Fundamentals of Object Oriented Programming

CSN- 103

Dr. R. Balasubramanian

Associate Professor

Department of Computer Science and Engineering

Indian Institute of Technology Roorkee

Roorkee 247 667

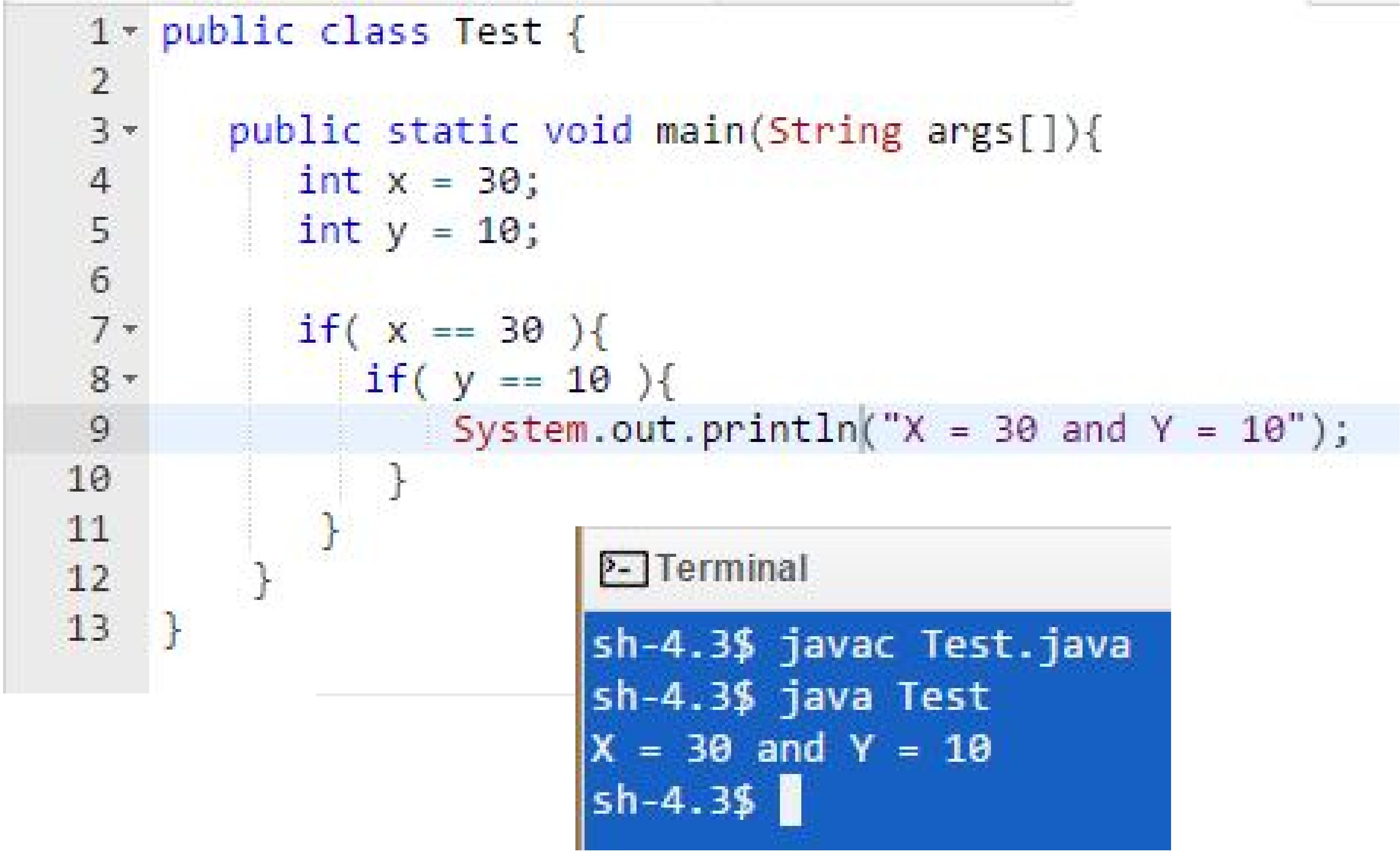
balarfcs@iitr.ac.in

https://sites.google.com/site/balaiitr/

Nested if...else Statement

- **Syntax**

```
if(Boolean_expression 1)
{
//Executes when the Boolean expression 1 is true
  if(Boolean_expression 2)
{ //Executes when the Boolean expression 2 is true }
}
```



```java
public class Test {

    public static void main(String args[]){
        int x = 30;
        int y = 10;

        if( x == 30 ){
            if( y == 10 ){
                System.out.println("X = 30 and Y = 10");
            }
        }
    }
}
```

```
sh-4.3$ javac Test.java
sh-4.3$ java Test
X = 30 and Y = 10
sh-4.3$ 
```

The switch Statement

- A switch statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.
- **Syntax**

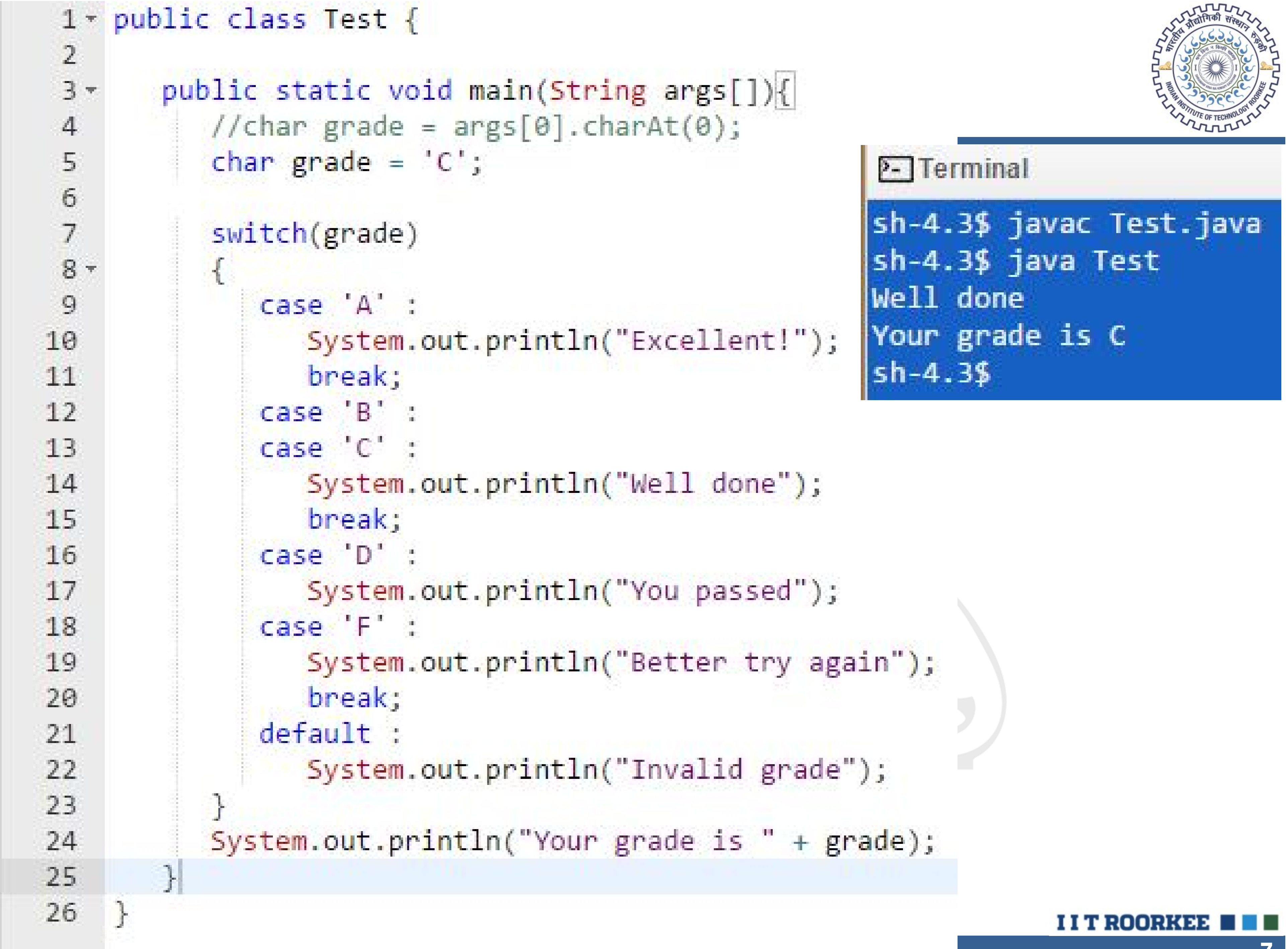
```
  switch(expression)
  { case value1 :
   //Statements
  break; //optional
   case value2 :
  //Statements
  break; //optional
  //You can have any number of case statements.
  default : //Optional
//Statements }
```

Rules

- The variable used in a switch statement can only be a byte, short, int, or char.
- You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.
- The value for a case must be the same data type as the variable in the switch and it must be a constant or a literal.
- When the variable being switched on is equal to a case, the statements following that case will execute until a *break* statement is reached.

Rules

- When a *break* statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
- Not every case needs to contain a break. If no break appears, the flow of control will *fall through* to subsequent cases until a break is reached.
- A *switch* statement can have an optional default case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No break is needed in the default case.



```java
public class Test {

    public static void main(String args[]){
        //char grade = args[0].charAt(0);
        char grade = 'C';

        switch(grade)
        {
            case 'A' :
                System.out.println("Excellent!");
                break;
            case 'B' :
            case 'C' :
                System.out.println("Well done");
                break;
            case 'D' :
                System.out.println("You passed");
            case 'F' :
                System.out.println("Better try again");
                break;
            default :
                System.out.println("Invalid grade");
        }
        System.out.println("Your grade is " + grade);
    }
}
```

Terminal

```
sh-4.3$ javac Test.java
sh-4.3$ java Test
Well done
Your grade is C
sh-4.3$
```

Decision Making and Looping

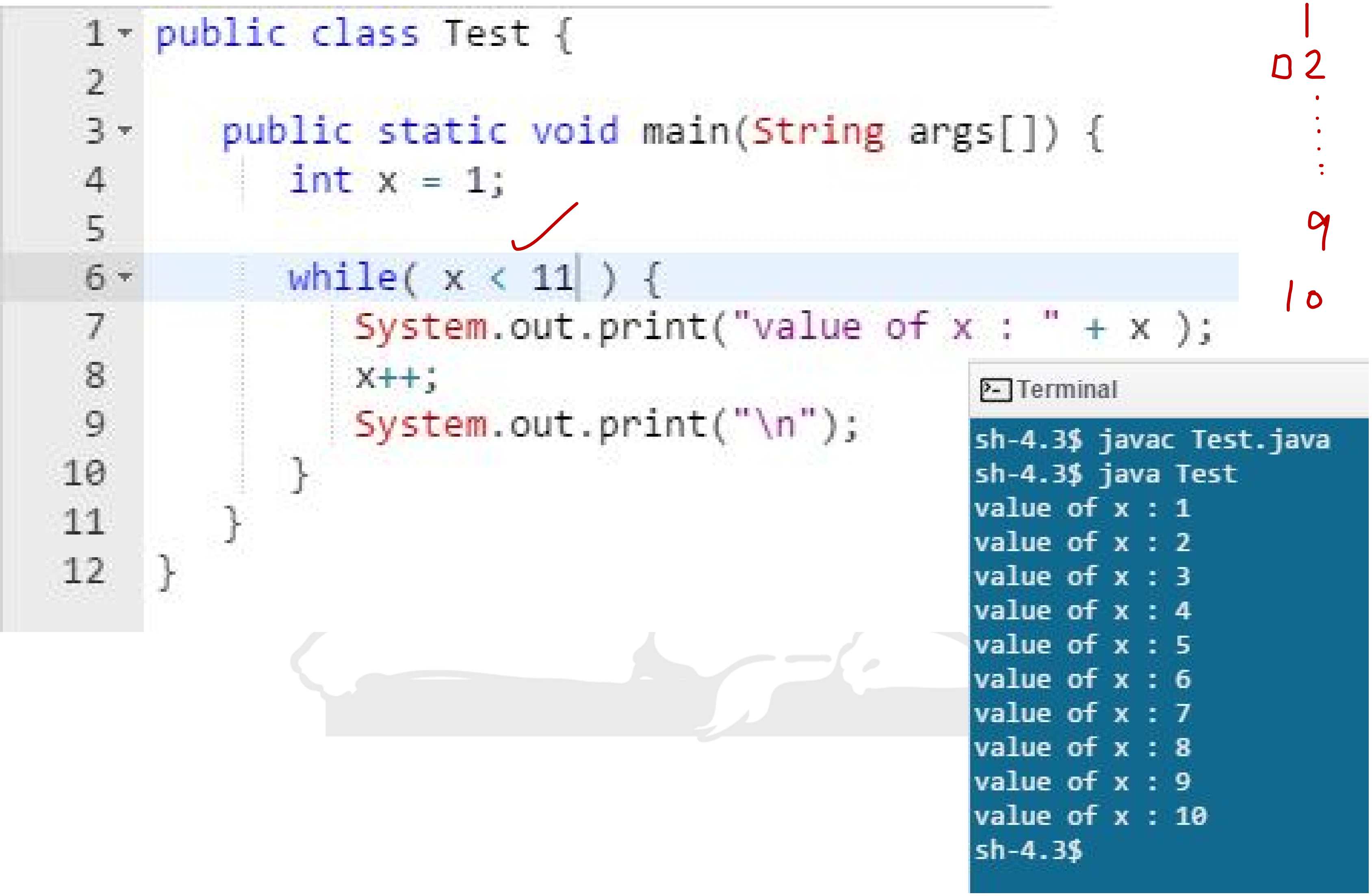
- The process of repeatedly executing a block of statements is known as looping
- The Java language provides for three constructs for performing loop operations
  - `while` construct
  - do construct
  - `for` construct

The while Loop

- A while loop is a control structure that allows you to repeat a task a certain number of times.
- **Syntax**

```
while(Boolean_expression)
{
   //Statements
}
```

The while Loop



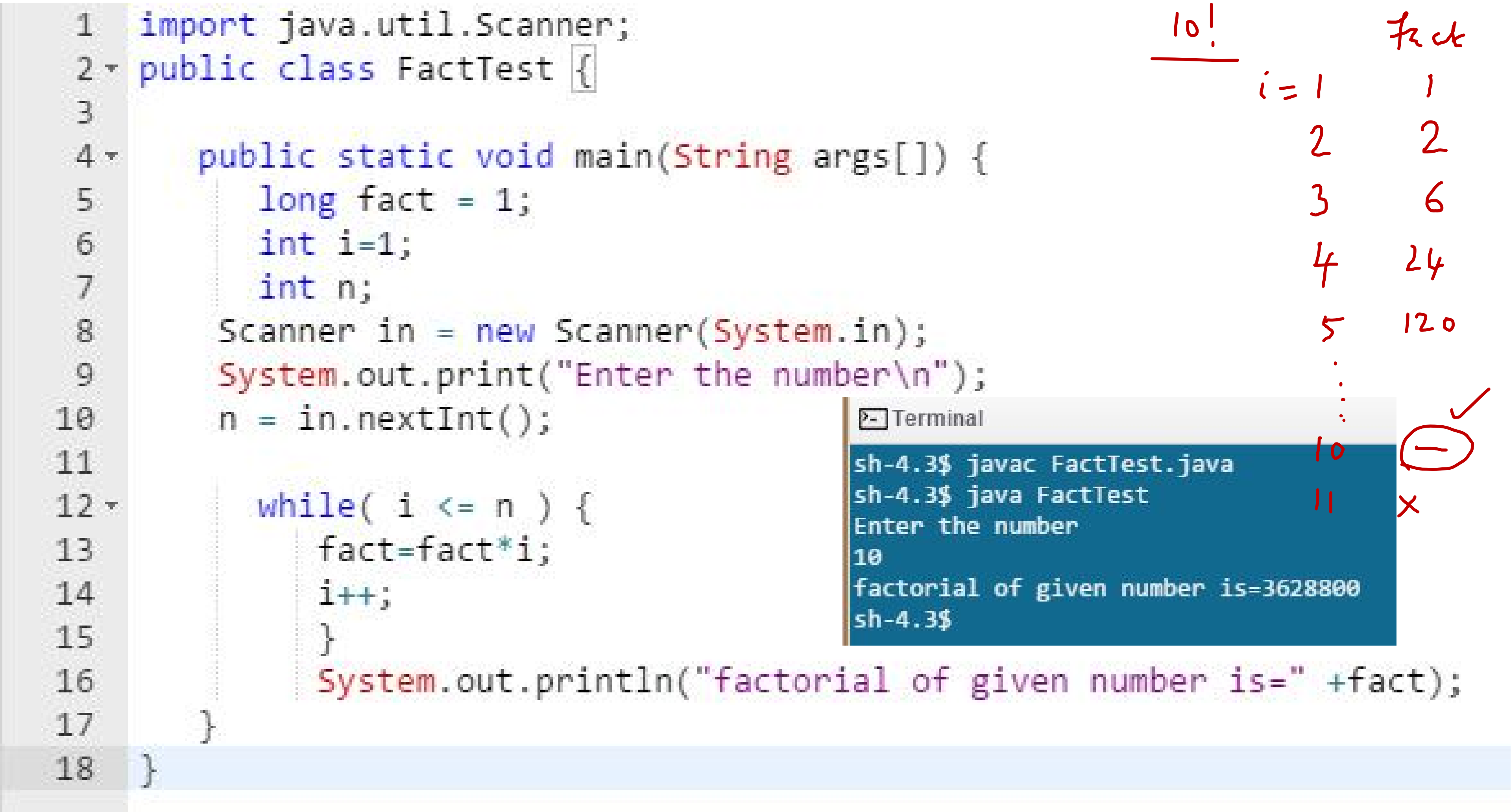
```java
public class Test {

    public static void main(String args[]) {
        int x = 1;

        while( x < 11 ) {
            System.out.print("value of x : " + x );
            x++;
            System.out.print("\n");
        }
    }
}
```

```
sh-4.3$ javac Test.java
sh-4.3$ java Test
value of x : 1
value of x : 2
value of x : 3
value of x : 4
value of x : 5
value of x : 6
value of x : 7
value of x : 8
value of x : 9
value of x : 10
sh-4.3$
```

✓

1
2
⋮
9
10

WAP to find the factorial of a given number



```java
import java.util.Scanner;
public class FactTest {

    public static void main(String args[]) {
        long fact = 1;
        int i=1;
        int n;
      Scanner in = new Scanner(System.in);
      System.out.print("Enter the number\n");
      n = in.nextInt();

        while( i <= n ) {
            fact=fact*i;
            i++;
            }
            System.out.println("factorial of given number is=" +fact);
    }
}
```

```
sh-4.3$ javac FactTest.java
sh-4.3$ java FactTest
Enter the number
10
factorial of given number is=3628800
sh-4.3$
```

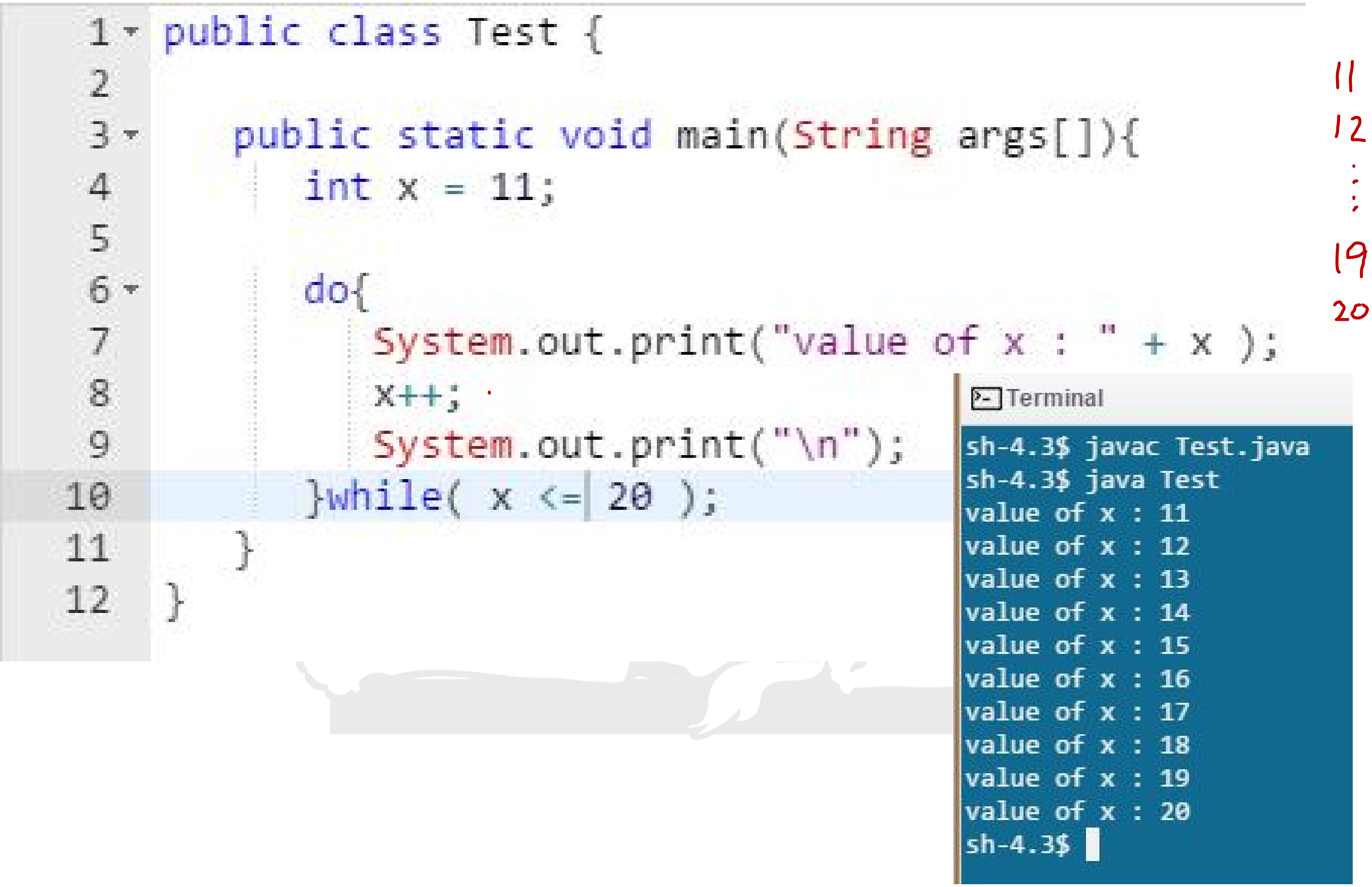
10!

	fact
i=1	1
2	2
3	6
4	24
5	120
⋮	
10	— ✓
11	×

The do...while Loop

- **Syntax**

```
do
{
  //Statements
}
while(Boolean_expression);
```



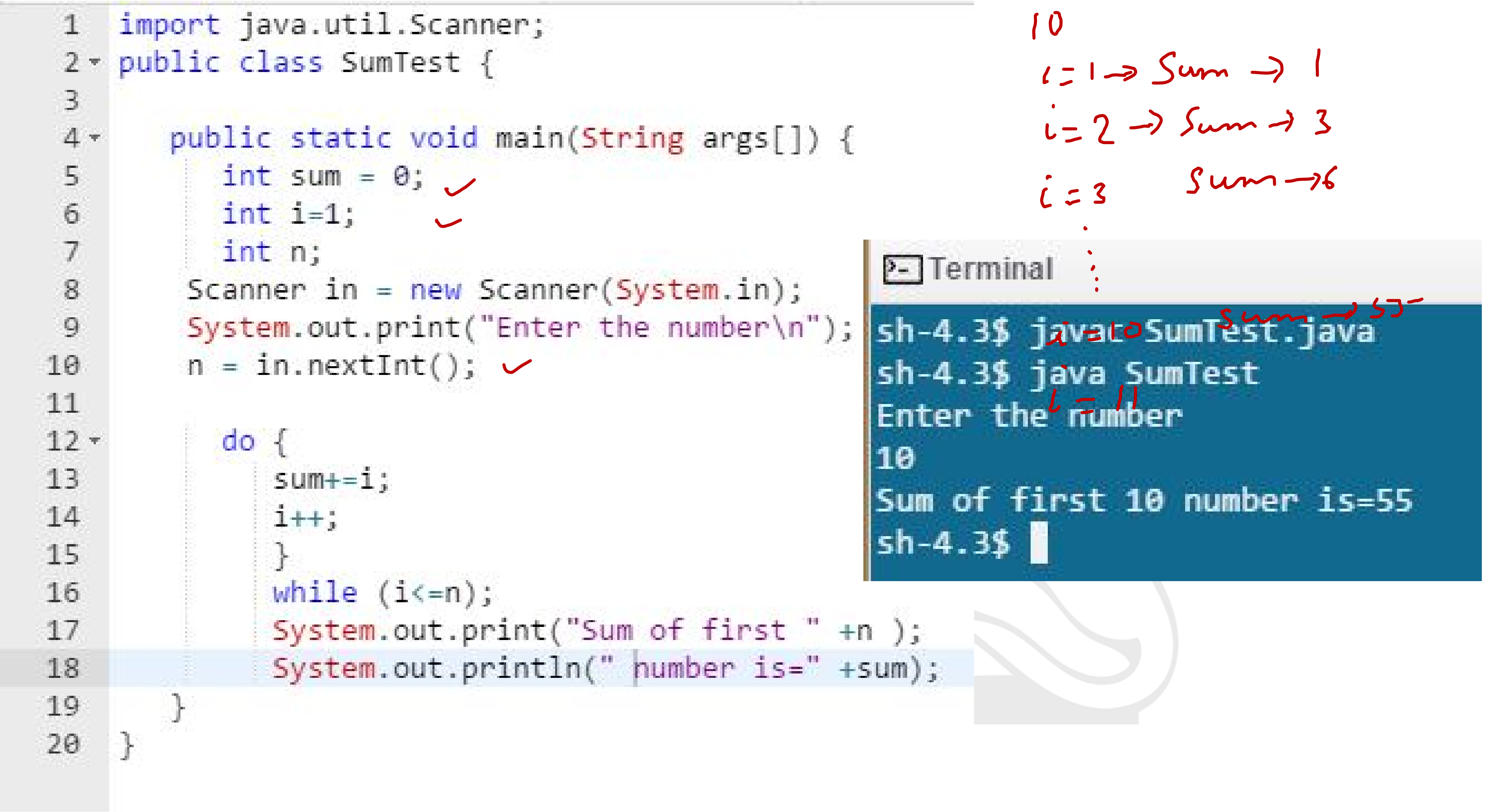
```java
public class Test {

    public static void main(String args[]){
        int x = 11;

        do{
            System.out.print("value of x : " + x );
            x++;
            System.out.print("\n");
        }while( x <= 20 );
    }
}
```

Terminal

```
sh-4.3$ javac Test.java
sh-4.3$ java Test
value of x : 11
value of x : 12
value of x : 13
value of x : 14
value of x : 15
value of x : 16
value of x : 17
value of x : 18
value of x : 19
value of x : 20
sh-4.3$
```

11
12
⋮
19
20

WAP to find the sum of a first *n* numbers



```java
import java.util.Scanner;
public class SumTest {

    public static void main(String args[]) {
        int sum = 0;
        int i=1;
        int n;
      Scanner in = new Scanner(System.in);
      System.out.print("Enter the number\n");
      n = in.nextInt();

        do {
            sum+=i;
            i++;
            }
            while (i<=n);
            System.out.print("Sum of first " +n );
            System.out.println(" number is=" +sum);
    }
}
```

```
sh-4.3$ javac SumTest.java
sh-4.3$ java SumTest
Enter the number
10
Sum of first 10 number is=55
sh-4.3$
```

10

i=1 → Sum → 1

i=2 → Sum → 3

i=3   Sum →6

⋮

i=10   Sum → 55

i=11

The for loop

int i;

for (i=0; i<100; i++)

- **Syntax**

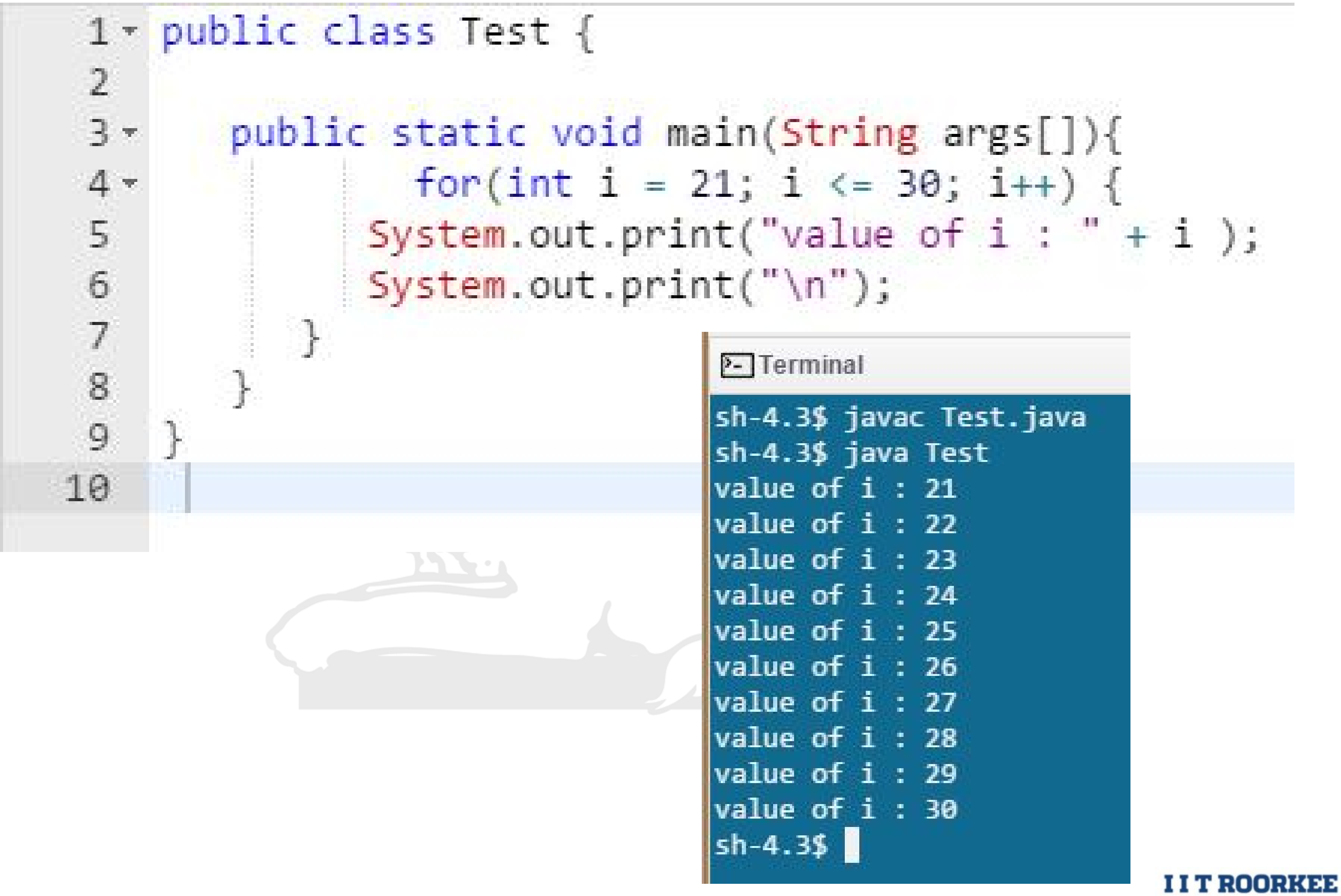
The syntax of a for loop is:

```
for(initialization; Boolean_expression; update)
{ //Statements }
```

The flow of control in a `for` loop

for(i=0,

- The initialization step is executed first, and only once. This step allows you to declare and initialize any loop control variables. You are not required to put a statement here, as long as a semicolon appears.
- Next, the Boolean expression is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and flow of control jumps to the next statement past the for loop.
- After the body of the for loop executes, the flow of control jumps back up to the update statement. This statement allows you to update any loop control variables. This statement can be left blank, as long as a semicolon appears after the Boolean expression.
- The Boolean expression is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then update step, then Boolean expression). After the Boolean expression is false, the for loop terminates.



```java
public class Test {

   public static void main(String args[]){
           for(int i = 21; i <= 30; i++) {
         System.out.print("value of i : " + i );
         System.out.print("\n");
      }
   }
}
```

```
sh-4.3$ javac Test.java
sh-4.3$ java Test
value of i : 21
value of i : 22
value of i : 23
value of i : 24
value of i : 25
value of i : 26
value of i : 27
value of i : 28
value of i : 29
value of i : 30
sh-4.3$
```