# Algoritma dan Pemrograman
## Leon Andretti Abdillah

09

Control Flow – Loop – For

- Structure control, consists of:
  1. Sequence
  2. Selection/Choice/Decision
     a) if..else
     b) switch..case
  3. Loop/Iteration/Repetition
     a) while
     b) do..while
     c) for..

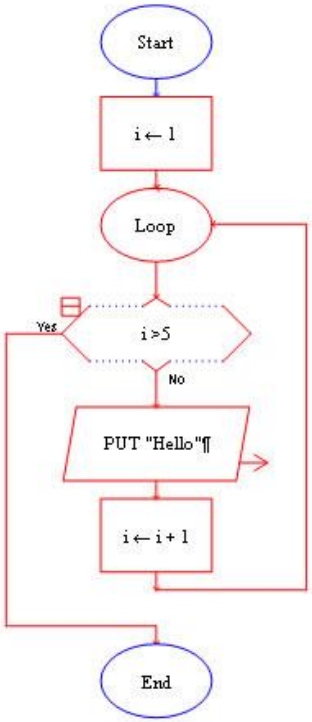
- The for statement provides a compact way to iterate over a range of values. Programmers often refer to it as the "for loop" because of the way in which it repeatedly loops until a particular condition is satisfied.
- A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.
- A for loop is useful when you know how many times a task is to be repeated.
- The for loop is a looping construct which can execute a set of instructions a specified number of times. It's a counter controlled loop.

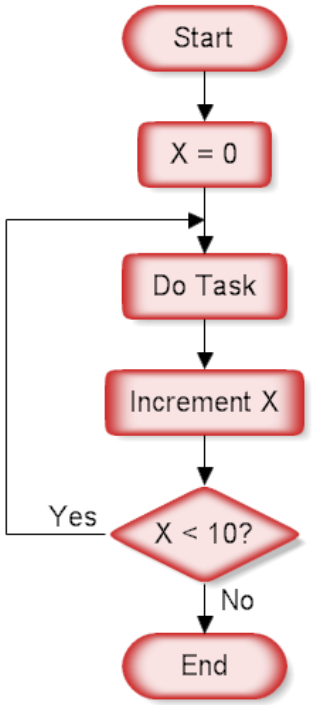
# Syntax

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

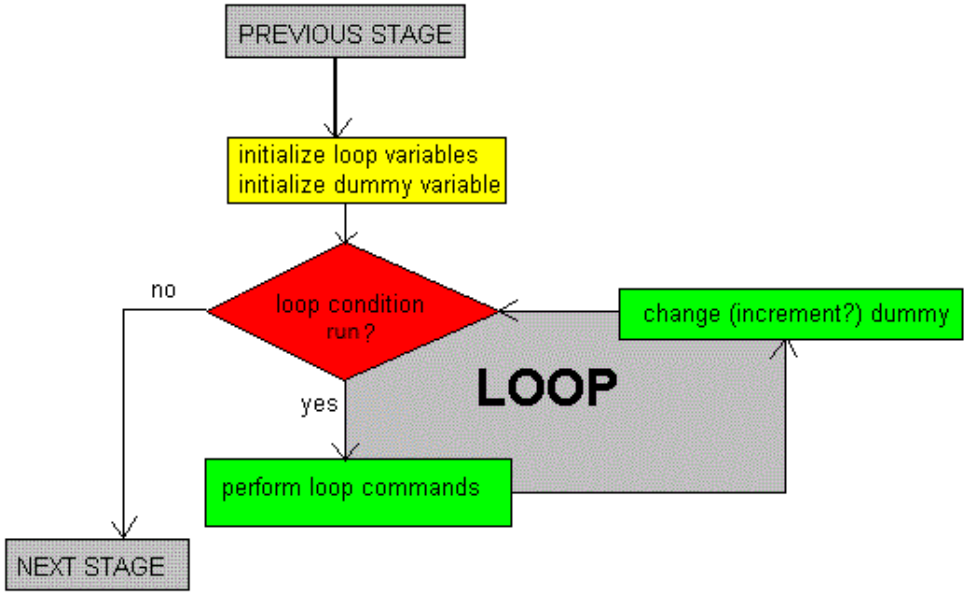
Or

```
for (initialization_expression ; loop_condition ; increment_expression) {
     // statements
}
```



# Comparison for and while



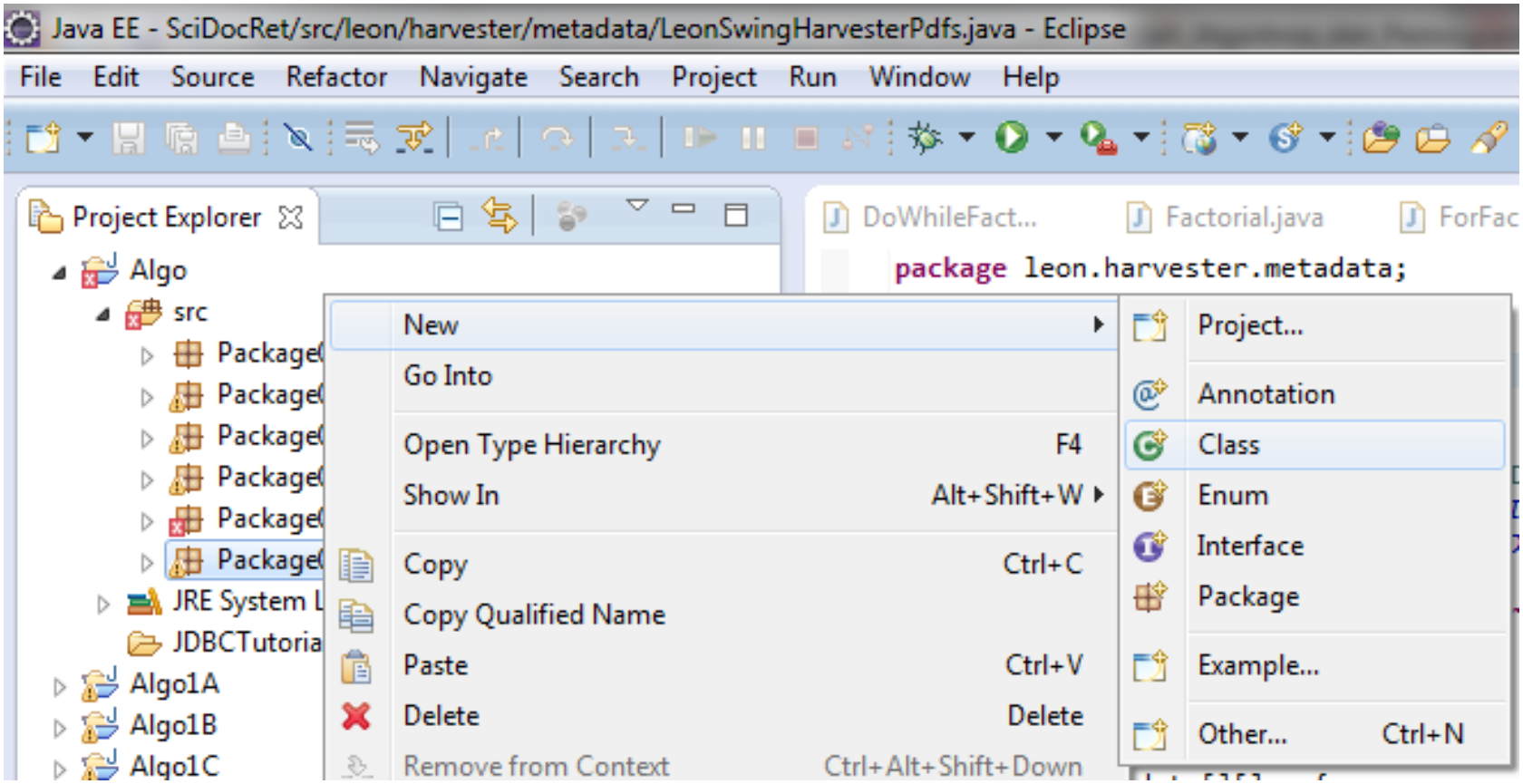

```
for (cnt = 0; cnt < N; cnt++)
 {
  inner code here
 }
```

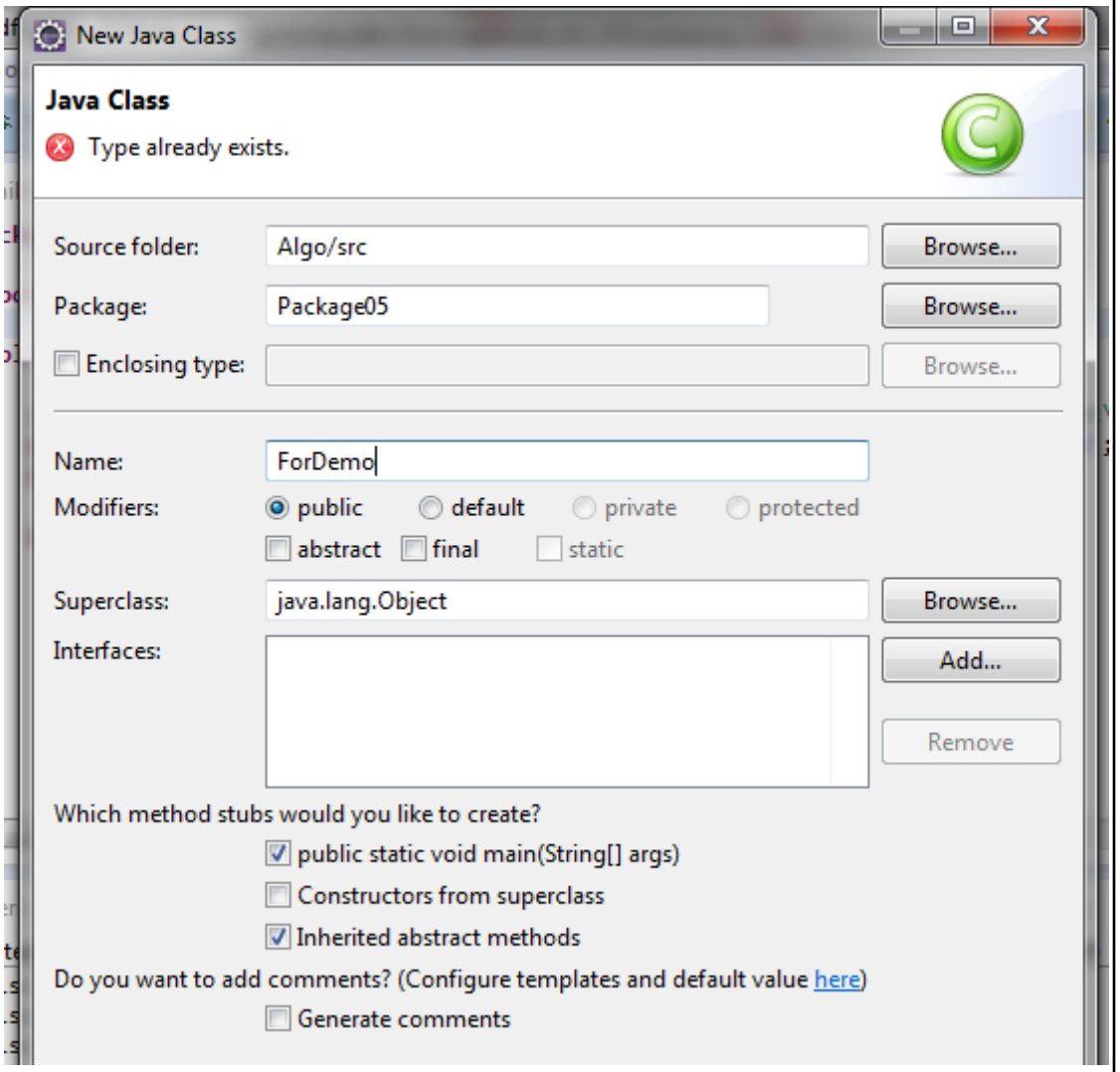
```
cnt = 0
while (cnt < N) {
   inner code here
   cnt++
  }
```

- Here is the flow of control in a for loop:
  - 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.

- Notes:
  - init is an initialization that will be performed before the first iteration.
  - booleanExpression is a boolean expression which will cause the execution of statement(s) if it evaluates to true.
  - update is a statement that will be executed after the execution of the statement block.
  - init, expression, and update are optional.

- The for statement will stop only if one of the following conditions is met:
  - booleanExpression evaluates to false
  - A break or continue statement is executed
  - A runtime error occurs.


Java EE - SciDocRet/src/leon/harvester/metadata/LeonSwingHarvesterPdfs.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Algo
src
JRE System L
JDBCTutoria
Algo1A
Algo1B
Algo1C
DoWhileFact...
Factorial.java
ForFac
package leon.harvester.metadata;
New
Go Into
Open Type Hierarchy F4
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Project...
Annotation
Class
Enum
Interface
Package
Example...
Other... Ctrl+N


New Java Class
Java Class
Type already exists.
Source folder:
Algo/src
Browse...
Package:
Package05
Browse...
Enclosing type:
Browse...
Name:
ForDemo
Modifiers:
public
default
private
protected
abstract
final
static
Superclass:
java.lang.Object
Browse...
Interfaces:
Add...
Remove
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments

```java
package Package05;

public class ForDemo {

    public static void main(String[] args){

        for(int i=1; i<=10; i++){
            System.out.println("Count is: " + i);
        }
    }
}
```

```
<terminated> ForDemo [Java Application] (
Count is: 1
Count is: 2
Count is: 3
Count is: 4
Count is: 5
Count is: 6
Count is: 7
Count is: 8
Count is: 9
Count is: 10
```

```java
package Package05;

public class ForCounter {
    public static void main (String args[]) {
        int jumlah =0;

        System.out.println("-----");
        for (int i=1; i<=10; i++) {
            System.out.println(i);
            jumlah = jumlah + i;
        }
        System.out.println("-----");
        System.out.println("Jumlah bilangan 1 s.d 10 : " + jumlah);
    }
}
```

```
<terminated> ForCounter [Java Application]
-----
1
2
3
4
5
6
7
8
9
10
-----
Jumlah bilangan 1 s.d 10 : 55
```

# For Demo 2

- Perhatikan inisialisasi (j++) diletakkan di atas (sebelum for)

```
package Package05;

public class ForDemo2 {

    public static void main(String[] args) {

        int j = 1;
        for (; j <= 5; j++) {
            System.out.println(j);
        }

    }

}
```

```
<terminated> ForDemo2 [Java Application]
1
2
3
4
5
```

# For Demo 3

- Perhatikan update (k++) diletakkan di dalam body

```
package Package05;

public class ForDemo3 {

    public static void main(String[] args) {
        int k = 1;
        for (; k <= 5;) {
            System.out.println(k);
            k++;
        }

    }

}
```

```
<terminated> ForDemo3 [Java Application]
1
2
3
4
5
```

# For Demo 4

- Perhatikan condition (m>5) diletakkan di dalam body

```
package Package05;

public class ForDemo4 {

    public static void main(String[] args) {

        int m = 1;
        for (;;) {
            System.out.println(m);
            m++;
            if (m > 5) {
                break;
            }
        }
    }
}
```

```
<terminated> ForDemo4 [Java Application]
1
2
3
4
5
```

# For Demo 5

- If we would like to show 1..10, called forward by using increment (++), then
- If we would like to show 10..1, called backward by using decrement (--)

```
package Package05;

public class ForDemo5 {

    public static void main(String[] args) {

        for (int i=10; i >= 1; i--){
            System.out.println(i);
        }

    }

}
```

```
<terminated> ForDemo5 [Java Application]
10
9
8
7
6
5
4
3
2
1
```

# Enhanced for / for each/for in

- The for statement also has another form designed for iteration through Collections and arrays This form is sometimes referred to as the *enhanced for* statement, and can be used to make your loops more compact and easy to read.
- Syntax

```
for(declaration : expression)  {
   //Statements
}
```

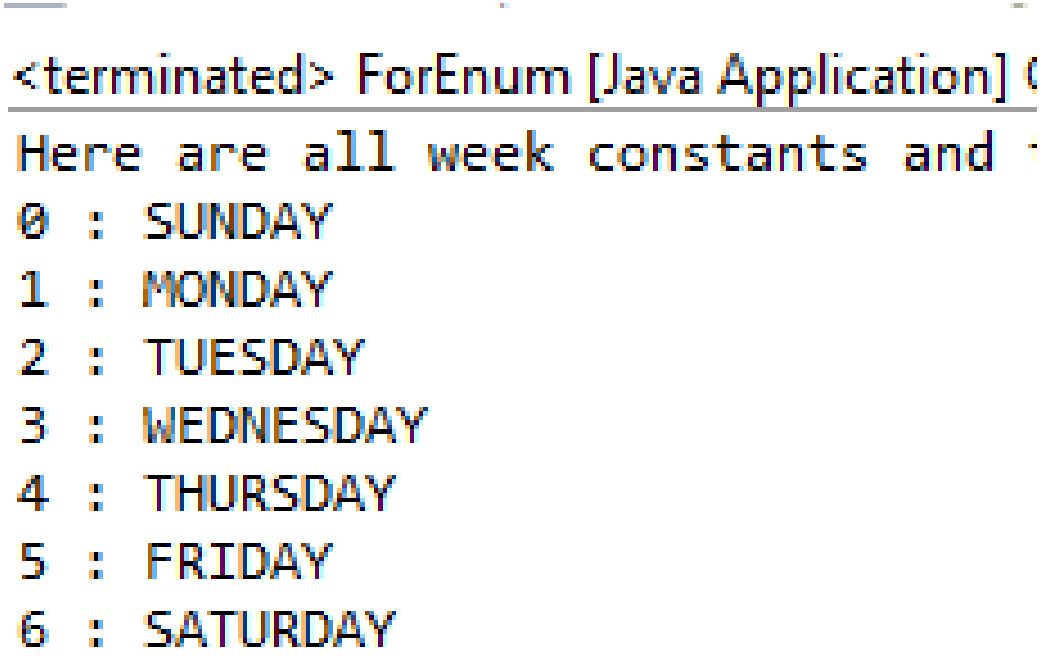
or

```
for(type itr-var : iterableObj)
statement-block
```

Or

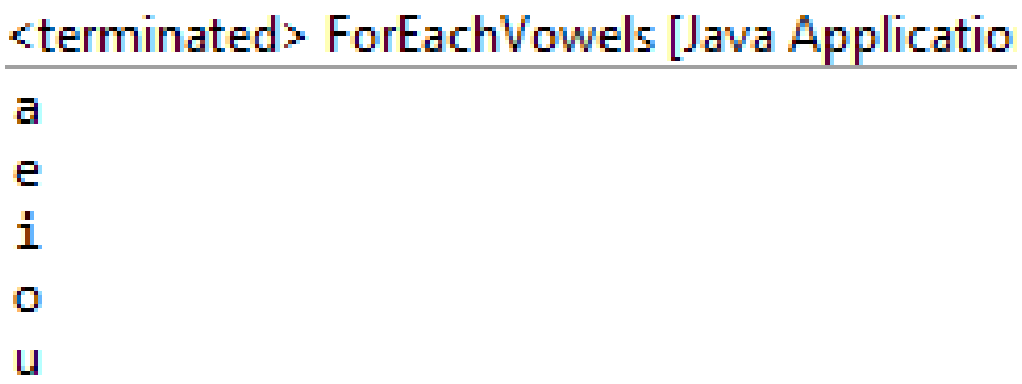
```java
package Package05;

enum Week {
    SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY
}

public class ForEnum {

    public static void main(String args[]) {

        System.out.println("Here are all week constants" +
                " and their ordinal values: ");
        for (Week day : Week.values())
            System.out.println(day.ordinal() + " : " + day);
    }
}
```

- Note that each **`enum`** type has a static **`values`** method that returns an array containing all of the values of the enum type in the order (**`ordinal`**) they are declared.
- This method is commonly used in combination with the for-each loop to iterate over the values of an enumerated type.


<terminated> ForEnum [Java Application]
Here are all week constants and
0 : SUNDAY
1 : MONDAY
2 : TUESDAY
3 : WEDNESDAY
4 : THURSDAY
5 : FRIDAY
6 : SATURDAY

# For Each Array

```
package Package05;

public class ForEachVowels {

    public static void main(String[] args) {

        char[] vowels = { 'a', 'e', 'i', 'o', 'u'};

        for(char ch: vowels){
            System.out.println(ch);
        }
    }
}
```


<terminated> ForEachVowels [Java Applicatio
a
e
i
o
u