

```

1  /*for loop: for loop is the best choice if we know the number of iterations in advance.
2  It is also called entry controlled loop.
3  Syntax:
4  for(initialization;condition;increment/decrement)
5  {
6      Statement or code to be executed;
7  }
8  */
9  /*-----*/
10
11 /*#1WAP to print numbers from 1 to 10 using for loop.*/
12 import java.util.*;
13 class Test1
14 {
15     public static void main(String[] args)
16     {
17         for(int i=1;i<=10;i++)
18         {
19             System.out.print(i+" ");
20         }
21     }
22 }
23 /* OUTPUT
24 1 2 3 4 5 6 7 8 9 10
25 */
26 /*-----*/
27
28 /* #2 Curly Braces are optional. we can take only one statement under for loop
29 without curly braces. */
30 //import java.util.*;
31 class Test2
32 {
33     public static void main(String[] args)
34     {
35         for(int i=1;i<=10;i++)
36
37             System.out.print(i+" ");
38
39     }
40 }
41 /* OUTPUT
42 1 2 3 4 5 6 7 8 9 10
43 */
44 /*-----*/
45
46 /* #3 Curly Braces are optional. we can take only one statement under for loop
47 without curly braces which should not be declarative statement. */
48 //import java.util.*;
49 class Test3
50 {
51     public static void main(String[] args)
52     {
53         //for(int i=1;i<=10;i++)
54
55         //int x;
56
57     }
58 }
59 /* OUTPUT
60 L1.java:50: error: variable declaration not allowed here
61         int x=10;
62         ^
63 1 error
64 */
65 /*-----*/
66
67 /* #4 We can take declarative statement under for loop using curly Braces. */
68 //import java.util.*;
69 class Test4

```

```

70 {
71     public static void main(String[] args)
72     {
73         for(int i=1;i<=10;i++)
74         {
75             int x=10;
76         }
77     }
78 }
79
80 /* OUTPUT
81 No compiler error
82 */
83 /*-----*/
84
85 /* #5 Initialization section will be executed only once. Here we declare
86 and initialize local variables for for loop.
87 int i=0,j=0; can declare multiple variables but should be of same type.
88 int i=0,String s="Amit"; it is not valid because we can declare any number of variables
89 but
90 should be of same type.
91 int i=0,int j=0; it is also not valid. We need to write data type only once.*/
92 //import java.util.*;
93 class Test5
94 {
95     public static void main(String[] args)
96     {
97         //for(int i=1, int j=10;i<j;i++,j--)
98         {
99             //System.out.print(i+j+" ");
100         }
101     }
102 }
103 /* OUTPUT
104 Compiler error
105 */
106 /*-----*/
107
108 /* #6 In the initialization section we can take any valid java statement including
109 sopl\n.*/
110 //import java.util.*;
111 class Test6
112 {
113     public static void main(String[] args)
114     {
115         int i=0;
116         for(System.out.print("HELLO");i<3;i++)
117         {
118             System.out.print(" HI");
119         }
120     }
121 }
122
123 /*OUTPUT
124 HELLO HI HI HI
125 */
126
127 /*-----*/
128
129 /* #7 In the Conditional section we can take any valid java statement but it should be
130 of boolean type. If we will not take anything in the conditional section then compiler
131 will place true in the conditional statement.*/
132 //import java.util.*;
133 class Test7
134 {
135     public static void main(String[] args)
136     {
137         for(int i=1;;i++)

```

```

138         {
139             System.out.print("HELLO");
140         }
141     }
142 }
143
144 /*OUTPUT
145 HELLO will be printed infinite times.
146 */
147 /*-----*/
148
149 /* #8 In the Increment/Decrement section we can take any valid java statement including
150 sopln. We can take multiple increment/decrement variables also.*/
151 //import java.util.*;
152 class Test8
153 {
154     public static void main(String[] args)
155     {
156         int i=0;
157         for(System.out.print("HELLO");i<3;System.out.print(" HI"))
158         {
159             i++;
160         }
161     }
162 }
163
164 /*OUTPUT
165 HELLO HI HI HI.
166 */
167 /*-----*/
168
169 /* #9 All the three parts of for loop are independent of each other and are optional.*/
170 //import java.util.*;
171 class Test9
172 {
173     public static void main(String[] args)
174     {
175         for(;;)
176         {
177
178         }
179     }
180 }
181
182 /*OUTPUT
183 It is infinite loop and will not display anything.
184 */
185 /*-----*/
186
187 /* #10 All the three parts of for loop are independent of each other and are optional.*/
188 //import java.util.*;
189 class Test10
190 {
191     public static void main(String[] args)
192     {
193         for(;;);
194     }
195 }
196
197 /*OUTPUT
198 It is infinite loop and will not display anything.
199 */
200 /*-----*/
201
202 /* #11 Unreachability: in the following java parogram the body of the for loop will be
203 executed infinite times as in the conditional section true is written so the statements
204 after the body of the for loop will never get the chance for its execution. This concept
205 is called Unreachability. And in this case compiler will give compiler Error.*/
206 //import java.util.*;

```

```

207 class Test11
208 {
209     public static void main(String[] args)
210     {
211         for(int i=0;true;i++)
212         {
213             System.out.print("HELLO");
214         }
215         // System.out.print("HI");
216     }
217 }
218
219 /*OUTPUT
220 error: unreachable statement
221         System.out.print("HI");
222         ^.
223 */
224 /*-----*/
225
226 /* #12 Unreachability: in the following java parogram the body of the for loop will not
227 be
228 executed as in the conditional section false is written so the statements within the
229 body
230 of the for loop will never get the chance for its execution. This concept is called
231 Unreachability. And in this case compiler will give compiler Error.*/
232 //import java.util.*;
233 class Test12
234 {
235     public static void main(String[] args)
236     {
237         /* Uncomment this block and see the result
238         for(int i=0;false;i++)
239         {
240             System.out.print("HELLO");
241         }
242         */
243         System.out.print("HI");
244     }
245 }
246
247 /*OUTPUT
248 error: unreachable statement
249         {
250         ^.
251 */
252 /*-----*/
253
254 /* #13 In the above two java parogram in conditional section boolean vaues true/false are
255 written which are known to compiler that's why it will give the concept of unreachability
256 but in case of any other valid condition in conditional section compiler will not
257 evaluate
258 its boolean value at compile time hence in this case it will not give the concept of
259 unreachability.*/
260 //import java.util.*;
261 class Test13
262 {
263     public static void main(String[] args)
264     {
265         int a=10,b=20;
266         for(int i=0;a<b;i++)
267         {
268             System.out.print("HELLO");
269         }
270         System.out.print("HI");
271     }
272 }
273
274 /*OUTPUT

```

```

273 Hello will be printed infinite times.
274 */
275 /*-----*/
276
277 /* #14 In the above two java parogram (Concept 11 and 12) in conditional section boolean
278 vaues true/false are written which are known to compiler that's why it will give the
279 concept
280 of unreachability but in case of any other valid condition in conditional section
281 compiler
282 will not evaluate its boolean value at compile time hence in this case it will not give
283 the
284 concept of unreachability.*/
285 //import java.util.*;
286 class Test14
287 {
288     public static void main(String[] args)
289     {
290         int a=10,b=20;
291         for(int i=0;a>b;i++)
292         {
293             System.out.print("HELLO");
294         }
295         System.out.print("HI");
296     }
297 }
298 /*OUTPUT
299 HI.
300 */
301 /*while loop: while loop is the best choice if we do not know the number of iterations
302 in advance.
303 It is also called entry controlled loop.
304 Syntax:
305 initialization
306 while(Condition of boolean type)
307 {
308     Statement or code to be executed;
309     Increment/Decrement;
310 }
311 */
312 /*-----*/
313
314 /*#15 WAP to print numbers from 1 to 10 using while loop.*/
315 //import java.util.*;
316 class Test15
317 {
318     public static void main(String[] args)
319     {
320         int i=1;
321         while(i<=10)
322         {
323             System.out.print(i+" ");
324             i++;
325         }
326     }
327 }
328 /* OUTPUT
329 1 2 3 4 5 6 7 8 9 10
330 */
331 /*-----*/
332
333 /* #16 Curly Braces are optional. we can take only one statement under while loop
334 without curly braces. */
335 //import java.util.*;
336 class Test16
337 {
338     public static void main(String[] args)
339     {

```

```

338         while(true)
339             System.out.print("Hello ");
340
341     }
342 }
343
344 /* OUTPUT
345 Hello will be printed infinite times.
346 */
347 /*-----*/
348
349 /* #17 Curly Braces are optional. we can take only one statement under while loop
350 without curly braces which should not be declarative statement. */
351 //import java.util.*;
352 class Test17
353 {
354     public static void main(String[] args)
355     {
356         //while(true)
357         // int x;
358
359     }
360 }
361 /* OUTPUT
362 L1.java:357: error: variable declaration not allowed here
363             int x;
364             ^
365 1 error
366 */
367 /*-----*/
368
369 /* #18 We can take declarative statement under while loop using curly Braces. */
370 //import java.util.*;
371 class Test18
372 {
373     public static void main(String[] args)
374     {
375         while(true)
376         {
377             int x=10;
378         }
379
380     }
381 }
382 /* OUTPUT
383 No compiler error
384 */
385 /*-----*/
386
387 /* #19 Unreachability: in the following java program the body of the while loop will be
388 executed infinite times as in the conditional section true is written so the statements
389 after the body of the while loop will never get the chance for its execution. This
390 concept
391 is called Unreachability. And in this case compiler will give compiler Error.*/
392 //import java.util.*;
393 class Test19
394 {
395     public static void main(String[] args)
396     {
397         while(true)
398         {
399             System.out.print("HELLO");
400         }
401         //System.out.print("HI");
402     }
403 }
404 /*OUTPUT
405 L1.java:400: error: unreachable statement

```

```

406         System.out.print("HI");
407         ^
408     */
409     /*-----*/
410
411     /* #20 Unreachability: in the following java parogram the body of the while loop will
412     not be
413     executed as in the conditional section false is written so the statements within the
414     body
415     of the while loop will never get the chance for its execution. This concept is called
416     Unreachability. And in this case compiler will give compiler Error.*/
417     //import java.util.*;
418     class Test20
419     {
420         public static void main(String[] args)
421         {
422             /* Uncomment this block and see the result
423             while(false)
424             {
425                 System.out.print("HELLO");
426             }
427             */
428             System.out.print("HI");
429         }
430     }
431     /*OUTPUT
432     L1.java:422: error: unreachable statement
433         {
434         ^
435     */
436     /*-----*/
437
438     /* #21 In the above two java parogram in conditional section boolean vaues true/false are
439     written which are known to compiler that's why it will give the concept of unreachability
440     but in case of any other valid condition in conditional section compiler will not
441     evaluate
442     its boolean value at compile time hence in this case it will not give the concept of
443     unreachability.*/
444     //import java.util.*;
445     class Test21
446     {
447         public static void main(String[] args)
448         {
449             int a=10,b=20;
450             while(a<b)
451             {
452                 System.out.print("HELLO");
453             }
454             System.out.print("HI");
455         }
456     }
457     /*OUTPUT
458     Hello will be printed infinite times.
459     */
460     /*-----*/
461
462     /* #22 In the above two java parogram (Concept 19 and 20) in conditional section boolean
463     vaues true/false are written which are known to compiler that's why it will give the
464     concept
465     of unreachability but in case of any other valid condition in conditional section
466     compiler
467     will not evaluate its boolean value at compile time hence in this case it will not give
468     the
469     concept of unreachability.*/
470     //import java.util.*;
471     class Test22

```

```

469 {
470     public static void main(String[] args)
471     {
472         int a=10,b=20;
473         while(a>b)
474         {
475             System.out.print("HELLO");
476         }
477
478         System.out.print("HI");
479
480     }
481 }
482 /*OUTPUT
483 HI
484 */
485 /*do while loop: If you want to execute loop body atleast once then you should go for
486 do while loop.
487 It is also called exit controlled loop.
488 Syntax:
489 Initialization
490 do
491 {
492     Statement or code to be executed;
493     Increment/Decrement;
494 }while(boolean type condition);
495 */
496 /*-----*/
497
498 /*#23 WAP to print numbers from 1 to 10 using do while loop.*/
499 //import java.util.*;
500 class Test23
501 {
502     public static void main(String[] args)
503     {
504         int i=1;
505         do
506         {
507             System.out.print(i+" ");
508             i++;
509         }while(i<=10);
510     }
511 }
512 /* OUTPUT
513 1 2 3 4 5 6 7 8 9 10
514 */
515 /*-----*/
516
517 /* #24 Curly Braces are optional. we can take only one statement under do while loop
518 without curly braces. */
519 //import java.util.*;
520 class Test24
521 {
522     public static void main(String[] args)
523     {
524         do
525             System.out.print("Hello ");
526
527         while(true);
528     }
529 }
530 /* OUTPUT
531 Hello will be printed infinite times.
532 */
533 /*-----*/
534
535 /* #25 Curly Braces are optional. we can take only one statement under do while loop
536 without curly braces which should not be declarative statement. */
537 //import java.util.*;

```



```

538 class Test25
539 {
540     public static void main(String[] args)
541     {
542         //do
543         // int x;
544         //while(true);
545
546     }
547 }
548 /* OUTPUT
549 L1.java:543: error: variable declaration not allowed here
550         int x;
551         ^
552 1 error
553 */
554 /*-----*/
555
556 /* #26 We can take declarative statement under do while loop using curly Braces. */
557 //import java.util.*;
558 class Test26
559 {
560     public static void main(String[] args)
561     {
562         do
563         {
564             int x=10;
565         }
566         while(true);
567
568     }
569 }
570 /* OUTPUT
571 No compiler error
572 */
573 /*-----*/
574
575 /* #27 Unreachability: in the following java parogram the body of the do while loop
576 will be
577 executed infinite times as in the conditional section true is written so the statements
578 after the body of the do while loop will never get the chance for its execution. This
579 concept
580 is called Unreachability. And in this case compiler will give compiler Error.*/
581 //import java.util.*;
582 class Test27
583 {
584     public static void main(String[] args)
585     {
586         do
587         {
588             System.out.print("HELLO");
589             }while(true);
590             //System.out.print("HI");
591
592     }
593 }
594 /*OUTPUT
595 L1.java:588: error: unreachable statement
596         System.out.print("HI");
597         ^
598 1 error
599 */
600 /*-----*/
601
602 /* #28 What will be the output of following java program?*/
603 //import java.util.*;
604 class Test28
605 {
606     public static void main(String[] args)

```

```

605     {
606
607         do
608         {
609             System.out.print("HELLO ");
610         }while(false);
611
612         System.out.print("HI");
613
614     }
615 }
616 /*OUTPUT
617 HELLO HI
618 */
619 /*-----*/
620
621 /* #29 In the above 27th java parogram in conditional section boolean vaue true is
622 written which is known to compiler that's why it will give the concept of unreachable
623 but in case of any other valid condition in conditional section compiler will not
624 evaluate
625 its boolean value at compile time hence in this case it will not give the concept of
626 unreachability.*/
627 //import java.util.*;
628 class Test29
629 {
630     public static void main(String[] args)
631     {
632         int a=10,b=20;
633         do
634         {
635             System.out.print("HELLO ");
636         }while(a<b);
637
638         System.out.print("HI");
639     }
640 }
641 /*OUTPUT
642 Hello will be printed infinite times.
643 */
644 /*-----*/
645
646 /* #30 In the above java parogram (Concept 27) in conditional section boolean
647 vaue true is written which is known to compiler that's why it will give the concept
648 of unreachable but in case of any other valid condition in conditional section
649 compiler
650 will not evaluate its boolean value at compile time hence in this case it will not give
651 the
652 concept of unreachability.*/
653 //import java.util.*;
654 class Test30
655 {
656     public static void main(String[] args)
657     {
658         int a=10,b=20;
659         do
660         {
661             System.out.print("HELLO ");
662         }while(a>b);
663
664         System.out.print("HI");
665     }
666 }
667 /*OUTPUT
668 HELLO HI
669 */
670 /*-----*/

```

```

671  /*
672  Transfer Statements:
673  1. break: #31 The break statement is used to terminate the loop immediately.
674  We can use break statement inside switch and inside the loops.
675  */
676  class Test31
677  {
678      public static void main(String[] args)
679      {
680          for(int i=1;i<=10;i++)
681          {
682              if(i==5)
683              {
684                  break;
685              }
686              else
687              {
688                  System.out.print(i+ " ");
689              }
690          }
691      }
692  }
693  /*OUTPUT
694  1 2 3 4
695  */
696  /*-----*/
697
698  /*
699  Transfer Statements:
700  2. continue: #32 The continue statement is used to skip the current iteration of the
701  loop.
702  We can use continue statement inside the loops.
703  */
704  class Test32
705  {
706      public static void main(String[] args)
707      {
708          for(int i=1;i<=10;i++)
709          {
710              if(i==5)
711              {
712                  continue;
713              }
714              else
715              {
716                  System.out.print(i+ " ");
717              }
718          }
719      }
720  }
721  /*OUTPUT
722  1 2 3 4 6 7 8 9 10
723  */

```