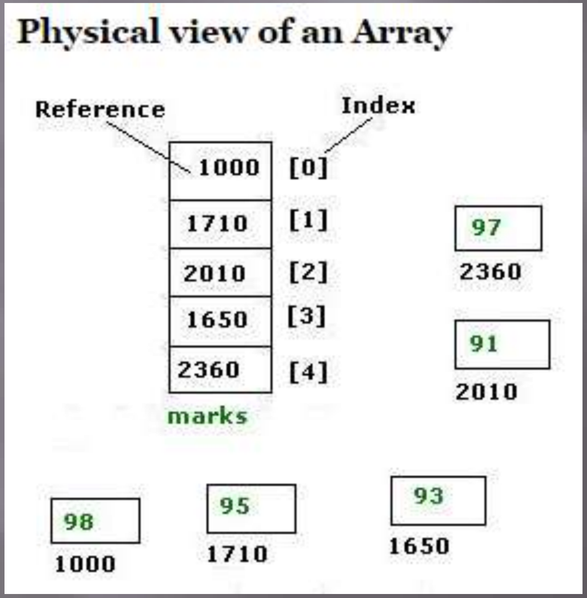
JAVA ARRAYS AND CONTROL STATEMENTS

Java Arrays

- In java arrays are objects. All methods of an Object can be invoked on an array. Arrays are stored in heap memory.



- **Array Declaration**

//Single Dimensional Array

int[] arr; *//recommended*

int arr[];

//Multi Dimensional Array

int[][] arr; *//recommended*

int arr[][];

int[] arr[];

- **Array Instantiation**

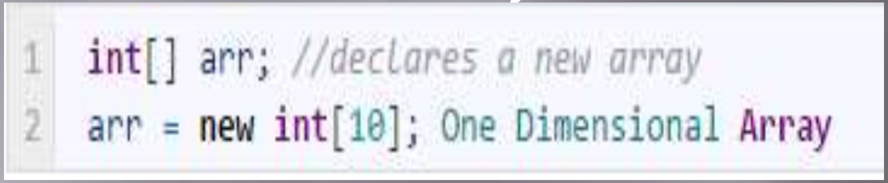
marks = new int[5];

- **Java array initialization and instantiation together**

int marks[] = {98, 95, 91, 93, 97};

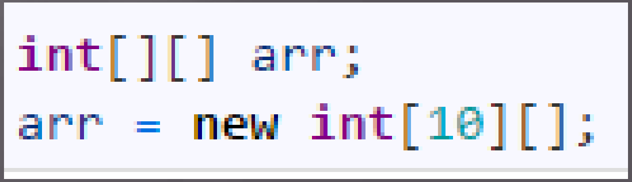
Constructing an Java Array

- **One Dimensional Array**
- New keyword will be used to construct one/multi dimensional array.



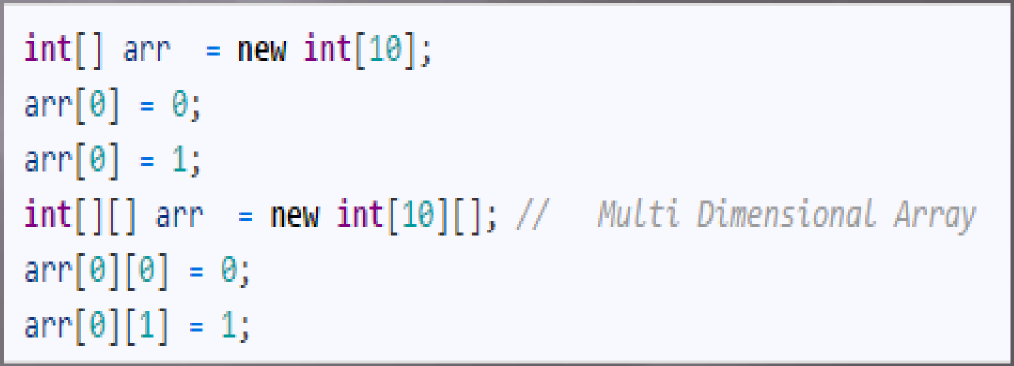
```
int[] arr; //declares a new array
arr = new int[10]; One Dimensional Array
```

- **Two Dimensional Array**
- These are array of arrays. So a two dimensional array is array of arrays of int



```
int[][] arr;
arr = new int[10][];
```

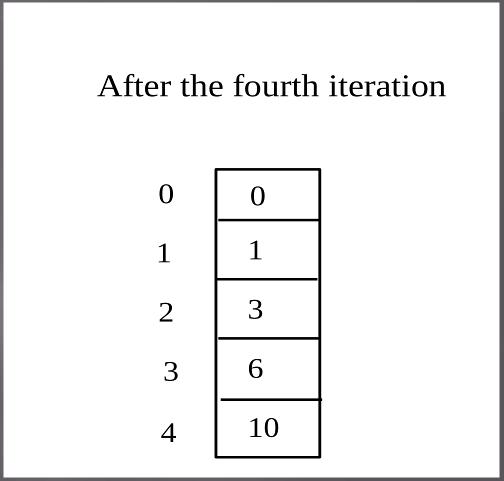
- **Initializing Array**



```
int[] arr  = new int[10];
arr[0] = 0;
arr[0] = 1;
int[][] arr  = new int[10][]; //   Multi Dimensional Array
arr[0][0] = 0;
arr[0][1] = 1;
```

Iterating a Java Array

```
public class Test {
  public static void main(String[] args) {
    int[] values = new int[5];
    for (int i = 1; i < 5; i++) {
      values[i] = i + values[i-1];
    }
    values[0] = values[1] + values[4];
  }
}
```



Initializing arrays with input values

```
myList.length=5;
java.util.Scanner input = new java.util.Scanner(System.in);
System.out.print("Enter " + myList.length + "
    values: ");
for (int i = 0; i < myList.length; i++)
 myList[i] = input.nextDouble();
```

- Initializing arrays with random values

```
for (int i = 0; i < myList.length; i++)
{
 myList[i] = Math.random() * 100;
}
```

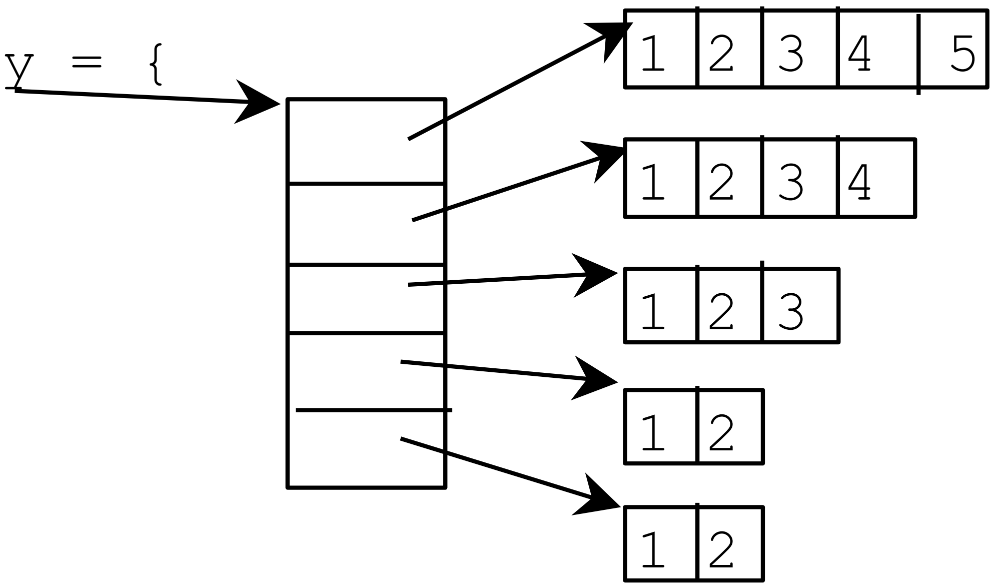
Declare/Create Two-dimensional Arrays

```
// Declare array ref var
dataType[][] refVar;

// Create array and assign its reference to variable
refVar = new dataType[10][10];

// Combine declaration and creation in one statement
dataType[][] refVar = new dataType[10][10];

// Alternative syntax
dataType refVar[][] = new dataType[10][10];
```

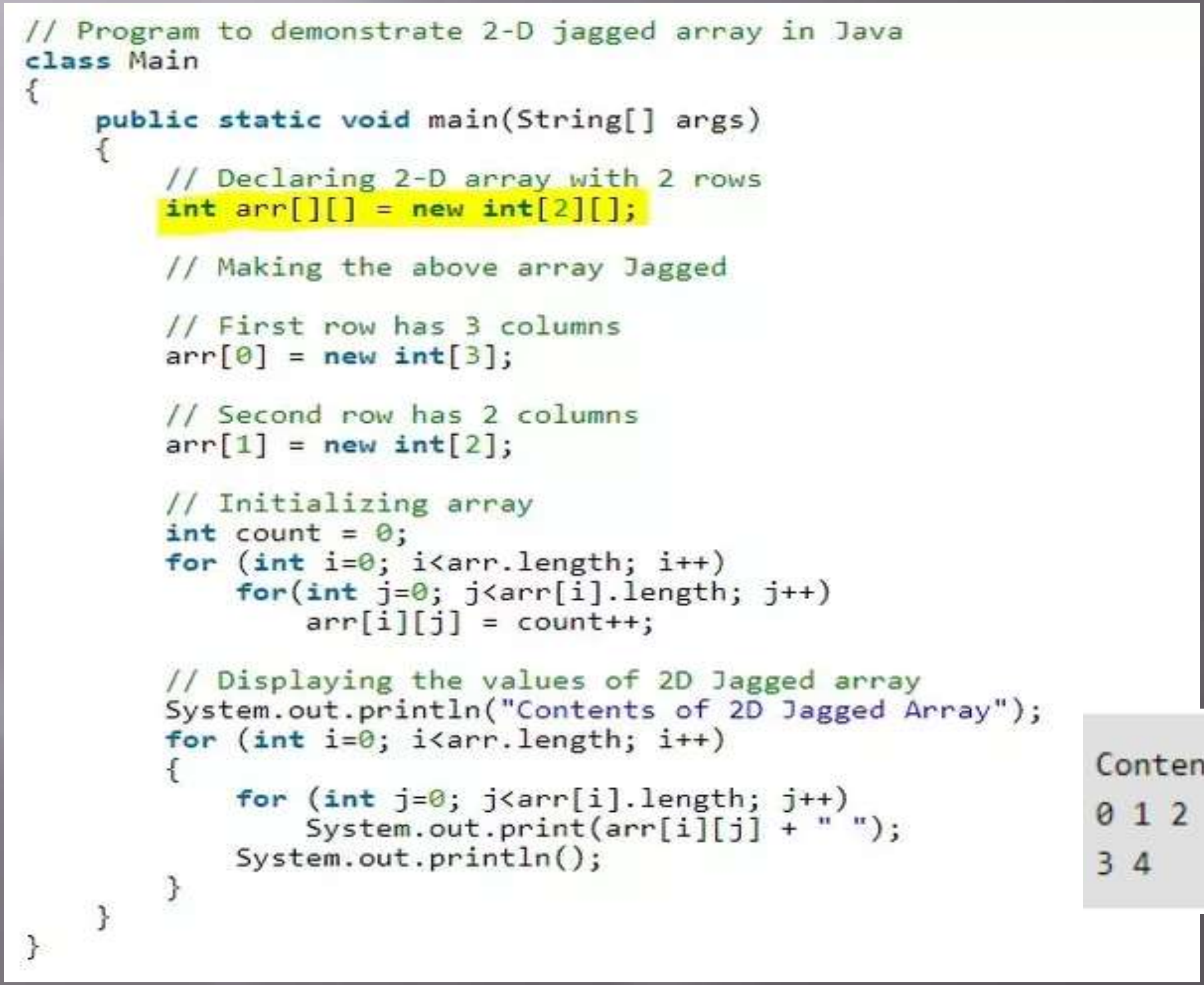
Ragged Arrays

Each row in a two-dimensional array is itself an array. So, the rows can have different lengths. Such an array is known as *a ragged array*. For example,

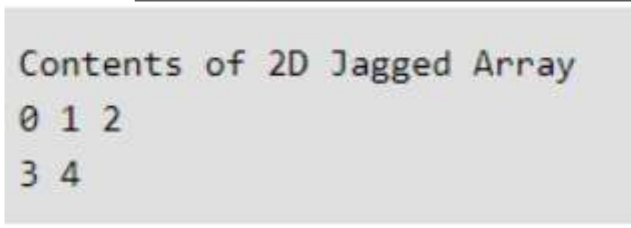
```
int[][] triangleArray = {
  {1, 2, 3, 4, 5},
  {2, 3, 4, 5},
  {3, 4, 5},
  {4, 5},
  {5}
};
```



Example



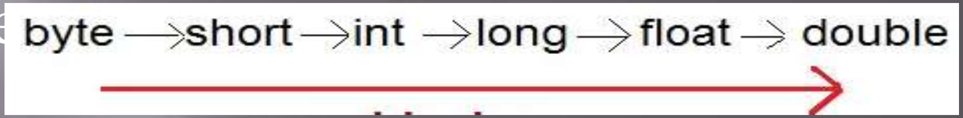
```
// Program to demonstrate 2-D jagged array in Java
class Main
{
    public static void main(String[] args)
    {
        // Declaring 2-D array with 2 rows
        int arr[][] = new int[2][];

        // Making the above array Jagged

        // First row has 3 columns
        arr[0] = new int[3];

        // Second row has 2 columns
        arr[1] = new int[2];

        // Initializing array
        int count = 0;
        for (int i=0; i<arr.length; i++)
            for(int j=0; j<arr[i].length; j++)
                arr[i][j] = count++;

        // Displaying the values of 2D Jagged array
        System.out.println("Contents of 2D Jagged Array");
        for (int i=0; i<arr.length; i++)
        {
            for (int j=0; j<arr[i].length; j++)
                System.out.print(arr[i][j] + " ");
            System.out.println();
        }
    }
}
```

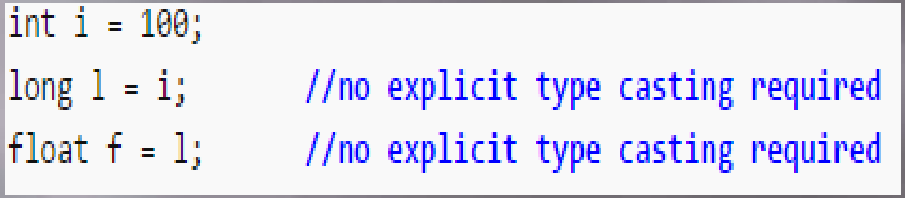


```
Contents of 2D Jagged Array
0 1 2
3 4
```

Type Casting

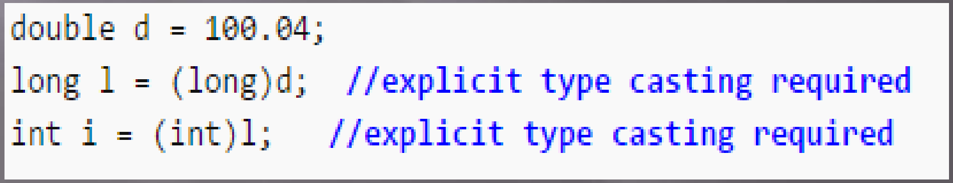
- Assigning a value of one type to a variable of another type is known as **Type Casting**.
- **Automatic Type casting (Implicit)** take place when, the two types are compatible and the target type is larger than the source





```
int i = 100;
long l = i;       //no explicit type casting required
float f = l;      //no explicit type casting required
```

- **Explicit type casting:** When you are assigning a larger type value to a variable of smaller type, then you need to perform explicit type casting.



```
double d = 100.04;
long l = (long)d;  //explicit type casting required
int i = (int)l;   //explicit type casting required
```

Flow control

Basically, it is exactly like c/c++.

if/else

```
If(x==4) {
   // act1
} else {
   // act2
}
```

do/while

```
int i=5;
do {
   // act1
   i--;
} while(i!=0);
```

for

```
int j;
for(int i=0;i<=9;i++)
{
   j+=i;
}
```

switch

```
char
c=IN.getChar();
switch(c) {
   case 'a':
   case 'b':
     // act1
    break;
   default:
     // act2
}
```

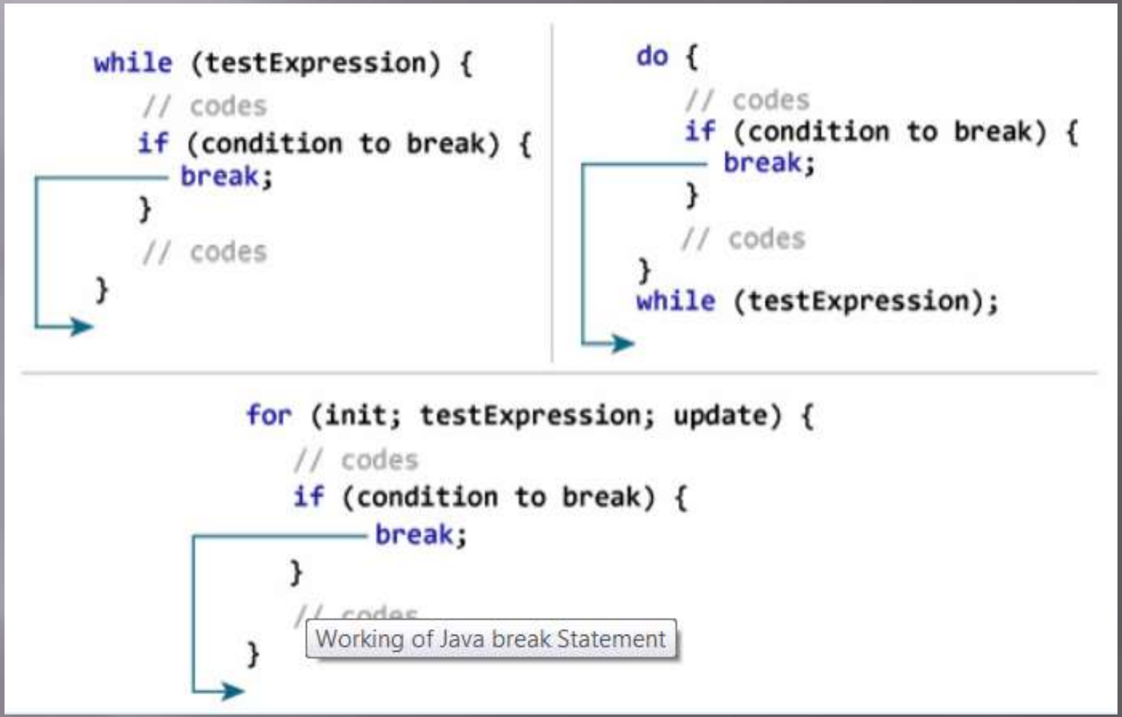
Flow control contd…

- break is used in the loops and when executed, the control of the execution will come out of the loop.
- continue makes the loop to skip the current execution and continues with the next iteration.
- return statement can be used to cause execution to branch back to the caller of the method.
- Labeled break and continue statements will break or continue from the loop that is mentioned. Used in nested loops.

switch Statements

```
switch (status) {
  case 0:  compute taxes for single filers;
           break;
  case 1:  compute taxes for married file jointly;
           break;
  case 2:  compute taxes for married file separately;
           break;
  case 3:  compute taxes for head of household;
           break;
  default: System.out.println("Errors: invalid status");
           System.exit(0);
}
```

Break statement



```
while (testExpression) {
   // codes
   if (condition to break) {
      break;
   }
   // codes
}
```

```
do {
   // codes
   if (condition to break) {
      break;
   }
   // codes
}
while (testExpression);
```

```
for (init; testExpression; update) {
   // codes
   if (condition to break) {
      break;
   }
   // codes
}
```

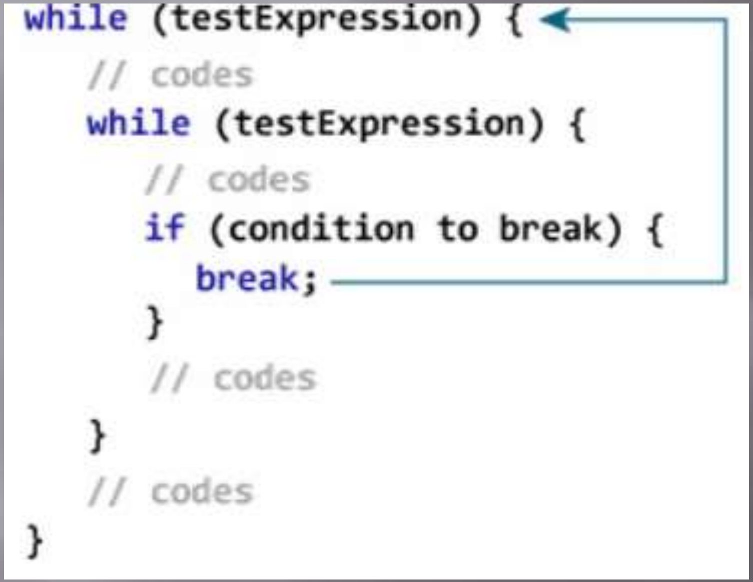
Working of Java break Statement

Example

```java
class UserInputSum {
    public static void main(String[] args) {

        Double number, sum = 0.0;

        // create an object of Scanner
        Scanner input = new Scanner(System.in);

        while (true) {
            System.out.print("Enter a number: ");

            // takes double input from user
            number = input.nextDouble();

            // if number is negative the loop terminates
            if (number < 0.0) {
                break;
            }

           sum += number;
        }
        System.out.println("Sum = " + sum);
    }
}
```

```
Enter a number: 3.2
Enter a number: 5
Enter a number: 2.3
Enter a number: 0
Enter a number: -4.5
Sum = 10.5
```

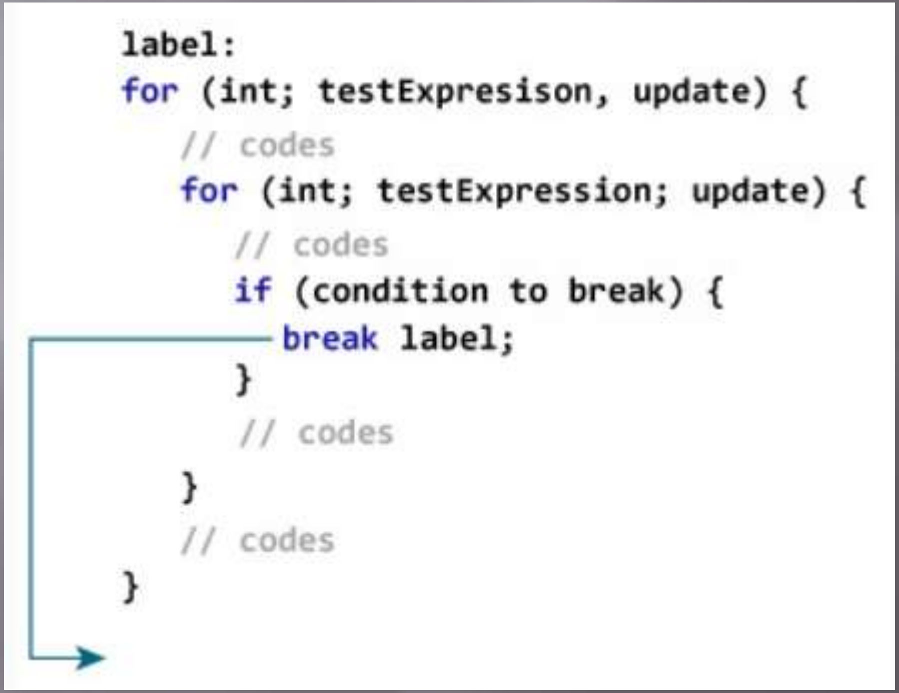
Java break and Nested Loop



```
while (testExpression) {
   // codes
   while (testExpression) {
      // codes
      if (condition to break) {
         break;
      }
      // codes
   }
   // codes
}
```

Labeled break Statement

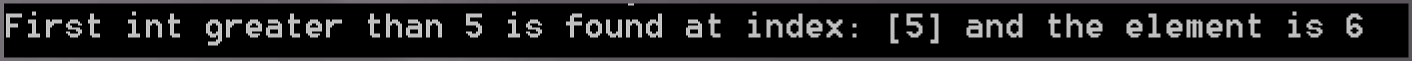
- We can use the labeled break statement to terminate the outermost loop.



```
label:
for (int; testExpresison, update) {
    // codes
    for (int; testExpression; update) {
        // codes
        if (condition to break) {
            break label;
        }
        // codes
    }
    // codes
}
```

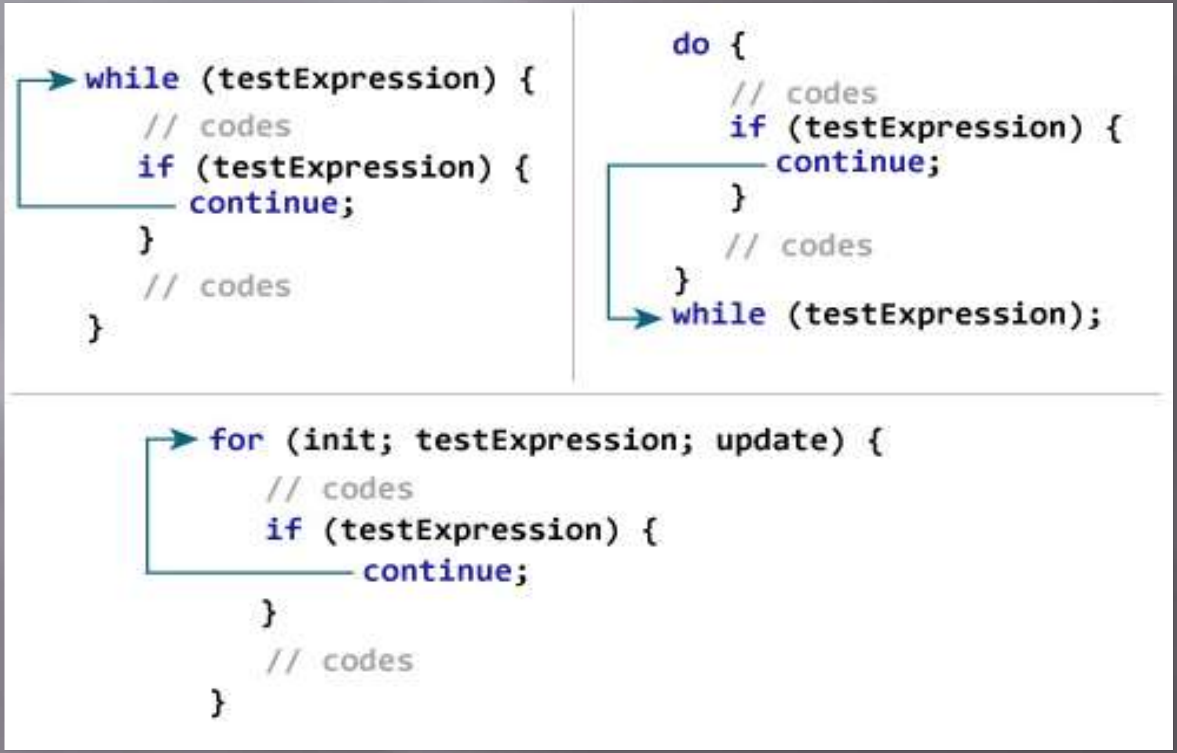
Java break statement with label example

```
public class JavaBreakLabel {
public static void main(String[] args) {
    int[]arr = { 1,2,3,4,5,6,7,6,8,9,10 };
    boolean found = false;
    int row = 0;
    // find index of first int greater than 5
    searchint:
    for (row = 0; row < arr.length; row++) {
                    if (arr[row] > 5) {
                            found = true;
                            // using break label to terminate outer statements
                            break searchint;
                    }
    }
    if (found)
  System.out.println("First int greater than 5 is found at index: [" + row + "] and the
element is "+arr[row]);
}
}
```



```
First int greater than 5 is found at index: [5] and the element is 6
```

Java continue statement

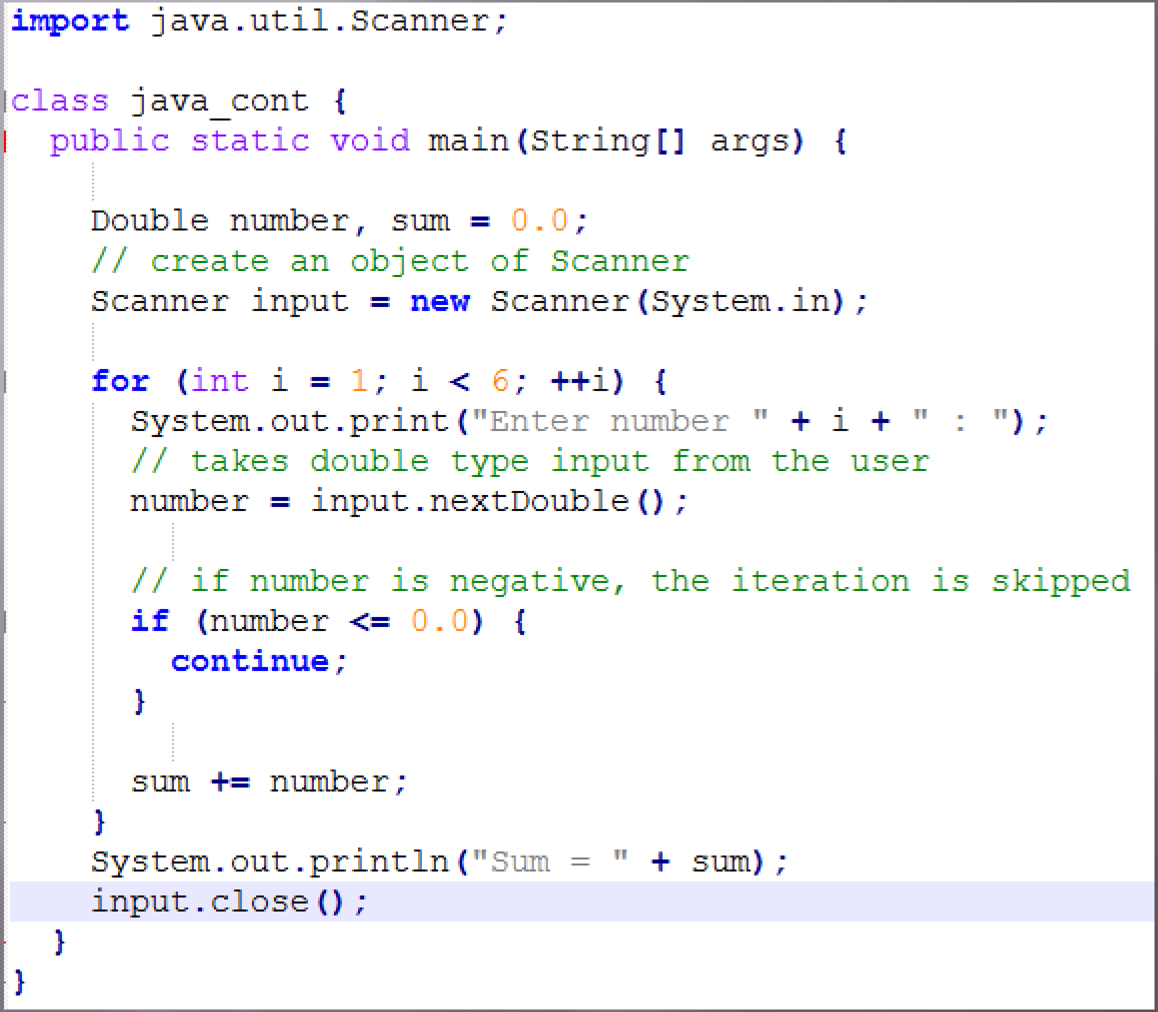


```
while (testExpression) {
    // codes
    if (testExpression) {
      continue;
    }
    // codes
}
```

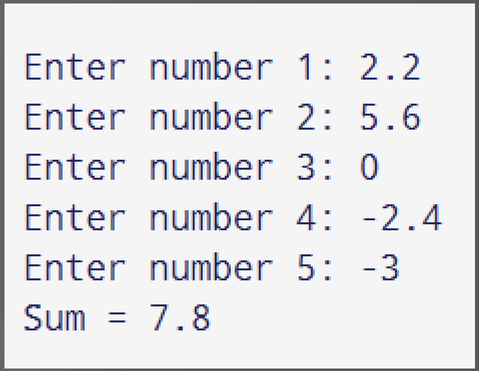
```
do {
   // codes
   if (testExpression) {
     continue;
   }
   // codes
}
while (testExpression);
```

```
for (init; testExpression; update) {
    // codes
    if (testExpression) {
      continue;
    }
    // codes
}
```

Example

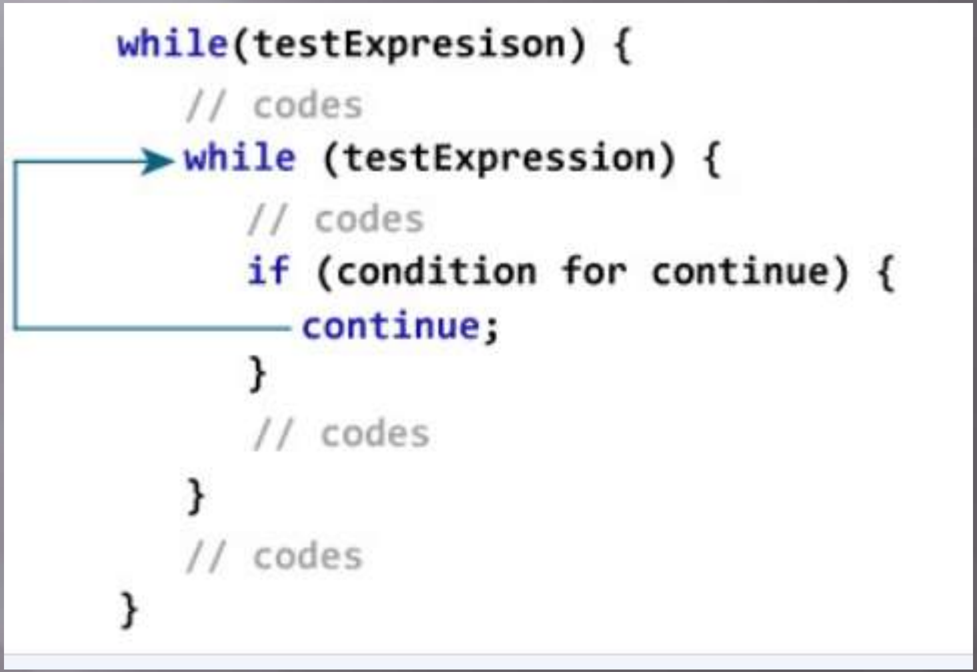


```java
import java.util.Scanner;

class java_cont {
  public static void main(String[] args) {

    Double number, sum = 0.0;
    // create an object of Scanner
    Scanner input = new Scanner(System.in);

    for (int i = 1; i < 6; ++i) {
      System.out.print("Enter number " + i + " : ");
      // takes double type input from the user
      number = input.nextDouble();

      // if number is negative, the iteration is skipped
      if (number <= 0.0) {
        continue;
      }

      sum += number;
    }
    System.out.println("Sum = " + sum);
    input.close();
  }
}
```



```
Enter number 1: 2.2
Enter number 2: 5.6
Enter number 3: 0
Enter number 4: -2.4
Enter number 5: -3
Sum = 7.8
```

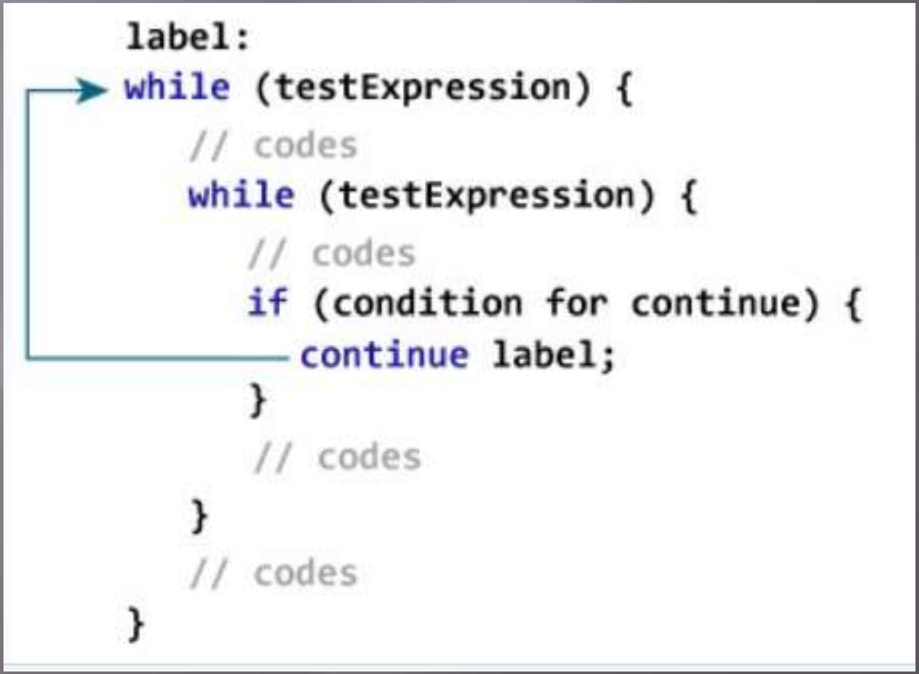
Java continue and Nested Loop



```
while(testExpresison) {
   // codes
   while (testExpression) {
      // codes
      if (condition for continue) {
        continue;
      }
      // codes
   }
   // codes
}
```

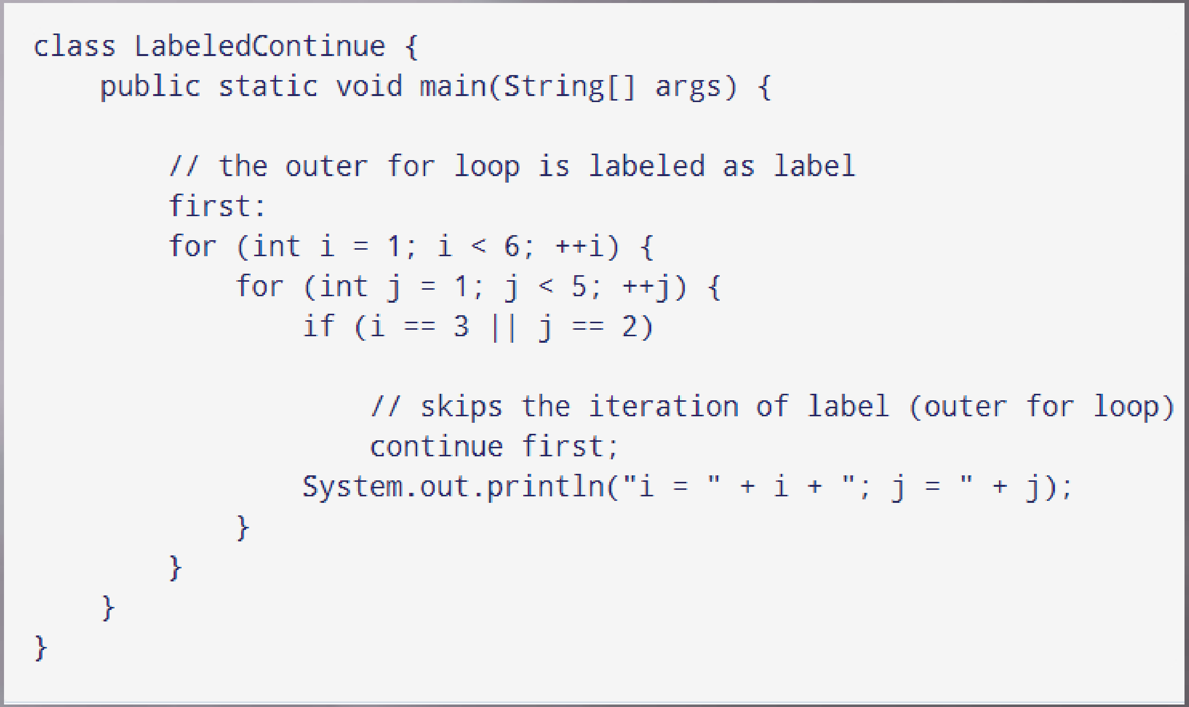
Labeled continue Statement

- It is used to terminate the innermost loop and switch statement. However, there is another form of continue statement in Java known as labeled continue.

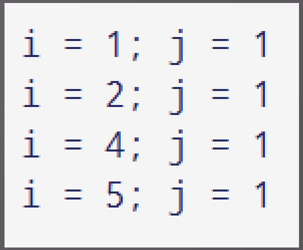


```
label:
while (testExpression) {
   // codes
   while (testExpression) {
      // codes
      if (condition for continue) {
         continue label;
      }
      // codes
   }
   // codes
}
```

Example



```java
class LabeledContinue {
    public static void main(String[] args) {

        // the outer for loop is labeled as label
        first:
        for (int i = 1; i < 6; ++i) {
            for (int j = 1; j < 5; ++j) {
                if (i == 3 || j == 2)

                    // skips the iteration of label (outer for loop)
                    continue first;
                System.out.println("i = " + i + "; j = " + j);
            }
        }
    }
}
```



```
i = 1; j = 1
i = 2; j = 1
i = 4; j = 1
i = 5; j = 1
```

Exercises-1

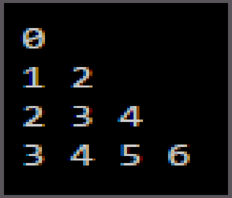
- Write a program that obtains hours and minutes from seconds using java operators
- Write a java program that takes your first name and last name as command line arguments to the program and displays your name and last name on separate lines.
- Write a program print odd and even numbers in the following format using continue – break – label

```
0  1
2  3
4  5
6  7
8  9
```

- Write a program to print factorial of a number using loops.
- Write a program that prompts the user to enter an integer from console. If the number is a multiple of 5, print HiFive. If the number is divisible by 2, print HiEven

Exercises on arrays

- Set up an array to hold the following values, and in this order: 23, 6, 47, 35, 2, 14. Write a Java program to print out the highest number in the array.
- Write a Java Program To Print Sum Of Upper Triangular Matrix
- Write a Java Program to sort the numbers using bubble sort.
- Write a Java program to print the numbers in the following format using Ragged array



```
0
1 2
2 3 4
3 4 5 6
```