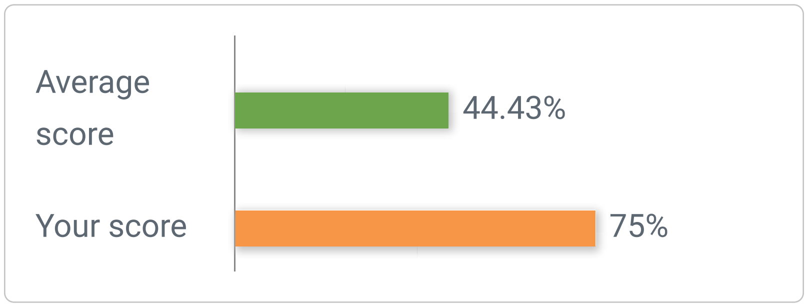
Java For Loop Quiz Online Test

89 Shares Share Tweet Pin Share Share Share

Result

15 out of 20 questions were answered correctly.

Your score is 15 out of 20, (75%)



Review Answers

What will happen when you compile and run the following code?

```
public class Test{

    public static void main(String[] args){

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

    }

}
```

1. 9
2. [✓ 10]

3. ○ 11
4. ○ Compilation error

Incorrect answer.
Option 4 is correct choice. Variable i is declared inside the for loop hence it cannot be accessed outside of it. Hence code will give compilation error “i cannot be resolved to a variable”.

What will happen when you compile and run the following code?

```
public class Test{

    public static void main(String[] args){

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

    }

}
```

1. ○ 9
2. [✓ 10]
3. ○ 11
4. ○ Compilation error

Correct answer.
Option 2 is correct choice. Code will print 10. If the condition part (middle part) is true, loop body is executed and then Increment/decrement is executed. Here, when i becomes 10, the

condition becomes false and control goes out of the loop.

What will happen when you compile and run the following code?

```java
public class Test{

    public static void main(String[] args){

        int i = 0;
        for(; i < 10; i++){
            break;
        }
        System.out.println(i);
    }

}
```

1. [✓ 0]
2. 1
3. 10
4. Compilation error

Correct answer.
Option 1 is correct choice. It is valid to skip the loop initialization part. Since we had already declared a variable i earlier, we can use that in the loop. Also, there are no other compilation problems in the code.

The increment/decrement part of the loop (third part) is executed after every iteration of the loop body. However, since there is a break statement in the loop, control goes out of the loop and i stays at 0.

What will happen when you compile and run the following code?

```
public class Test{

    public static void main(String[] args){
        int i = 0;
        for(i = 100; i < = 0; i -= 10 ){
            System.out.print(i + ", ");
        }
    }
}
```

1. ✓ 100, 90, 80, 70, 60, 50, 40, 20, 10, 0,
2. 100, 90, 80, 70, 60, 50, 40, 20, 10,
3. 90, 80, 70, 60, 50, 40, 20, 10, 0,
4. None of the above

Incorrect answer.
Option 4 is correct choice. There are no compilation errors in the program. Variable i is initialized with 100 and condition is i < = 0 which is false in the first iteration itself and control will not go inside the for loop at all. So there will be no output when we run the program. Had it been i >=0, there would have been output of 100 to 0.

What will happen when you compile and run the following code?

```
public class Test{

    public static void main(String[] args){
        int i = 0;
        for(i = 0; i < 3 ; i++){
            continue;
        }
```

```
            System.out.println(i);
        }
    }
```

1. ◯ 0
2. ✓ 2
3. ◯ 3
4. ◯ Compilation error

Incorrect answer.
Option 3 is correct choice. Java continue keyword makes for loop to skip the current iteration and continue with the next iteration. There will be total 3 iterations after which the value of variable i becomes 3 and that would make the for loop condition false. So finally value of variable i is 3 after the loop.

What will happen when you compile and run the following code?

```
public class Test{

    public static void main(String[] args){
        for(int i = 0; i < 10; i++){
            if(i % 2 == 0)
                continue;
            System.out.println(i);
        }
    }
}
```

1. ◯ Code will print all even numbers between 0 to 10
2. ✓ Code will print all odd numbers between 0 to 10

3. ○ Code will not compile
4. ○ None of the above

Correct answer.
Option 2 is correct choice. For loop starts with 0 and goes up to 9 after which condition becomes false. Inside the loop, if condition checks if the current value of variable i is divisible by 2 by checking the remainder. If it is 0, the current iteration is skipped using continue statement. If not, the number is odd (not divisible by 2) and the value is printed.

What will happen when you compile and run the following code?

```java
public class Test{

    public static void main(String[] args){
        for(int i = 0; i < 5; i++){
            System.out.print( (char)('a' + i) );
        }
    }
}
```

1. ○ Compilation error
2. ○ No output
3. ○ 01234
4. [✓ abcde]

Correct answer.
Option 4 is correct choice. For loop start with 0 and goes up to 4 so in total there will be 5 iterations. Expression ('a' + i) is automatically promoted to int value so it becomes (97 + current value of i) where 97 is the ASCII value of character 'a'. The result is again converted to char by explicit cast using (char). So in the first iteration we get 97 + 0 = 97 which is explicitly casted to char data type so we get character 'a' (ASCII value 97).

In the second iteration ('a' + 1) gives us 98 which is ASCII value of character 'b'. Thus, the code will print a to e in 5 iterations. Similarly, if you want to print capital alphabets, you can replace 'a' with 'A' and the output would be ABCDE.

What will happen when you compile and run the following code?

```
public class Test{

    public static void main(String[] args){
        for(char c = 'a' ; c < 'd'; c++){
            System.out.print(c);
        }
    }
}
```

1. Code will not compile
2. [✓ Code will print abc]
3. Code will print 012
4. None of the above

Correct answer.
Option 2 is correct choice. Using character in for loop is valid. There will be no compilation errors. Loop starts with 'a' and goes up to 'c', so the code will print abc.

What will happen when you compile and run the following code?

```
public class Test{

    public static void main(String[] args){
        for(int i = 65; i < 68 ; i++){
            System.out.print((char)i);
        }
    }
}
```

1. Code will not compile
2. 656667
3. 65666768
4. ✓ None of the above

Correct answer.
Option 4 is correct choice. For loop starts with 65 and goes up to 67. But before printing the value of variable i, it is converted to char using explicit cast (char). 65, 67 and 68 are ASCII values of 'A', 'B', and 'C' respectively so they are printed. Output of the program will be ABC.

What will happen when you compile and run the following code?

```
public class Test{

    public static void main(String[] args){
        for(int i = 0; i < 5 ; i++){
            System.out.print(i++);
        }
    }
}
```

1. Code will not compile
2. 01234
3. 135

4. [✓ 024]

Correct answer.
Option 4 is correct choice. For loop goes from 0 to 4 but the value of variable i is incremented in the loop body as well. So for each iteration variable i is incremented by 2. Code will print 0, 2 and 4 in respective iterations before the condition becomes false (i = 6 which makes i < 5 condition false).

What will happen when you compile and run the following code?

```
public class Test{

    public static void main(String[] args){
        int i = 100;
        for(; i > 50 ;){
            i -= 10;
            System.out.print(i + " ");
        }
    }
}
```

1. Code will not compile
2. 100 90 80 70 60
3. 100 90 80 70 60 50
4. [✓ 90 80 70 60 50]

Correct answer.
Option 4 is correct choice. it is valid to skip any and all parts of the for loop in Java. There are no compilation errors in the code. However, the value of i is decremented in the loop body itself at the start. So the code will start printing from 90 instead of 100 and goes up to 50.

What will happen when you compile and run the following code?

```
public class Test{

    public static void main(String[] args){
        for(int a = 0, b = 3; a < 3 && b > 0; a++, b--){
            System.out.print(a + " " + b + ", ");
        }
    }

}
```

1. ✓ `0 3, 1 2, 2 1,`
2. `1 2, 1 1,`
3. `1 3, 2 2, 3 1,`
4. `Compilation error`

Correct answer.
Option 1 is correct choice. It is valid to define multiple variables of the same type in initialization section and multiple increments and decrements in for loop. It is also valid to define multiple conditions as long as it returns boolean as a whole. There are no compilation errors. For loop will have 3 iterations and prints values of a and b starting from 0 and 3 respectively while increasing value of a and reducing value of b with each iteration.

Will this code compile?

```
public class Test{
    public static void main(String[] args){
        for(;;){}
    }
}
```

1. Yes
2. [✓ No]

Incorrect answer.
Yes is correct choice. It is valid to skip all 3 sections of the for loop. Plus, it is also valid to provide empty loop body. The program will compile without any errors. However, it will create an infinite loop in the absence of loop condition. Exit condition should be provided either as loop condition (middle part of the for loop) or in the loop body itself.

What will happen when you compile and run the following code?

```
public class Test{

    public static void main(String[] args){
        int i = 10;
        for(; i > 0;)
            System.out.print(i + " ");
            i--;
    }

}
```

1. Compilation error
2. [✓ 10 9 8 7 6 5 4 3 2 1]

3. `9 8 7 6 5 4 3 2 0`
4. `None of the above`

Incorrect answer.
Option 4 is correct choice. Since curly braces are not used for for loop body, only System.out.print statement is considered as a part of it. Since variable i is not decremented in the loop, the loop condition will always be true and the loop becomes infinite loop. Code will print 10 infinitely.

What will happen when you compile and run the following code?

```java
public class Test{

    public static void main(String[] args){
        int i = 0, j = 3;
        for(;i < 3 && j > 0;i++, j--);
        {
            System.out.print(i + " " + j + ", ");
        }
    }

}
```

1. `0 3, 1 2, 2 1,`
2. `1 2, 2 1,`
3. `0 3, 1 2, 2 1, 3 0,`
4. [✓ `None of the above`]

Correct answer.
Option 4 is correct choice. There is a semicolon after for loop, so basically it does not have a body. The code block which contains System.out.print statement is considered a separate

block and will be executed independently.

There are no compilation errors in the program. The loop executes normally and after third and last iteration, value of variable i becomes 3 and value of j becomes 0. After that, loop ends and System.out.print statement prints values as "3 0, ".

What will you write inside the for loop to make the code print numbers divisible by 3 between 1 and 10 (both inclusive)?

```
public class Test{

    public static void main(String[] args){
        for( _____________ ){
            if( i % 3 == 0 )
                System.out.println(i);
        }
    }
}
```

1. `int i = 0 ; i < 10; i++`
2. [✓ `int i = 1 ; i <= 10; i++`]
3. `int i = 0 ; i <= 9; i++`
4. `int j = 1 ; j <= 10; j++`

Correct answer.
Option 2 is correct choice. First option starts with 0 and goes up to 9 only. Third option is also same, starting from 0 to 9. Fourth option is logically correct but uses variable name as j which will cause compilation error because we have used variable i in the if condition inside for loop.

What will happen when you compile and run the following code?

```
public class Test{

    public static void main(String[] args){

    for(int i = 0 ; i < 3 ; i++)
        System.out.println(i + " ");
        System.out.println(i + " ");

    }

}
```

1. 0 0 1 1 2 2
2. 0 0 1 1 2 2 3 3
3. 0 1 2 3
4. [✓ None of the above]

Correct answer.
Option 4 is correct choice. Program will not compile. Since there are no curly braces specified after for, only first System.out.println statement is considered part of it. The second System.out.println statement tries to print value of variable i which is local to the for loop only and cannot be accessed outside. Thus program will give compilation error "i cannot be resolved to a variable".

What will happen when you compile and run the following code?

```
public class Test{

    public static void main(String[] args){
        for(int i = 0; i < 3; i++){
                for(int j = 0; j < i; j++){
                    System.out.print(i + " " + j + ", ");
                }
            }
        }
    }
```

1. ◯ `0 0, 0 1, 0 2, 1 0, 1 1, 1 2, 2 0, 2 1, 2 2,`
2. ◯ `0 0, 0 1, 0 2, 0 3, 1 0, 1 1, 1 2, 1 3, 2 0, 2 1, 2 2, 2 3,`
3. [✓ `1 0, 2 0, 2 1`]
4. ◯ `None of the above`

Correct answer.
Option 3 is correct choice. Condition of inner for loop is j < i so in the first iteration of the outer loop, there will not be any iteration of the inner for loop (because of 0 is not less than 0). In the second iteration of the outer for loop, value of variable i is 1 so there will be one iteration of the inner loop which will print 1 and 0 for i and j respectively. In the third iteration i becomes 2, so the inner loop will have two iterations and will print 2 0 and 2 1 respectively.

What will happen when you compile and run the following code?

```
public class Test{

    public static void main(String[] args){
        outer:
        for(int i = 0; i < 3; i++){
            inner:
            for(int j = 0; j < 3; j++){
                if(i == j)
                    break outer;

                System.out.print(i + " " + j + ", ");
```

```
                    }
                }
            }
        }
```

1. 1 0, 2 0, 2 1,
2. 0 1, 0 2, 1 0, 1 2, 2 0, 2 1,
3. 1 0, 2 0, 2 1, 3 0, 3 1, 3 2,
4. [✓ None of the above]

Correct answer.
Option 4 is correct choice. The program will not print anything. Variable i starts with 0, and in the first iteration of the inner loop, j is also 0 which will make i == j is true and break outer statement is executed which breaks the outer loop.

What will happen when you compile and run the following code?

```java
public class Test{

    public static void main(String[] args){
        outer:
        for(int i = 0; i < 3; i++){
            inner:
            for(int j = 0; j < 3; j++){
                if(i == j)
                    continue outer;
                System.out.print(i + " " + j + ", ");
            }
        }
    }
}
```

1. ○ 0 1, 0 2, 1 0, 1 2, 2 0, 2 1,
2. [✓ 1 0, 2 0, 2 1,]
3. ○ 0 2, 0 3, 0 3, 1 0, 1 2, 1 3, 2 0, 2 1, 2 3, 3 0, 3 1, 3 2,3 3,
4. ○ None of the above

Correct answer.

Option 2 is correct choice. The continue outer statement is executed every time condition i == j becomes true, and that it will cause skipping all the iterations of the inner loop as well the current iteration of the outer loop. Code will not print anything for the first iteration of the outer loop (because 0 == 0), 1 0 in the second iteration and 2 0, 2 1 in the third iteration of the outer loop.

89 Shares Share Tweet Pin Share Share

Share