<u>Decision making and looping</u>

- the process of repeatedly executing a block of code is known as looping.
- any loop consists of of two parts (1) body of the loop (2) control statement

eg:-

```
while (<expression>){ // test condition
        <statement 1>;  // body
        <statement 2>;  // body
        ….
        x++; // control statement
        }
```

- depending on the position of the test condition, the structure may be classified into two groups
  1. entry control: the condition is tested before the start of the loop execution
  2. exit control: the condition is tested at the end of the loop execution

(1) WHILE statement

- entry control statement

General form:

```
Initialization;
while (test condition){
        body of the loop;
        }
```

eg:-

```
int sum = 0, num = 1;
while (num <= 10){
        sum += num * num;
        num ++;
        }
```

(2) DO statement

- exit control statement
- body of the loop get executed for the first time regardless of the test condition

General form:

```
Initialization;
do{
        body of the loop;
        }
while (test condition);
```

eg:-

```
int sum = 0, num = 1;
do{
        sum += num*num;
        num ++;
        }
while (num <= 10);
```

(3) FOR statement
entry control statement

General form:

```
        for (<initialization>;<test condition>;<control>){
                body of the loop;
                }
```

eg:-

```
int sum =0;
for (int num = 0; num < 10; num++){
        sum = num*num;
        }
```

- the variable num is local to the loop, outside the loop it cannot be used

Note:-

1. can use an expression for initialization, test condition and control.

eg:-

```
int x;
for (x=(10+20)/2; x>0; x = x/2){

        }
```

eg:-

```
int x, sum = 10;
for (x=0; x<10 && sum <100; x = x/2){

        }
```

2. one or more sections can be omitted

eg:-

```
int m = 5;
for (; m != 10;){
        System.out.println("test");
        m++;
        }
```

<u>Nesting of FOR loops</u>

- having a for loop within another for loop

```
for (int i = 0; i<10; i++){
        for (int j = 0; j<10; j++){
                System.out.println("test");
                }
        }
```

(4) Jump in loops

- in some situations it is necessary to exit from the loop regardless of the test condition

eg:-
```
outer: for (int i = 0; i<10; i++){
        inner: for (int j = 0; j<10; j++){
                System.out.println(j);
                if (j==5)
                        break outer; //break the outer loop
                        /* break; //this will just break the current loop, that is inner loop*/
                }
        }
```

- outer and inner are just labels (you can use any names)
- break is used to jump out from the loop

<u>continue is used to skip a part of the code</u>

eg:-
```
for (int i=0; i<10; i++){
        if (i==5)
                continue;
        System.out.println(i);
        }
```

try the output of the above code

<u>Labeled loops</u>

- must be a valid identifier
- we have used labeled loops in the above example
- label must be a valid identifier
- place it before the loop, followed by colon (:)

```
        eg:-
                loop1: for (….){
                                …
                                }
```

following examples show the role of break and continue in a labeled loop

eg:-
```
loop1: for (int m=1;m<11;m++){
        loop2: for (int n=1;n<11;n++){
                        System.out.println("test");
                        if (m==5)
                        break loop1;
                        }
                }
```

eg:-

```
outer: for (int m=1;m<11;m++){
        for (int n=1;n<11;n++){
                System.out.println(“test”);
                if (m==n)
                        continue outer;
                }
        }
```