Control Structures

- Loop: while, for, do
- Decision: if, switch
- Branching: break, continue, return
- enum

LOOP: WHILE, FOR, DO

Loop

- A **loop** statement allows us to execute a statement or group of statements multiple times

loop	Description
while	Repeats a statement or group of statements while a given condition is true
for	Execute a sequence of statements for a specific number of times
do	Like a while statement, except that it tests the condition at the end of the loop body

while loop

- A while loop statement repeatedly executes a target statement as long as a given condition is true

```
while ( condition ) {
   // Statements
}
```

while loop

```
public class WhileLoop_1 {
  public static void main(String args[]) {
    int x = 1;
    while ( x <=10 ) {
      System.out.printf("value of x : %d%n", x );
      x++;
    }
  }
}
```

```
value of x : 1
value of x : 2
value of x : 3
value of x : 4
value of x : 5
value of x : 6
value of x : 7
value of x : 8
value of x : 9
value of x : 10
```

while loop

```
public class WhileLoop_2 {
   public static void main(String args[]) {
      int sum = 0 ;
      int i = 1 ;
      while ( (i <= 10) && (sum < 30) ) {
         sum += i ;
         System.out.printf("Sum of 1 to %d: %d%n", i, sum);
         i ++ ;
      }
   }
}
```

```
Sum of 1 to 1: 1
Sum of 1 to 2: 3
Sum of 1 to 3: 6
Sum of 1 to 4: 10
Sum of 1 to 5: 15
Sum of 1 to 6: 21
Sum of 1 to 7: 28
Sum of 1 to 8: 36
```

while loop

```
import java.util.Scanner;
public class WhileLoop_3 {
   public static void main(String[] args) {
      final String inputString = "10 20 30 50";
      final Scanner scanner = new Scanner(inputString);

      int sum = 0;
      while ( scanner.hasNext() && (sum <= 50) ) {
         final int value = scanner.nextInt();
         sum += value;
      }
      scanner.close();

      System.out.println(sum);      // 60(=10+20+30)
   }
}
```

for loop

- A **for** loop is a repetition control structure that allows you to efficiently write a loop that needs to be executed a specific number of times.
- A **for** loop is useful when you know how many times a task is to be repeated

```
for ( init; condition; update ) {
   // Statements
}
```

for loop

```
public class ForLoop_1 {
   public static void main(String args[]) {
      for ( int i = 1; i <= 10; i ++ ) {
         System.out.printf("value of x : %d%n", i );
      }
   }
}
```

```
value of x : 1
value of x : 2
value of x : 3
value of x : 4
value of x : 5
value of x : 6
value of x : 7
value of x : 8
value of x : 9
value of x : 10
```

Enhanced for loop

- for statement also has another form designed for iteration through Collections and arrays

```
public class ForLoop_2 {
  public static void main(String args[]) {
    final int[] numbers = new int[10];
    for ( int i = 0; i < numbers.length; i ++ )
      numbers[i] = i+1;

    for ( final int i : numbers ) {
      System.out.printf("value of x : %d%n", i );
    }
  }
}
```

Enhanced for loop

```
import java.util.ArrayList;
import java.util.List;
public class ForLoop_3 {
   public static void main(String args[]) {
      final List<String> messages = new ArrayList<>();
      messages.add("Hello");
      messages.add(“자바₩tGreat");
      messages.add(“₩t10 !");

      int wordCount = 0;
      int charCount = 0;
      for ( final String message : messages ) {
         System.out.println(message);
         wordCount ++;
         charCount += message.trim().length();
      }
      System.out.println("Word: " + wordCount + ", Chars: " + charCount);
   }
}
```

```
Hello
자바        Great
            10 !
Word: 3, Chars: 17
```

Enhanced for loop

```
import java.util.ArrayList;
import java.util.List;

public class ForLoop_4 {
   public static void main(String args[]) {
      final List<String> messages = new ArrayList<>();

      messages.add("Hello");
      messages.add("Java");

      for ( final String message : messages ) {
         final char[] charArray = message.toCharArray();
         for ( final char aChar : charArray )
            System.out.print(Character.toUpperCase(aChar));
         System.out.println();
      }
   }
}
```

```
HELLO
JAVA
```

do loop

- A **do...while** loop is similar to a while loop, except that a do...while loop is guaranteed to execute at least one time

```
do {
   // Statements
} while ( condition ) ;
```

do loop

```
public class DoLoop_1 {
   public static void main(String args[]) {
      int x = 1;
      do {
         System.out.printf("value of x : %d%n", x );
         x++;
      } while ( x <= 10 );
   }
}
```

```
value of x : 1
value of x : 2
value of x : 3
value of x : 4
value of x : 5
value of x : 6
value of x : 7
value of x : 8
value of x : 9
value of x : 10
```

while loop vs do loop

```
public class Do_While_Compare {
   public static void main(String args[]) {
      int x = 0;
      int sum1 = 0;
      do {
         x ++ ;
         sum1 += x;
      } while (x < 10 ) ;
      System.out.println(sum1);    // 55

      int y = 0;
      int sum2 = 0;
      while ( y < 10 ) {
         y ++;
         sum2 += y;
      }
      System.out.println(sum2);    // 55
   }
}
```

while loop vs do loop

```
public class Do_While_Compare {
   public static void main(String args[]) {
      int num;
      Scanner scanner = new Scanner(System.in);
      do {
         System.out.println("Enter number: ");
          num = scanner.nextInt(); //11
      } while (num < 10 ) ;
      System.out.println(sum1);        // 11
   }
}
```

```
public class Do_While_Compare2 {
   public static void main(String args[]) {
      int num;
      Scanner scanner = new Scanner(System.in);
     //num??
      while (num < 10) {
         System.out.println("Enter number: ");
        num = scanner.nextInt(); //11
      }
      System.out.println(sum1);        // 11
   }
}
```