|  |
| --- |
| //Initialization part is optional  **public** **class** Eg1 {  **public** **static** **void** main(String[] args) {  **int** i = 0;  **for** (; i < 5; i++) {  System.***out***.println(i);  }  }  }  0  1  2  3  4 |

|  |
| --- |
| //Multiple Statements at initialization  **public** **class** Eg2 {  **public** **static** **void** main(String[] args) {  **int** i = 0;  **for** (System.***out***.println("Hello Java"), System.***out***.println("Hello Python"); i < 10; i++) {  System.***out***.println(i);  }  }  } |

|  |
| --- |
| //Can Take only single Initialization  **public** **class** Eg3 {  **public** **static** **void** main(String[] args) {    **for** (**int** i = 0, j =0; i < 10 && j < 10; i++, j++) {  System.***out***.println(i + " " + j);  }  }  }  0 0  1 1  2 2  3 3  4 4  5 5  6 6  7 7  8 8  9 9 |

|  |
| --- |
| //condition part is optional, works like infinity loop  **public** **class** Eg4 {  **public** **static** **void** main(String[] args) {    //works infinity  // for (int i = 5; ; i++) {  // System.out.println(i);  // }    //dead code  // for (int i = 0; false; i++) {  // System.out.println(i);  // }    // Type mismatch: cannot convert from void to boolean  // for (int i = 0; System.out.println(i);; i++) {  // System.out.println(i);  // }  }  } |

|  |
| --- |
| //increment/decrement part is optional works infinity loop  **public** **class** Eg5 {  **public** **static** **void** main(String[] args) {    **for** (**int** i = 0; i < 5; ) {  System.***out***.println(i);  }  }  }  0  0  Infinity…  //Here JVM will evaluate the variable expression  **public** **class** Eg6 {  **public** **static** **void** main(String[] args) {  //Here JVM will evaluate the variable expression  System.***out***.println("Outside ForLoop");  **for** (**int** i = 0; i <= 5 || i >= 10; i++) {  System.***out***.println(i + " In ForLoop");  }  System.***out***.println("Inside ForLoop");    }  }  Outside ForLoop  0 In ForLoop  1 In ForLoop  2 In ForLoop  3 In ForLoop  4 In ForLoop  5 In ForLoop  Inside ForLoop |

|  |
| --- |
| **public** **class** Eg7 {  **public** **static** **void** main(String[] args) {  //Here Compiler will evaluate the constant expression, not variable expression, here constant expression is true, so we get Unreachable code  System.***out***.println("Ouside ForLoop");  **for** (**int** i = 0; **true**; i++) {  System.***out***.println(i + " In ForLoop");  }  //System.out.println("Inside ForLoop"); // Unreachable code, bcoz loop will move infinity times  //int a = 10; // Unreachable code  }  }  416215 In ForLoop  416216 In ForLoop  416217 In ForLoop |

|  |
| --- |
| //increment/decrement part is optional  **public** **class** Eg8{  **public** **static** **void** main(String[] args) {  //works like infinity loop  // for(int i=0; i<5;) {  // System.out.println(i); // 0 0 0 infinity...  // }  //use manually increment part  **for**(**int** i=5; i<10;) {  System.***out***.println(i); // 5 6 7 8 9  i+=1;  }  5  6  7  8  9  //multiple statements in increment part  **for** (**int** i = 0; i < 5; System.***out***.println("Hello Java"), System.***out***.println("Hello Python")) {  System.***out***.println(i);  i++;  }  }  }  0  Hello Java  Hello Python  1  Hello Java  Hello Python  2  Hello Java  Hello Python  3  Hello Java  Hello Python  4  Hello Java  Hello Python |

|  |
| --- |
| **public** **class** Eg10 {  **public** **static** **void** main(String[] args) {    **int** i =0;  **while** (**true**) {  System.***out***.println(i); // infinite loop  i++;  }  // System.out.println("Hello Java"); //Unreachable code  }  } |

|  |
| --- |
| //do block will execute at least one time  **public** **class** Eg11 {  **public** **static** **void** main(String[] args) {    **do** {  System.***out***.println("Do Block");  } **while** (**false**);  }  }  Do Block |

|  |
| --- |
| //in for loop conditional part is optional, but not in while and do while loop  **public** **class** Eg12 {  **public** **static** **void** main(String[] args) {  **int** i = 0;  **do** {  System.***out***.println(i);  i++;  } **while** (i < 5);  }  }  0  1  2  3  4 |