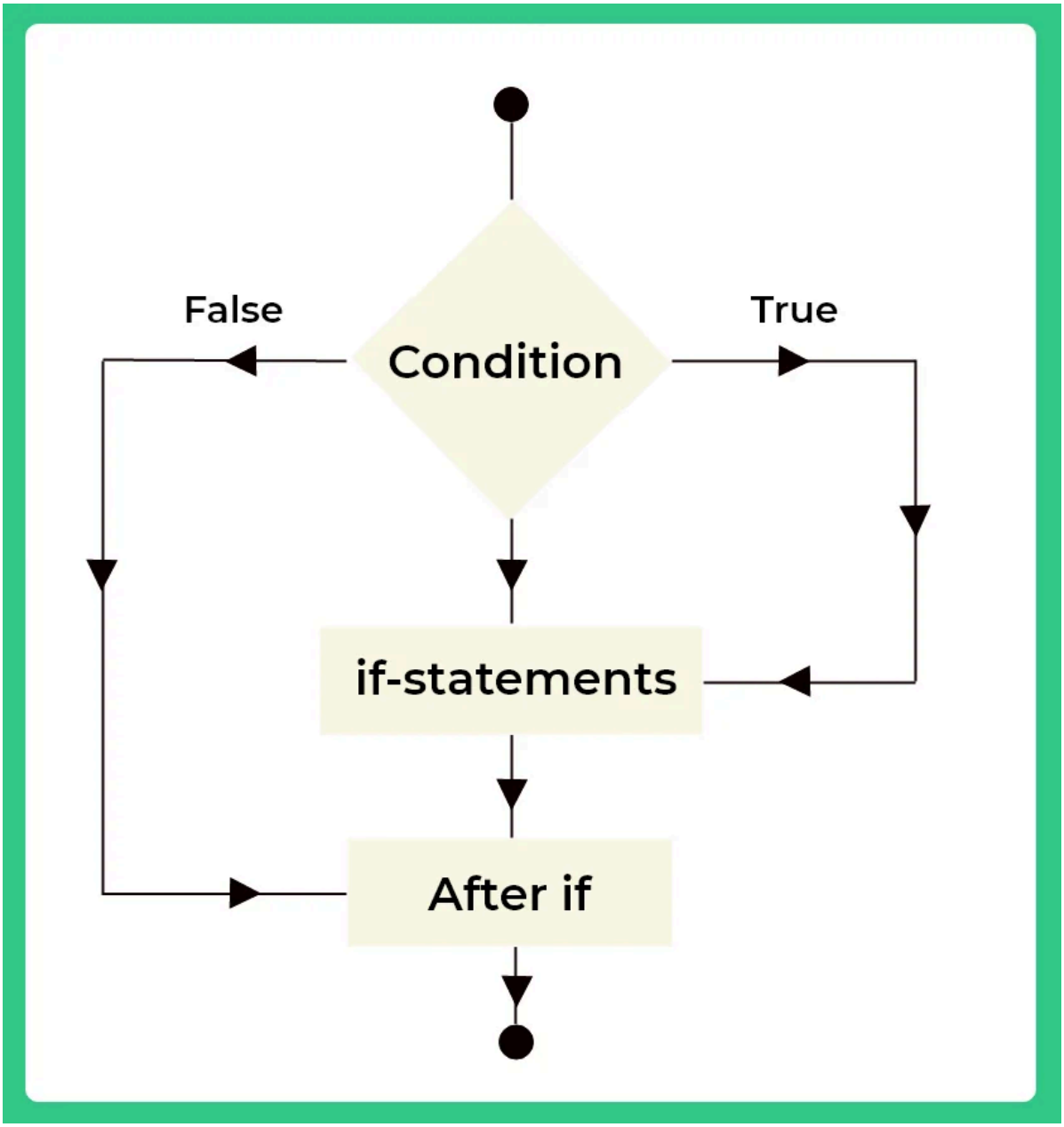
Conditional Statement

If Statement

- The control initially falls into the if block.
- The flow then jumps to the specified ‘ Condition ‘.
- Now, The condition is tested.
    1. If Condition yields true, *execute the part of code within the block*
    2. If Condition yields false, *control falls out of the if- block and the statements after if are executed*.

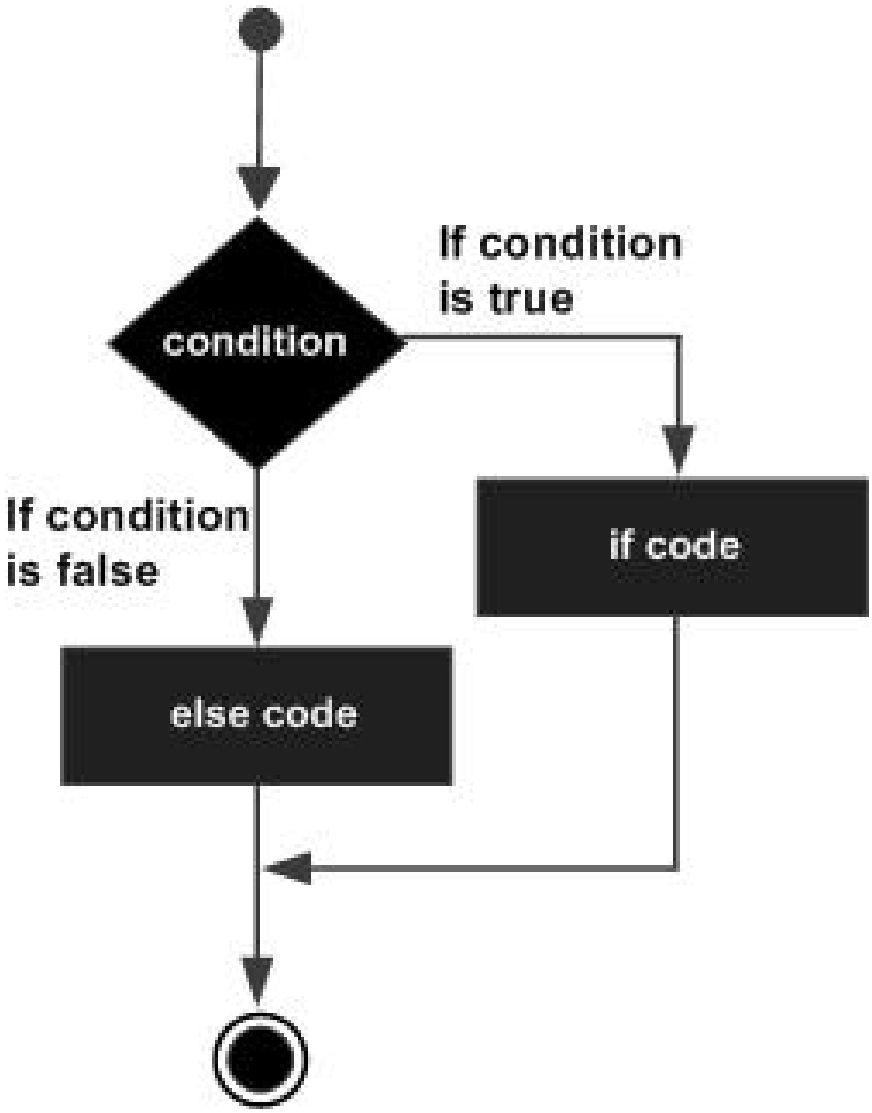



```
class Main{
  public static void main(String[] args) {
      int income = 5000;
      // checks if income is greater than 1000

      if (income < 4000)
      {
         System.out.println("No tax");
```

```
    }
  }
}
```

If- Else Statement

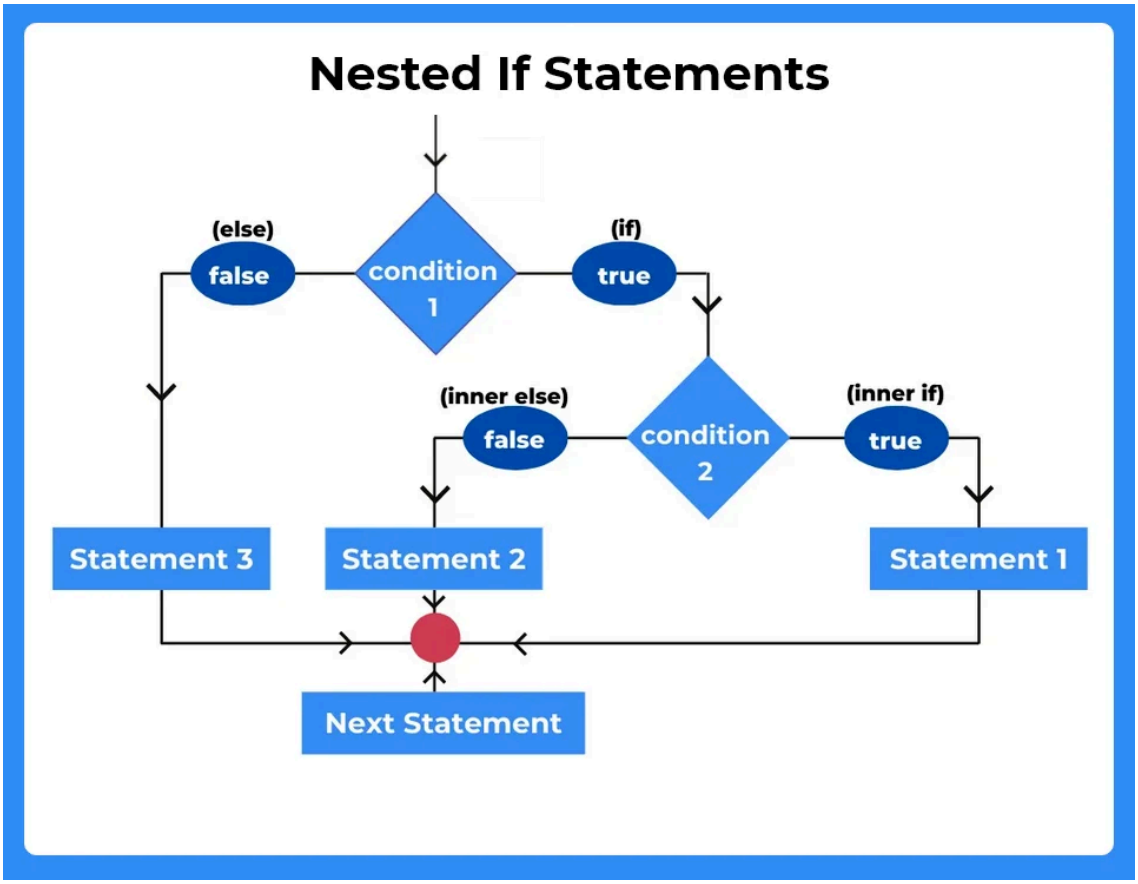
- The control initially falls into the if block.
- The flow then jumps to the specified ‘ Condition ‘.
- Now, The condition is tested.
    1. If Condition yields true, *execute the part of code within the block*
    2. If Condition yields false, *the set of statements within the body of ‘else’ are executed*.




```
public class Main
{
   public static void main(String[] args) {
      int x = 150;
      if (x<100)
            {
           System.out.println("This is the if block");
           }
     else
          {
             System.out.println("This is the else block");
```

```
        }
    }
}
```

Nested If/Else Statement

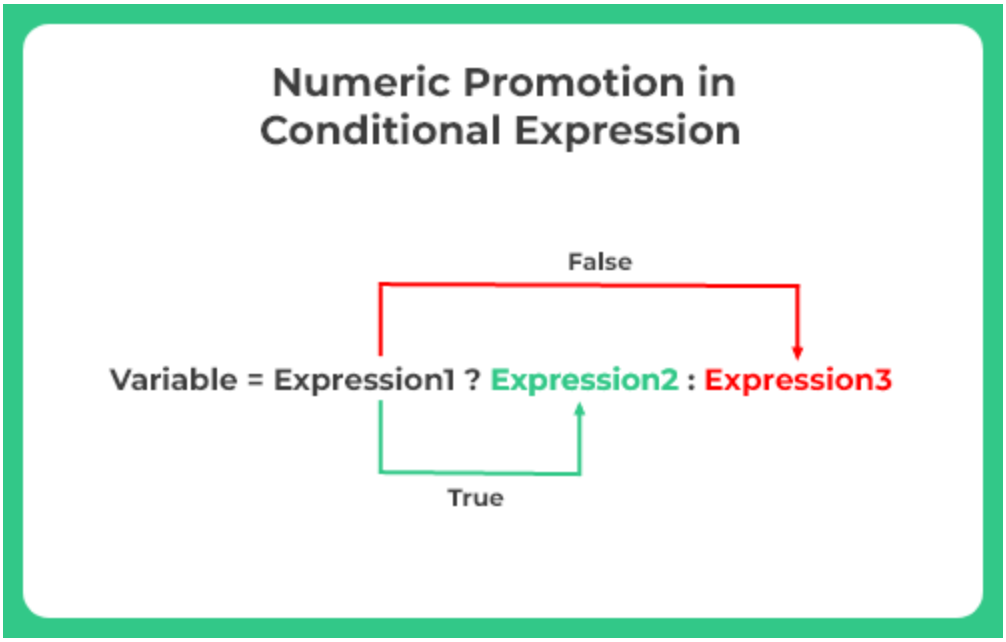



```
public class Main
{
   public static void main(String[] args) {
       int a= 100;
       if(a<250)
       {
          if(a<150)
          System.out.println("This is bigger then  150");
```

```
        else
           System.out.println("This is less then 150");
        }

        else
        System.out.println("This is Less then 250");

    }
}
```

Numeric Promotion in Conditional Expression

The boolean value of one expression is used by the conditional operator?: to determine which of the other two expressions should be evaluated



Looping

Looping is a cardinal feature of any programming language. It allows the execution of
a set of instructions or commands repetitively for a specified number of iteration, until a certain condition is met.

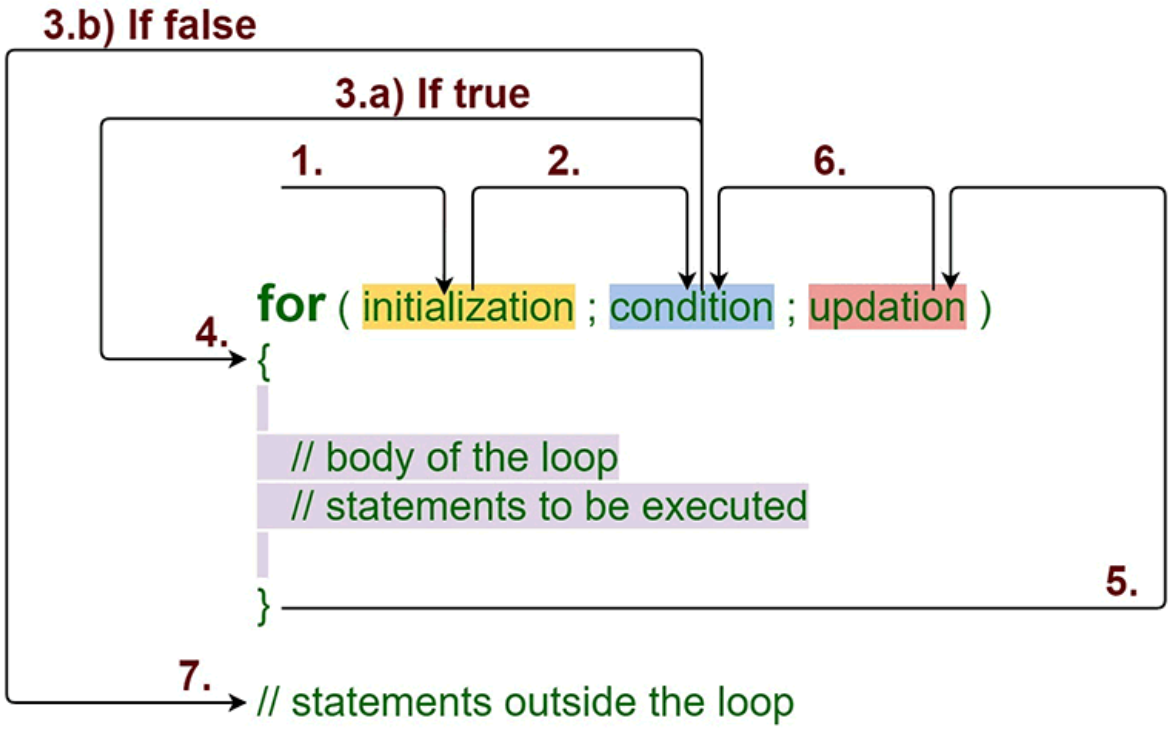
Using loops saves the programmer from the hassle of writing the same code again and again to obtain the desired results.

For Loop

The For loop is a functionality of a programming language that comes into play when some code needs to be executed iteratively. It is the most favorable looping technique to be used where the number of iterations are known beforehand.

- **Initialization:** It marks the beginning of the for loop. We can use an already declared variable or a local variable can be declared for the loop only.
- **Check Condition:** It is basically the exit condition for the loop which yields a boolean
  value. This condition is evaluated in each iteration. If the condition is true then the statements inside for loop body gets executed. But once the condition returns false, the statements in for loop does not execute and the control gets transferred to the next statement in the
  program after for loop.
- **Code/Statements inside loop:** If the check condition yields true this block of code is executed for a specific number of iterations.
- **Update Counter:** After each iteration of the loop the counter is updated. It is either incremented or decremented.
- **Loop Termination:** When the check condition yields false the control flow comes out of the loop and it is terminated.

For Loop




```
public class Main
{
   public static void main(String[] args) {
 for(int i = 0; i < 5; i++)
      System.out.println("Number:"+i);
   }
}
```

While Loop

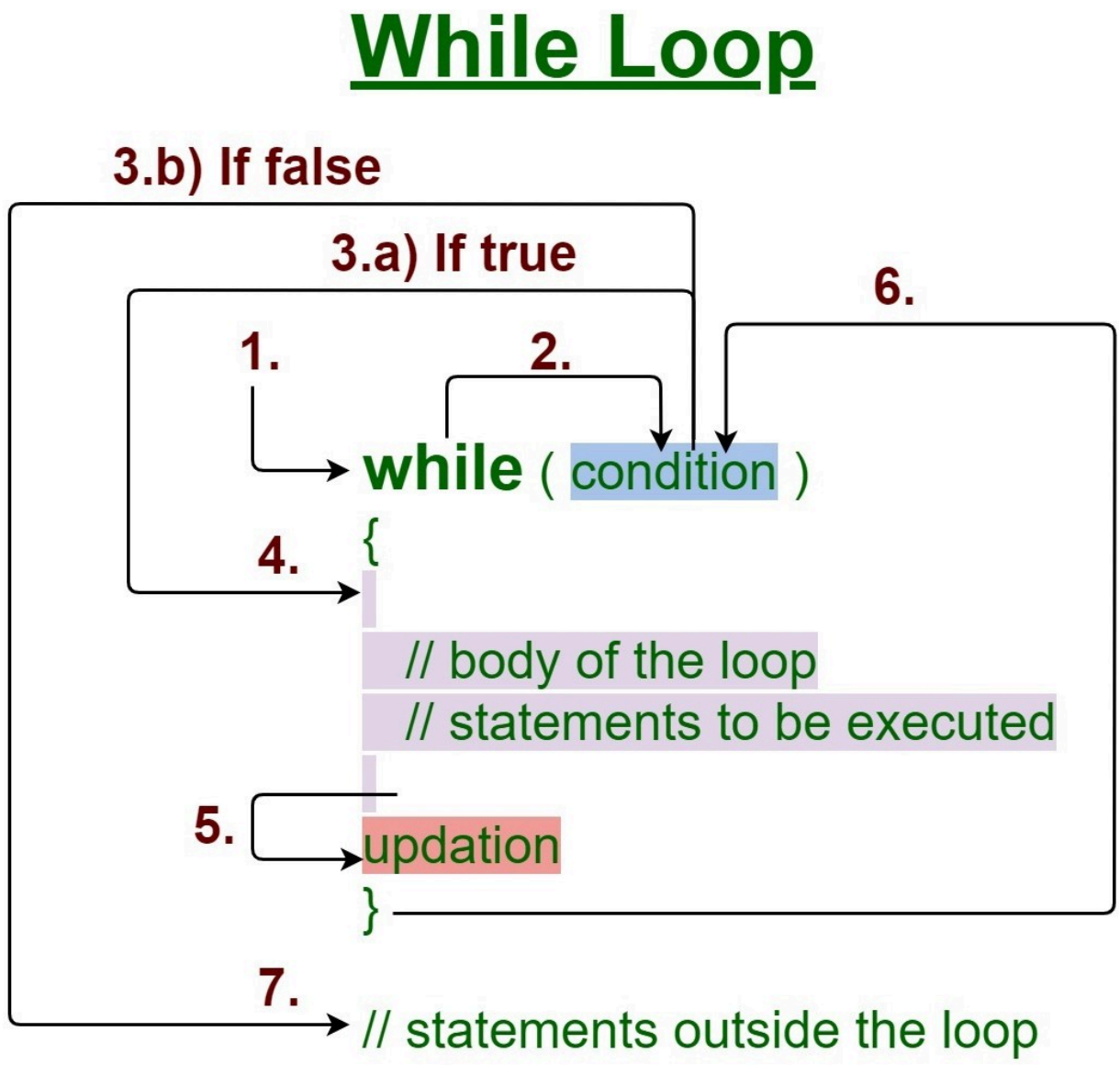
The While loop is a functionality of a programming language that comes into play when some code needs to be executed iteratively until a given condition is true. In other words, the code is repetitively executed based on a given boolean condition. In this type of looping
technique the control flow enters the loop only when the boolean condition returns true.

The three steps in while loop can be seen as:

Test: A boolean function which when tested true advances the control flow. It needs to be tested after each iteration.

Statement/Conditional code: Once the condition is satisfied the the body of the code specified by the programmer as per his need is executed.

Increment/Decrement: The code is incremented/decremented in order to bring the condition close to zero or false so that the loop can terminate.

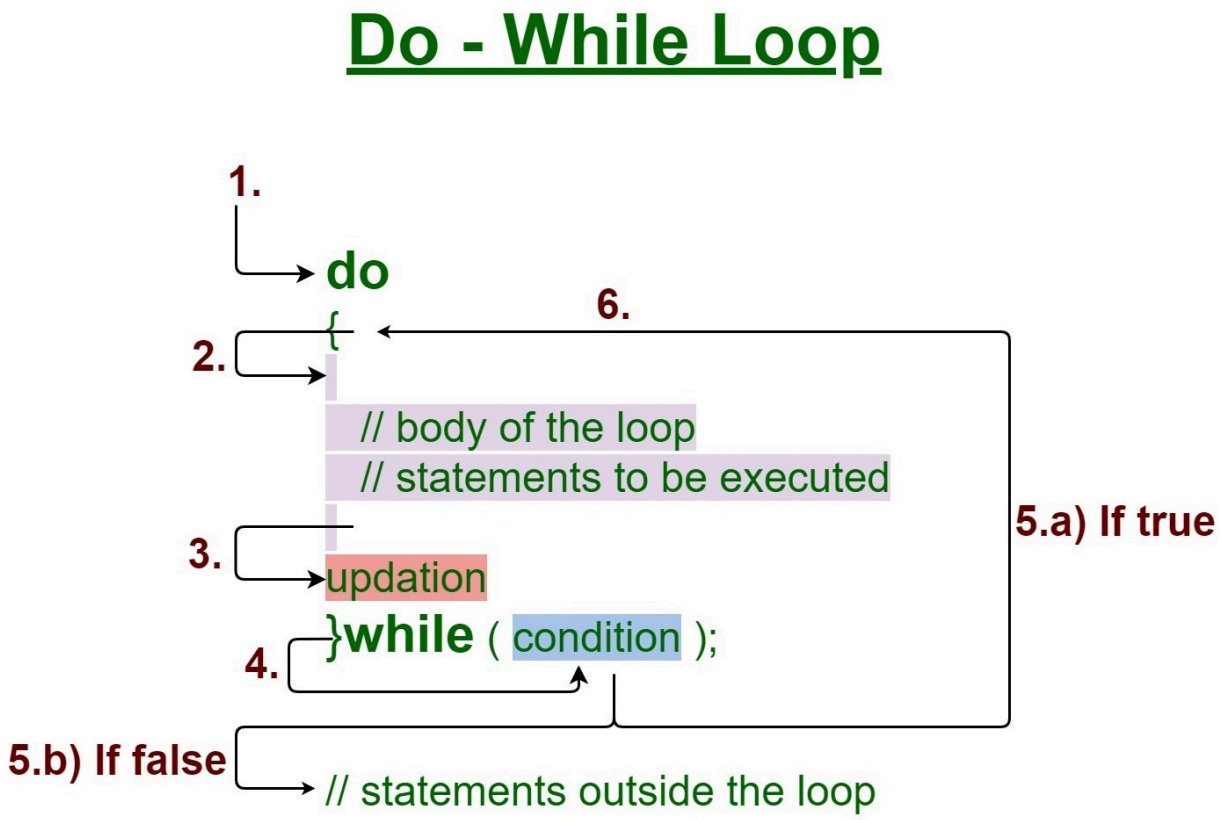


```
public class Main
{
  public static void main (String[]args)
  {
    int i, num = 1, c;
      System.out.println (" Prime Numbers from 1 to 100 are : ");
    while (num <= 100)
      {
    c = 0;
    i = 2;
    while (i <= num / 2)
      {
        if (num % i == 0)
          {
        c++;
          }
        i++;
      }
    if (c == 0 && num != 1)
      {
        System.out.print (num + " ");
      }
    num++;
      }
  }
}
```

Do While Loop

A do-while loop is a functionality that is similar to the standard while loop except the fact that in this type of looping technique the body of the loop will be executed at least once, even if the condition check returns false.. Also, a do-while loop evaluates the condition at the end of the loop.

- The control flow first enters the loop.
- The body of the loop is executed initially.
- Now, the condition specified is evaluated.
    - If it returns true, the body of the loop is executed again.
    - If it returns false, the control flow falls out of the loop

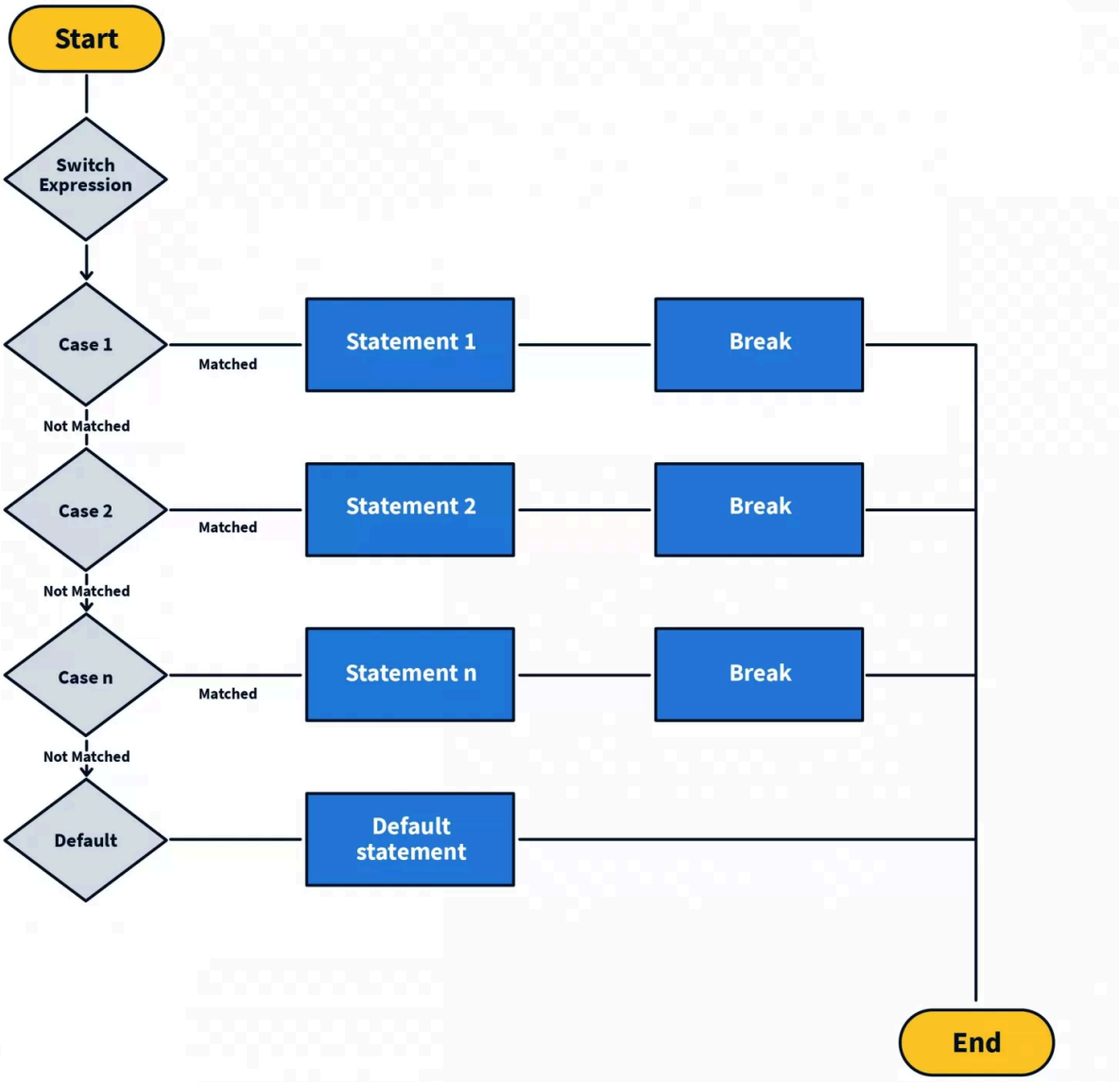



```
public class Main
{
   public static void main(String args[])
   {
      int i = 1;
```

```
      do {

          System.out.println(i*4);

            i++;
        }

       while (i < 11);
   }
}
```

Switch Statement

- The expression used in a switch statement must have an integral or enumerated type, or be of a class type in which the class has a single conversion function to an integral or enumerated type.
- You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.
- The constant-expression for a case must be the same data type as the variable in the switch, and it must be a constant or a literal.
- When the variable being switched on is equal to a case, the statements following that case will execute until a break statement is reached.
- When a break statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
- Not every case needs to contain a break. If no break appears, the flow of control will *fall through* to subsequent cases until a break is reached.
- A switch statement can have an optional default case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No break is needed in the default case.




```
public class Main
{
   public static void main(String[] args)
   {
```

```
        int month = 6;
        String monthname;
        switch (month) {
            case 1: monthname = "January";
            break;
            case 2: monthname = "February";
            break;
            case 3: monthname = "March";
            break;
            case 4: monthname = "April";
            break;
            case 5: monthname = "May";
            break;
            case 6: monthname = "June";
            break;
            case 7: monthname = "July";
            break;
            case 8: monthname = "August";
            break;
            case 9: monthname = "September";
            break;
            case 10: monthname = "October";
            break;
            case 11: monthname = "November";
            break;
            case 12: monthname = "December";
            break;
            default: monthname = "Invalid month";
            break;
        }
        System.out.println("Hey Prepster, Happy " + monthname);
    }
}
```