Chapter - 3

# Program Control

- Input characters from the keyword
- if statement
- Nested ifs
- if-else-if Ladder
- Switch Statement
- Nested switch statements
- for loop, Enhanced for loop
- while loop, do-while
- use continue
- Nested loops

## Input Characters from Keyboard

- Reading User Input
  - ✓Reading Character
  - ✓Reading other data type
  - ✓Using Scanner class

## Reading User input: Inputting Character

- Syntax:

  *System.in.read();*

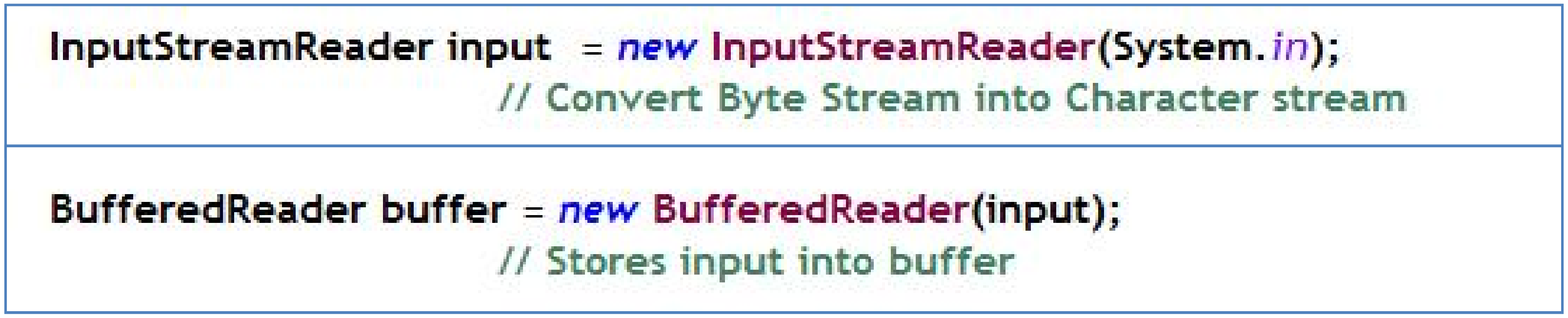
  – Waits till user supplies input character

  – Character is returned as number, Need to typecast to character

Example:

```
class  charInput {

      public static void main()   throws  java.io.Exception   {
        char ch;
        System.out.println("Input a character: ");
        ch = (char) System.in.read();
        System.out.println(" character is: " + ch);
      }
}
```

## 4. Reading User input: other Datatypes



```
InputStreamReader input  = new InputStreamReader(System.in);
                           // Convert Byte Stream into Character stream

BufferedReader buffer = new BufferedReader(input);
                           // Stores input into buffer
```

*Combining above 2 lines:*

```
BufferedReader  buffer = new BufferedReader ( new InputStreamReader ( System.in)  );
```

4. Reading User input: other Datatypes

## *Reading Integer:*

```
BufferedReader  buffer = new BufferedReader ( new InputStreamReader ( System.in)  );

        String  str = buffer.readLine() ;
        int a = Integer.parseInt ( str );
        float b = Float.parseFloat ( str );
```

## 4. Reading User input: other Datatypes

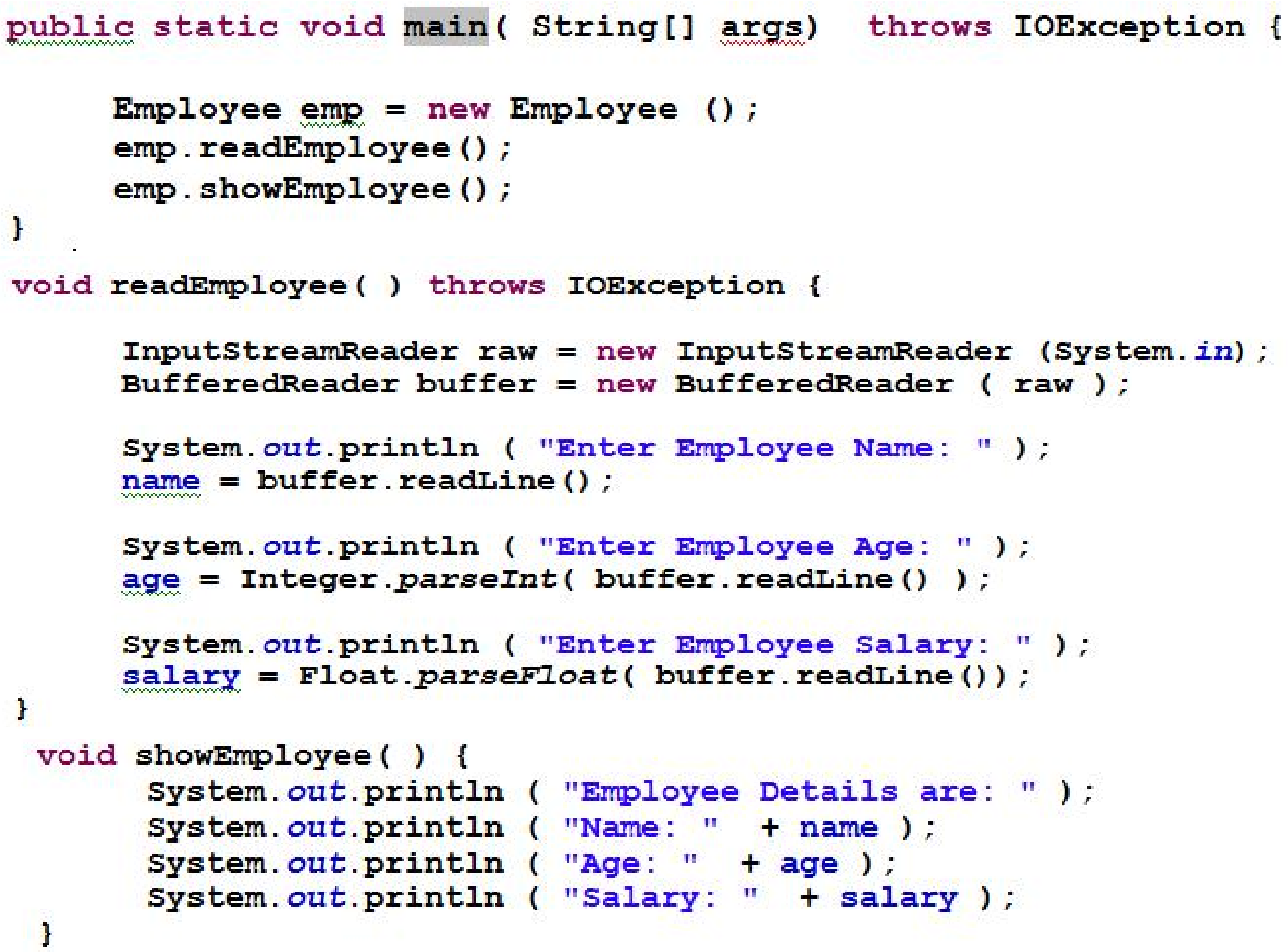
- *Write program to do the following*
  - *Create Class with name Employee*
  - *Method main()*
    - *to read name, age and Salary of employee.*
  - *Method showEmployee()*
    - *to show name, age and Salary of employee.*

Reading in main method itself

```
class   Employee {

      String    name;
      int       age;
      float     salary;

   public static void main( String[] args)   throws IOException {

         Employee emp  =  new   Employee ();
         InputStreamReader input  = new InputStreamReader(System.in);
         BufferedReader buffer = new BufferedReader(input);

         System.out.println ("Enter Employee Name  :");
         emp.name = buffer.readLine();

         System.out.println ("Enter Employee Age  :");
         emp.age = Integer.parseInt(buffer.readLine());

         System.out.println ("Enter Employee Salary  :");
         emp.salary = Float.parseFloat( buffer.readLine());

         emp.showEmployee();
   }

   void showEmployee( )  {
          System.out.println ( "Employee Details are:  " );
          System.out.println ( "Name:  "   + name );
          System.out.println ( "Age:  "   + age );
          System.out.println ( "Salary:  "   + salary );
   }
}  //end of class
```

## 4. Reading User input: other Datatypes

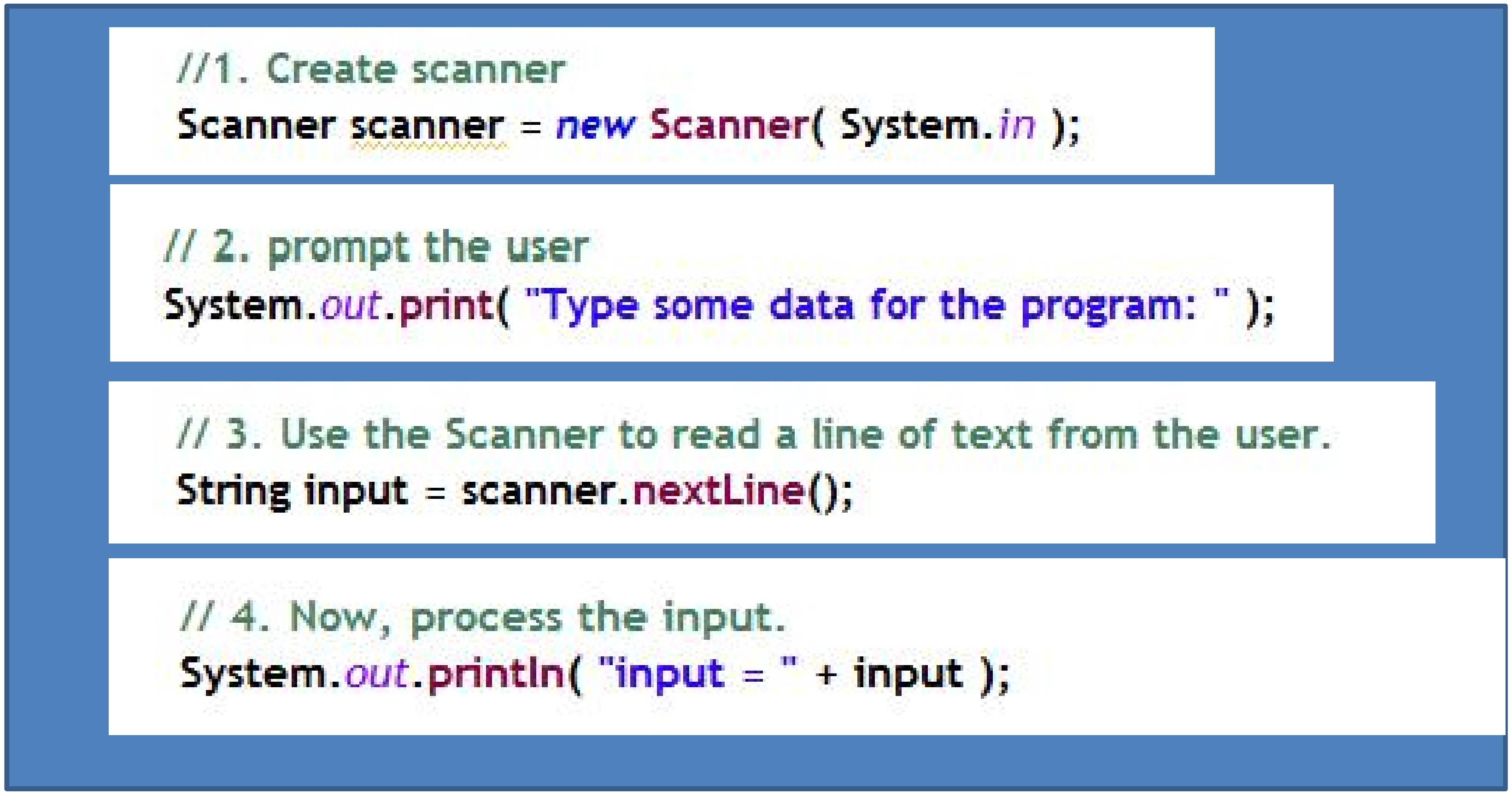
- *Modify the program to add*
  - *Method readEmployee()*
    - *to read name, age and Salary of employee.*
    - *Call it from main*



```
class   Employee {       String    name;
                 int         age;    Reading in main method itself
                 float       salary;

      public static void main( String[] args)   throws IOException {

            Employee emp = new Employee ();
            emp.readEmployee();
            emp.showEmployee();
      }

      void readEmployee( ) throws IOException {

            InputStreamReader raw = new InputStreamReader (System.in);
            BufferedReader buffer = new BufferedReader ( raw );

            System.out.println ( "Enter Employee Name:  " );
            name = buffer.readLine();

            System.out.println ( "Enter Employee Age:  " );
            age = Integer.parseInt( buffer.readLine() );

            System.out.println ( "Enter Employee Salary:  " );
            salary = Float.parseFloat( buffer.readLine());
      }
       void showEmployee( )  {
             System.out.println ( "Employee Details are:  " );
             System.out.println ( "Name:  "  + name );
             System.out.println ( "Age:  "  + age );
             System.out.println ( "Salary:  "  + salary );
       }
}  //end of class
```

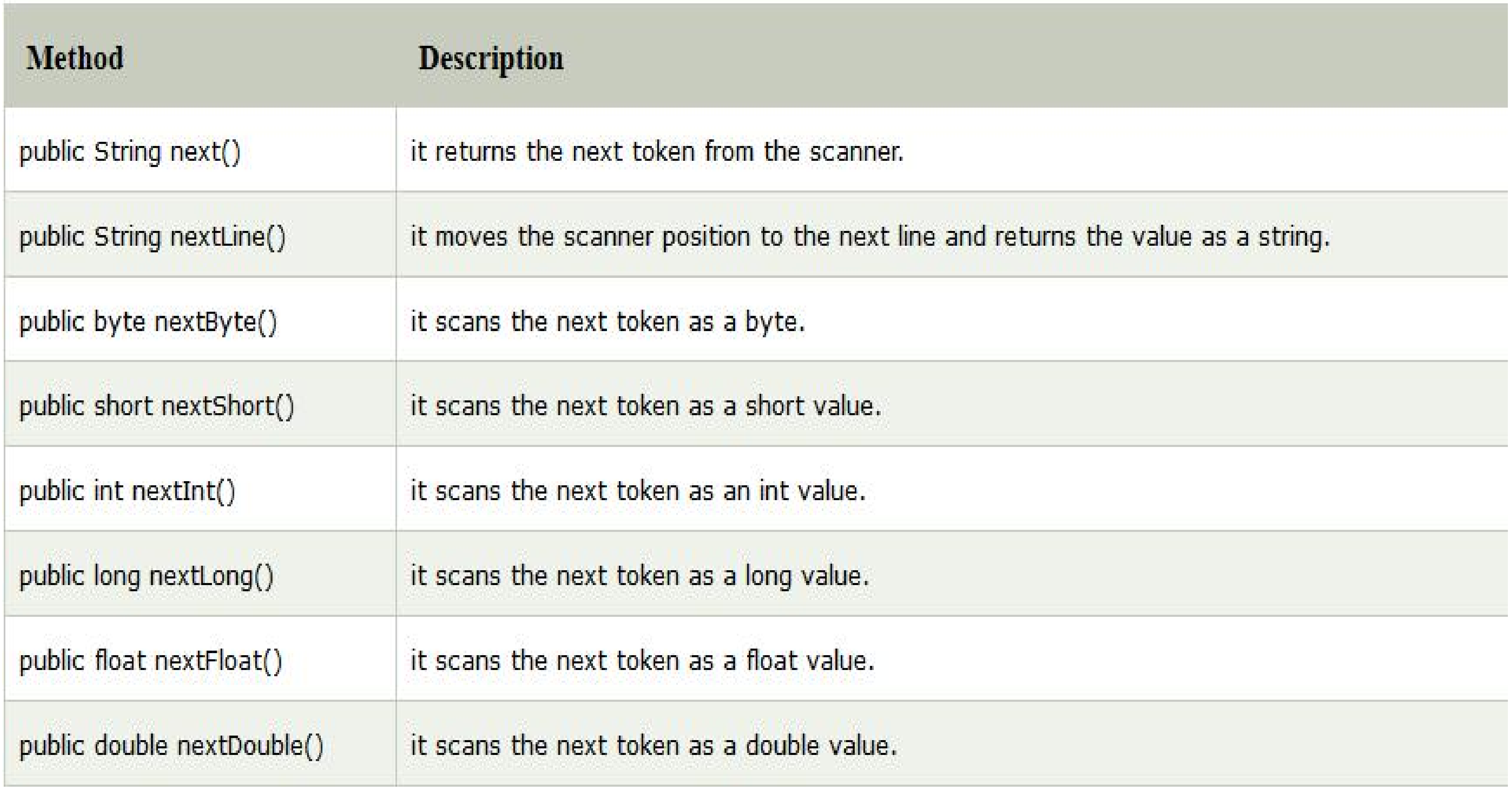
# 4. Using Scanner class

- Defined in Package
  - java.util.Scanner;



```java
//1. Create scanner
Scanner scanner = new Scanner( System.in );

// 2. prompt the user
System.out.print( "Type some data for the program: " );

// 3. Use the Scanner to read a line of text from the user.
String input = scanner.nextLine();

// 4. Now, process the input.
System.out.println( "input = " + input );
```

## Scanner class methods



| Method | Description |
|---|---|
| public String next() | it returns the next token from the scanner. |
| public String nextLine() | it moves the scanner position to the next line and returns the value as a string. |
| public byte nextByte() | it scans the next token as a byte. |
| public short nextShort() | it scans the next token as a short value. |
| public int nextInt() | it scans the next token as an int value. |
| public long nextLong() | it scans the next token as a long value. |
| public float nextFloat() | it scans the next token as a float value. |
| public double nextDouble() | it scans the next token as a double value. |

## 4. Reading User input: other Datatypes

- *Re Write program to do the following*
  - *Create Class with name Employee*
  - *Method readEmployee()*
    - *to read name, age and Salary of employee.*
  - *Method showEmployee()*
    - *to show name, age and Salary of employee.*

***Using Scanner class***

```
class   Employee {        String    name;
                 int          age;    Reading in main method itself
                 float        salary;

      public static void main( String[] args)   throws IOException {

            Employee emp = new Employee ();
            emp.readEmployee();
            emp.showEmployee();
      }

       void readEmployee( ) throws IOException{

             Scanner input = new Scanner (System.in)  ;

             System.out.println ( "Enter Employee Name:  " );
             name = input.nextLine();

             System.out.println ( "Enter Employee Age:  " );
             age = Integer.parseInt( input.nextLine() );

             System.out.println ( "Enter Employee Salary:  " );
             salary = Float.parseFloat( input.nextLine());
       }

       void showEmployee( )  {
             System.out.println ( "Employee Details are:  " );
             System.out.println ( "Name:  "  + name );
             System.out.println ( "Age:  "  + age );
             System.out.println ( "Salary:  "  + salary );
       }
}  //end of class
```

## Control Statements

- If statement
- Nested if
- If-else-ladder
- Switch
- Nested switch
- for loop
- for-each
- while loop
- do-while loop

## 2. Control Statements

### if statement

- Syntax: *single statement*

```
if(condition)   statement;
else   statement;
```

- Syntax: *Multiple statement*

```
if(condition)
{
    statement list;
}
else
{
    statement list;
}
```

Control Statements

## Nested If statement

Syntax: *single statement*

```
if(condition)
  if(condition)
        statement;
```

Syntax: ***Multiple statement***

```
if (condition)
{
        if (condition)
        {
                statement list;
        }
        else
        {
                statement list;
        }
}
```

## Else-if ladder

```
if(condition)
        statement;
else if(condition)
        statement;
else if(condition)
        statement;
            .
            .
            .
else
        statement;
```

# Control Statements

## Switch statement

```
switch(expression)
{
    case  constant-1 :
            statement list;
            break;
    case  constant-2 :
            statement list;
            break;
       .
       .
       .
    default :
            statement list;
}
```

- Prior to JDK 7,
  - (expression) in switch must be of type byte, short, int, char only
- After JDK 7,
  - (expression) can be String type
- default is executed If no case constant matches the value of (expression)

# Control Statements

## Nested Switch

*case constant of inner and outer switch can contain common values. ( ex: case 'A')*

```
switch ( myVal )
 {
        case  'A':
                 ..................;

        switch ( myVal )
        {
                case  'A':
                         ..................;
                         break;
                case  'B' :
                         statement list;
                          //...........
        }
break;
        case  'B' :
                 //...........
}
```

# Control Statements

**for loop**

```
for ( initialization ; condition ; iteration)
      single-line-statement ;
```

```
for ( initialization ; condition ; iteration)
{
      Multi-line-statements ;
}
```

**Example**

```
for(  int m = 0 ;  m < 3  ; m++ )
      System.out.println ( m );
```

## Control Statements

**For-each loop**

```
for ( type var : array)
    for-statements ;
```

```
for ( type var : collection)
    for-statements ;
```

**Example**

```
int  num[] =  { 10, 20, 30 };

for ( int   a : num)
   System.out.println ( " " + a ) ;
```

- used to access each successive value in a collection of values
- commonly used to iterate over an array or a Collections class
- JDK 5 onwards

## Control Statements

**While loop**

```
while( condition)
    statement ;
```

```
while( condition)
{
    statement-list ;
}
```

**Do-while loop**

```
do
    statement ;
while( condition)
```

```
do
{
    statement ;
}
while( condition);
```

## Print Pattern

| Pattern 1 | | Pattern 2 |
|---|---|---|
| 1<br>1 2<br>1 2 3<br>1 2 3 4<br>Slide 24 | *<br>* *<br>* * *<br>* * * * | 1<br>2 2<br>3 3 3<br>4 4 4 4<br>@ |
| **Pattern 3** | **Pattern 4 (Flyod's)** | |
|    *<br>  * *<br> * * *<br>* * * * | 1<br>2 3<br>4 5 6<br>7 8 9 10 | |

```
public class MainClass
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);

        System.out.println("How many rows you want in this pattern?");

        int rows = sc.nextInt();

        System.out.println("Here is your pattern....!!!");

        for (int i = 1; i <= rows; i++)
        {
            for (int j = 1; j <= i; j++)
            {
                System.out.print( j +" ");
            }
            System.out.println();
        }
        sc.close();
    }
}
```

```
1
1 2
1 2 3
1 2 3 4
```

@

```
for (int i = 1; i <= rows; i++)
      {
         for (int j = 1; j <= i; j++)
         {
            System.out.print( i +" ");
         }

         System.out.println( );
      }
```

```
1
2 2
3 3 3
4 4 4 4
```

```
int rowCount = 1;

for (int i = noOfRows; i > 0; i--)
{
    for (int j = 1; j <= i; j++)              // for spaces to print
      {
                System.out.print("  ");
      }

  //Printing 'rowCount' value 'rowCount' times at the end of each row

   for (int j = 1; j <= rowCount; j++)
                {
                 System.out.print(rowCount+" ");
                 }

   System.out.println();

   rowCount++;
}
```

## Floyds

```
for (int i = 1; i <= noOfRows; i++)
    {
        for (int j = 1; j <= i; j++)
        {
            System.out.print(value+"\t");

            value++;
        }

        System.out.println();
    }
```

```
1
2 3
4 5 6
7 8 9 10
```

## Jump Statements

- break
- *continue*
- *return*

## Jump Statements

**`Uses of break`**

- Using break to *exit* loop
- Using break as *jump*

## Using break to exit for

***- With simple for loop***

```
for(initialization ; condition ; iteration)
{
      //statement_list;
      if ( condition)
            break ;
}
```

```
for( i=0 ; i<7 ; i++)
{
      System.out.println(  i );
      if ( i==4 )
            break ;
}
```

*output:*

```
      0, 1, 2, 3, 4
```

## using break to exit for loop

**- With nested for loop**

```
for( i=1 ; i<4 ; i++)
{
    for( j=1 ; j<4 ; j++)
    {
            System.out.println (" inner loop" );
            if( j==2 )
              break ;
    }
    System.out.println (" Outer loop" );
}
```

**Output:**

```
inner loop        //  i= 1, j= 1
inner loop        //  i= 1, j= 2
Outer loop

inner loop //  i= 2, j= 1
inner loop //  i= 2, j= 2
Outer loop

inner loop //  i= 3, j= 1
inner loop //  i= 3, j= 2
Outer loop
```

3. Use of beak as jump – with nested for

```
for( i=1 ; i<=3 ; i++)
{
   one: {
          two: {
                 three: {       System.out.println ("for i= " + i );
                                if( i==1 )
                                    break one;

                                if( i==2 )
                                    break two;

                                if( i==3 )
                                   break three;
                        } System.out.println ("  End Three" );

                 } System.out.println ("  End Two" );

          } System.out.println ("  End One" );

 } // end for
```

Output:

```
 for  i = 1

      End One

 for  i = 2

      End Two
      End One

 for  i = 3

      End Three
      End Two
      End One
```

# Jump Statements

## Uses of continue

The Java *continue statement* is used to continue loop. It continues the current flow of the program and skips the remaining code at specified condition

```
for(int i=1;i<=10;i++) {
    if(i==5) {
        continue;
      }
System.out.println( i );
}
}
```

***Output:***

```
1 2 3 4 6 7 8 9 10
```

# Jump Statements

## Uses of return

```
int x;
System.out.println(“Before return”);
If(x==1) return;
System.out.println(“After return”);
```