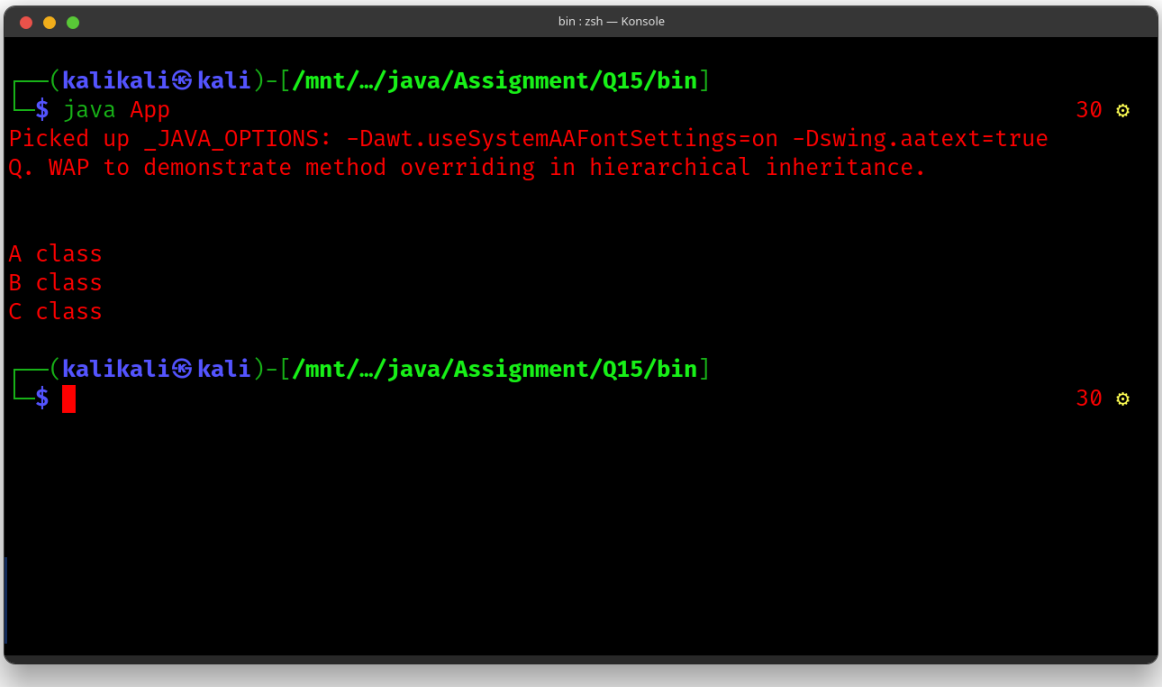
src : vim — Konsole

```
public class App {
    public static void main(String[] args) throws Exception {
        System.out.println("Q. WAP to demonstrate method overriding in hierarchical
inheritance.\n\n");
        A objA = new A();
        objA.printInfo();

        B objB = new B();
        objB.printInfo();


        C objC = new C();
        objC.printInfo();
    }
}
~
~
~
~
~
"App.java" 14L, 360B                                    1,1           All
```

src : vim — Konsole

```
public class Hierarchical {
    void printInfo(){
        System.out.println("Hierarchical class");
    }
}
class A extends Hierarchical{
    void printInfo(){
        System.out.println("A class");
    }
}
class B extends Hierarchical{
    void printInfo(){
        System.out.println("B class");
    }
}
class C extends Hierarchical{
    void printInfo(){
        System.out.println("C class");
    }
}
                                                        20,1          All
```

## Output:


bin : zsh — Konsole
┌──(kalikali㉿kali)-[/mnt/…/java/Assignment/Q15/bin]
└─$ java App
30
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Q. WAP to demonstrate method overriding in hierarchical inheritance.
A class
B class
C class
┌──(kalikali㉿kali)-[/mnt/…/java/Assignment/Q15/bin]
└─$
30