Programming in Java

Dr. Nyein Aye Maung Maung
Dr. Eng (Ritsumeikan University, Japan)
Lecturer
Computer Engineering and Information Technology Dept.
Yangon Technological University

Course Schedule

Week	Topics
Week 1	Overview of JAVA, Data Types, Variables and Arrays
Week 2	Operators and Control Statements
Week 3	Classes and A closer look at Methods and Classes
Week 4	Inheritance, Overriding and Polymorphism
Week 5	Interfaces and Packages
Week 6	Exception Handling and Multi-threaded Programming
Week 7	String Handling
Week 8	Exploring java.lang and More utilities classes
Week 9	Java Collections Framework
Week 10	Java I/O
Week 11	Basic Graphical User Interface
Week 12	Event Handling
Week 13	Database Programming
Week 14	Applet and Networking

Lecture 2

Operators and Control Statements

Outline of Class (Lecture 1)

- Operators
- Control Statements

Lecture Objectives

- To introduce different operators
  - To apply the use of operators in mathematical expressions
- To introduce several control statements used in Java
  - To implement selection control using ***if*** and **switch** statements
  - To combine conditions using logical operators
  - To follow the loop design strategy to develop **loops**
  - To write programs for executing statements repeatedly using loops
  - To implement program control with ***break*** and ***continue***
  - To use control statements in application development

Topic 1

Operators

Operators

- Arithmetic Operators
- Assignment Operators
- Unary Operators
- Relational Operators
- Boolean Logical Operators
- The Bitwise and Shift Operators
- Operators Precedence
- Using Parentheses

Operators - Arithmetic

- Perform arithmetic operations such as addition, subtraction, multiplication, division and modulus

Operator	Result
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus

Example 2.1. Demonstrate basic arithmetic operators

```
class BasicMath {
public static void main(String args[])
{
// arithmetic using integers
 System.out.println("Integer Arithmetic");
int a = 1 + 1;
int b = a * 3;
int c = b / 4;
int d = c - a;
int e = -d;
System.out.println("a = " + a);
System.out.println("b = " + b);
System.out.println("c = " + c);
System.out.println("d = " + d);
System.out.println("e = " + e);
// arithmetic using doubles
System.out.println("\nFloating Point Arithmetic
");
double da = 1 + 1;
double db = da * 3;
double dc = db / 4;
double dd = dc - a;
double de = -dd;
System.out.println("da = " + da);
System.out.println("db = " + db);
System.out.println("dc = " + dc);
System.out.println("dd = " + dd);
System.out.println("de = " + de);
}
}
```

Output

```
Integer Arithmetic
a = 2
b = 6
c = 1
d = -1
e = 1
```

```
Floating Point Arithmetic
da = 2.0
db = 6.0
dc = 1.5
dd = -0.5
de = 0.5
```

Example 2.2. Demonstrate modulus operator

```
class Modulus
{
        public static void main(String args[])
        {
                int x = 42;
                double y = 42.25;
                System.out.println("x mod 10 = " + x % 10);
                System.out.println("y mod 10 = " + y % 10);
        }
}
```

Output

```
x mod 10 = 2
y mod 10 = 2.25
```

Operators- Assignment

- Also called Arithmetic Compound Assignment Operators
  - **eg.** a = a + 4; → a += 4; a=a+1 → a++
    a = a % 2; → a %= 2;

Operator	Result
+=	Addition assignment
-=	Subtraction assignment
*=	Multiplication assignment
/=	Division assignment
%=	Modulus assignment

Example 2.3. Demonstrate compound assignment operators

```
class OpEquals
{
        public static void main(String args[])
        {
            int a = 1;
            int b = 2;
            int c = 3;
            a += 5;
            b *= 4;
            c += a * b;
            c %= 6;
            System.out.println("a = " + a);
            System.out.println("b = " + b);
            System.out.println("c = " + c);
        }
}
```

Output

a = 6
b = 8
c = 3

Operators - Unary

- Unary operators are used to increment or decrement a particular value.

Operator	Result
++	Increment
--	Decrement

- The increment operator increases its operand by one. The decrement operator decreases its operand by one.

```
x = x + 1;   →        x++;
x = x – 1;   →         x--;
y=x++        →        y=x;       x=x+1;
y=++x        →         x=x+1;  y=x;
```

Example 2.4. Demonstrate increment operators

```
class IncDec
{
        public static void main(String args[])
         {
            int a = 1;
            int b = 2;
            int c;
            int d;
            Int e;
            c = ++b;
            d = a++;
            e=++a;
            c++;
            System.out.println("a = " + a);
            System.out.println("b = " + b);
            System.out.println("c = " + c);
            System.out.println("d = " + d);
        }
}
```

Output

a = 3
b = 3
c = 4
d = 1

Operators - Relational

- It defines some kind of relation between two entities.

Operator	Description	Result
==	Equal to	x==y → True if x equals y, otherwise false
!=	Not equal to	x!=y → True if x is not equal to y, otherwise false
<	Less than	x<y → True if x is less than y, otherwise false
>	Greater than	x > y → True if x is greater than y, otherwise false
>=	Greater than or equal to	x>=y → True if x is greater than or equal to y, otherwise false
<=	Less than or equal to	x<=y → True if x is less than or equal to y, otherwise false

Example 2.5. Demonstrate relational operators

```
public class operators
{
        public static void main(String[] args)
        {
        int a = 20, b = 10;
        int ar[] = { 1, 2, 3 };
        int br[] = { 1, 2, 3 };
        boolean condition = true;
        //various conditional operators
        System.out.println("a == b :" + (a == b));
        System.out.println("a < b :" + (a < b));
        System.out.println("a <= b :" + (a <= b));
        System.out.println("a > b :" + (a > b));
        System.out.println("a >= b :" + (a >= b));
        System.out.println("a != b :" + (a != b));
        // Arrays cannot be compared with relational operators because objects
        store references not the value
        System.out.println("Arrays: " + (ar == br));
        System.out.println("condition==true :" + (condition == true));
        }
}
```

Output

```
a == b :false
a < b :false
a <= b :false
a > b :true
a >= b :true
a != b :true
Arrays: false
condition==true :true
```

Operators- Boolean Logical

Operator	Result
&	Logical AND
|	Logical OR
^	Logical XOR
||	Short-circuit OR
&&	Short-circuit AND
!	Logical unary NOT
&=	AND assignment
|=	OR assignment
^=	XOR assignment
!=	Not equal to
?:	Ternary if-then-else

Example 2.6. Demonstrate boolean logical operators

```
// Demonstrate the boolean logical operators.
class BoolLogic {
  public static void main(String args[]) {
    boolean a = true;
    boolean b = false;
    boolean c = a | b;
    boolean d = a & b;
    boolean e = a ^ b;
    boolean f = (!a & b) | (a & !b);
    boolean g = !a;
    System.out.println("        a = " + a);
    System.out.println("        b = " + b);
    System.out.println("      a|b = " + c);
    System.out.println("      a&b = " + d);
    System.out.println("      a^b = " + e);
    System.out.println("!a&b|a&!b = " + f);
    System.out.println("       !a = " + g);
  }
}
```

Output

```
a = true
b = false
a|b = true
a&b = false
a^b = true
!a&b|a&!b = true
!a = false
```

- **Ternary if-then-else**

expression1 **?** expression2 **:** expression3

- Here, expression1 can be any expression that evaluates to a boolean value.
- If expression1 is true, then expression2 is evaluated; otherwise, expression3 is evaluated.

Example 2.7. Demonstrate Ternary if-then-else operator

```
// Demonstrate ?.
class Ternary {
  public static void main(String args[]) {
    int i, k;

    i = 10;
    k = i < 0 ? -i : i; // get absolute value of i
    System.out.print("Absolute value of ");
    System.out.println(i + " is " + k);

    i = -10;
    k = i < 0 ? -i : i; // get absolute value of i
    System.out.print("Absolute value of ");
    System.out.println(i + " is " + k);
  }
}
```

Output

Absolute value of 10 is 10
Absolute value of -10 is 10

Operator- Bitwise

Operator	Result
& - AND	The & operator compares corresponding bits of two operands. If both bits are 1, it gives 1 else 0.
| - OR	The | operator compares corresponding bits of two operands. If either of the bits is 1, it gives 1 else 0.
^ - XOR	The ^ operator compares corresponding bits of two operands. If corresponding bits are different, it gives 1 else 0.
~ - NOT	The ~ operator inverts the bit pattern. It makes every 0 to 1 and every 1 to 0.

- **The Bitwise NOT**
  - Also called the bitwise complement, the unary NOT operator, ~, inverts all of the bits of its operand
  - 00101010 → 11010101

- **The Bitwise AND**
  - Produce a 1 bit if both operands are also 1

```
  00101010   42
&00001111    15
 ___________
  00001010   10
```

- **The Bitwise OR**
  - If either of the bits in the operands is a 1, then the resultant bit is a 1,

```
  00101010   42
| 00001111   15
 ___________
  00101111   47
```

- The Bitwise XOR
  - If exactly one operand is 1, then the result is 1. Otherwise, the result is zero

```
  00101010   42
^ 00001111   15
  ---------
  00100101   37
```

Example 2.8. Demonstrate bitwise operators

```
public class operators
{
    public static void main(String[] args)
    {
        int a = 0x0005;   // 0101
        int b = 0x0007;  //0111
        System.out.println("a&b = " + (a & b));
        System.out.println("a|b = " + (a | b));
        System.out.println("a^b = " + (a ^ b));
        System.out.println("~a = " + ~a);
        a=a&b
        a &= b;
        System.out.println("a= " + a);
    }
}
```

Output

```
a&b = 5
a|b = 7
a^b = 2
~a = -6
a= 5
```

Shift Operators

Operator	Result
<< (left shift operator)	Shifts the value to the left which specified by the right operand.
>> (right shift operator)	Shifts the value by zeroes defined by left operand.
>>> (unsigned right shift operator)	It fills the void on left with zeroes which are set by the right operand.

Example 2.9. Demonstrate the shift operators

```
public class operators
{
	public static void main(String[] args)
	{
		int a = 0x0005;
		int b = -10;
		// left shift operator
		// 0000 0101<<2 =0001 0100(20), similar to 5*(2^2)
		System.out.println("a<<2 = " + (a << 2));
		// right shift operator, 0000 0101 >> 2 =0000 0001(1)
		// similar to 5/(2^2)
		System.out.println("a>>2 = " + (a >> 2));
		// unsigned right shift operator
		System.out.println("b>>>2 = "+ (b >>> 2));
	}
}
```

Output

```
a<<2 = 20
a>>2 = 1
b>>>2 = 1073741821
```

Operator Precedence

Highest			
( )	[ ]	.	
++	– –	~	!
*	/	%	
+	–		
>>	>>>	<<	
>	>=	<	<=
==	!=		
&			
^			
|			
&&			
||			
?:			
=	op=		
Lowest			

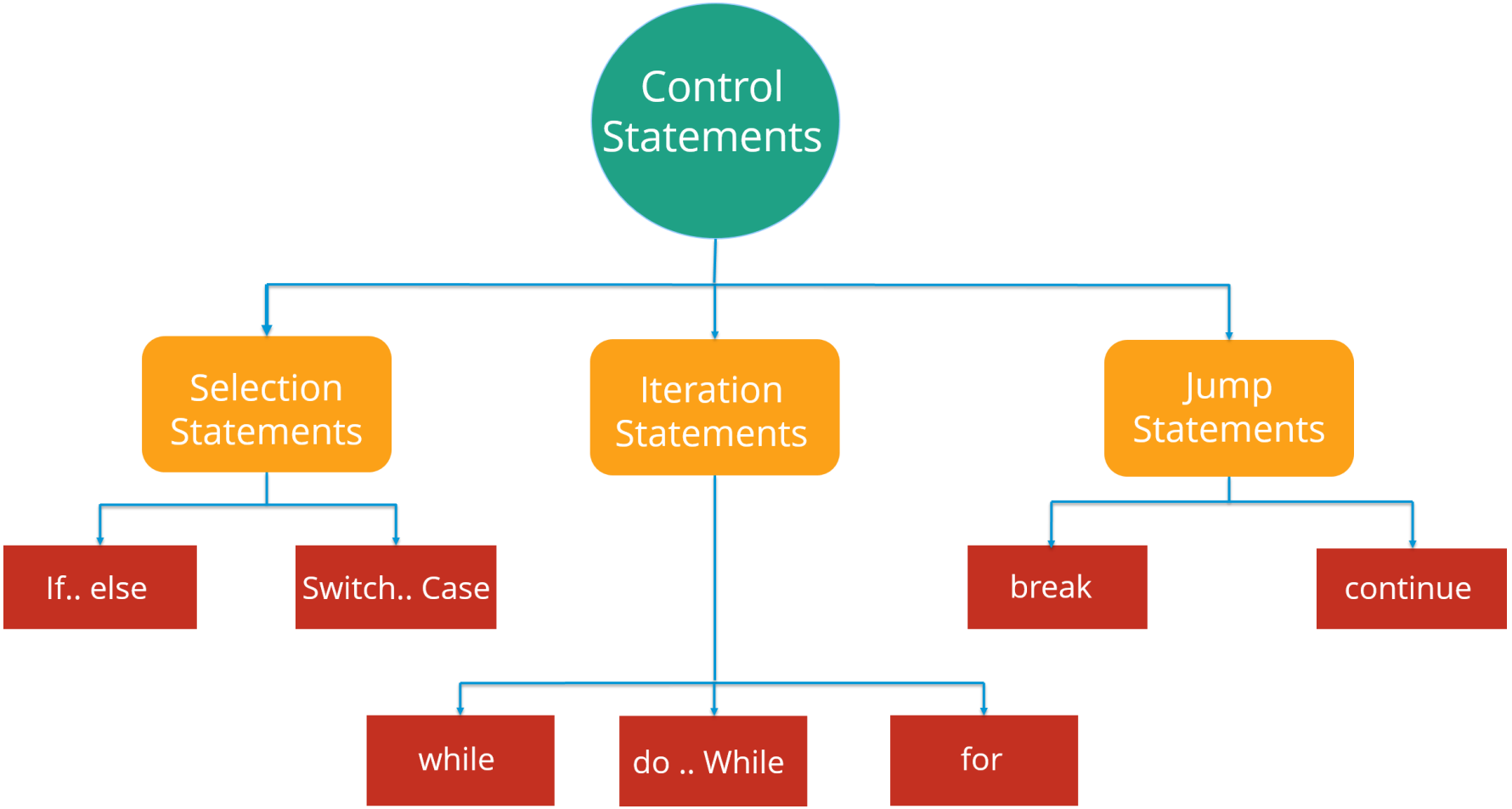
Topic 2

Control Statements

Control Statements

1) **Selection Statements:** Control the flow of the program during run time on the basis of the outcome of an expression or state of a variable
   - If
   - Nested ifs
   - The if-else-if Ladder
   - switch
2) **Iteration Statements:** Also called Loop which iterates through small pieces of code
   - while
   - do-while
   - for
   - For-Each
3) **Jump Statements:** Used to transfer the control to another part of your program
   - break
   - continue
   - return

Control Statements



1) Selection Statement - if

```
if (condition) statement1;
else statement2;
```

Example 2.10. Demonstrate if statement

```
public class Compare {
    int a=10,
    int b=5;

if(a>b)
     {  // if condition
    System.out.println(" A is greater than B");
     }
else
     {     // else condition
     System.out.println(" B is greater");
     }
}
```

1) Selection Statement - if

- Nested ifs

```
if(i == 10) {
  if(j < 20) a = b;
  if(k > 100) c = d; // this if is
  else a = c;        // associated with this else
}
else a = d;          // this else refers to if(i == 10)
```

- The if-else-if ladder

```
if (condition)
   Statement;
else if (condition)
   Statement;
else if (conditon)
   Statement;
else
   Statement;
```

Example 2.11. Demonstrate if-else-if statement

```
class IfElse {
  public static void main(String args[]) {
    int month = 4; // April
    String season;

    if(month == 12 || month == 1 || month == 2)
      season = "Winter";
    else if(month == 3 || month == 4 || month == 5)
      season = "Spring";
    else if(month == 6 || month == 7 || month == 8)
      season = "Summer";
    else if(month == 9 || month == 10 || month == 11)
      season = "Autumn";
    else
      season = "Bogus Month";

    System.out.println("April is in the " + season + ".");
  }
}
```

Output

April is in the Spring.

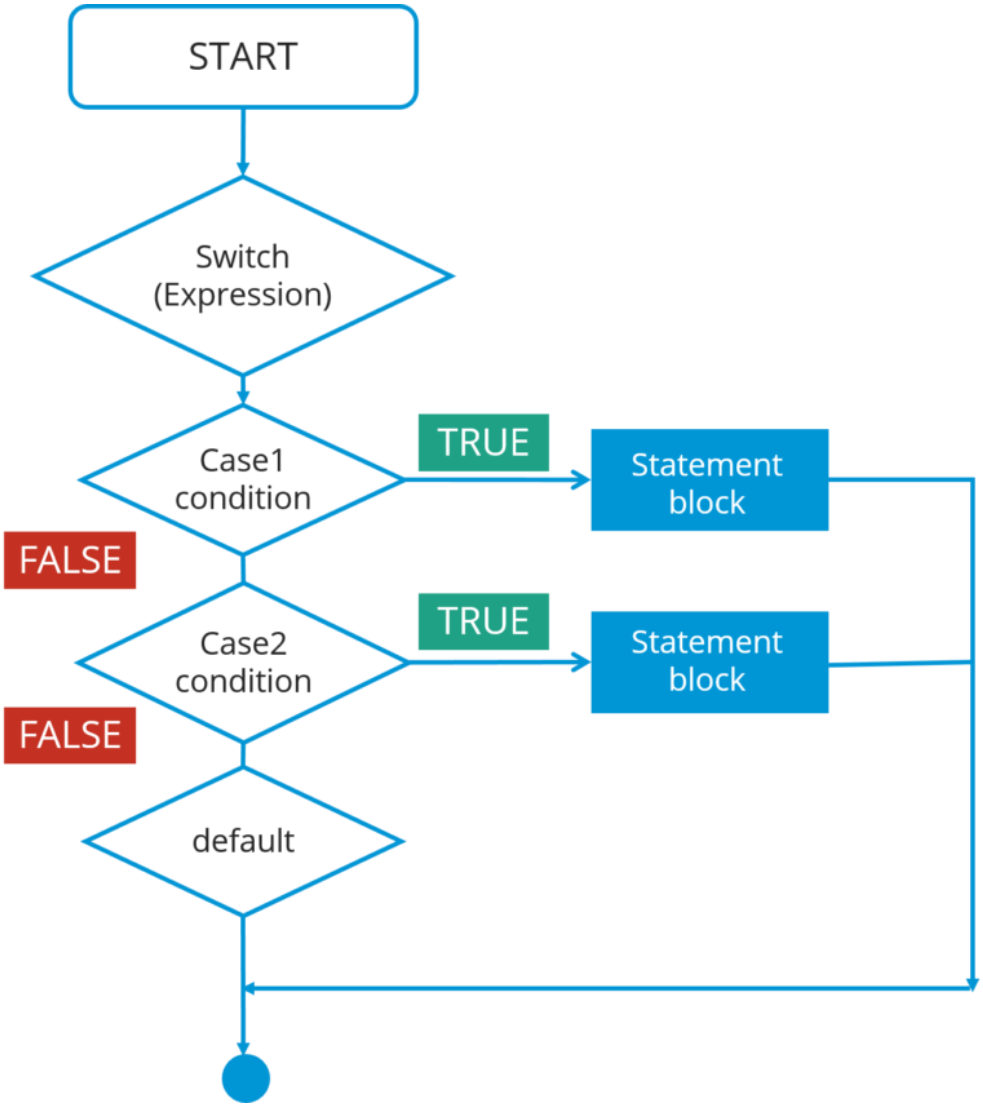
Exercise 2.1: Write a program that prompts the user to enter a weight in pounds and height in inches and displays the BMI. It can be calculated by taking your weight in kilograms and dividing it by the square of your height in meters. Note that one pound is 0.45359237 kilograms and one inch is 0.0254 meters. The interpretation of BMI for people 20 years or older is as follows:

BMI	Interpretation
Below 18.5	Underweight
18.5–24.9	Normal
25.0–29.9	Overweight
Above 30.0	Obese

```java
import java.util.Scanner;

public class ComputeAndInterpretBMI {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

    // Prompt the user to enter weight in pounds
    System.out.print("Enter weight in pounds: ");
    double weight = input.nextDouble();

    // Prompt the user to enter height in inches
    System.out.print("Enter height in inches: ");
    double height = input.nextDouble();

    final double KILOGRAMS_PER_POUND = 0.45359237; // Constant
    final double METERS_PER_INCH = 0.0254; // Constant

    // Compute BMI
    double weightInKilograms = weight * KILOGRAMS_PER_POUND;
    double heightInMeters = height * METERS_PER_INCH;
    double bmi = weightInKilograms /
      (heightInMeters * heightInMeters);
```

```
23
24        // Display result
25        System.out.println("BMI is " + bmi);
26        if (bmi < 18.5)
27          System.out.println("Underweight");
28        else if (bmi < 25)
29          System.out.println("Normal");
30        else if (bmi < 30)
31          System.out.println("Overweight");
32        else
33          System.out.println("Obese");
34      }
35    }
```

1) Selection Statement - switch




```
switch (expression) {
  case value1:
     // statement sequence
     break;
  case value2:
     // statement sequence
     break;
.
.
.
  case valueN:
     // statement sequence
     break;
  default:
    // default statement sequence
}
```

Example 2.12. Demonstrate switch-statement

```
// A simple example of the switch.
class SampleSwitch {
  public static void main(String args[]) {
    for(int i=0; i<6; i++)
      switch(i) {
        case 0:
          System.out.println("i is zero.");
          break;
        case 1:
          System.out.println("i is one.");
          break;
        case 2:
          System.out.println("i is two.");
          break;
        case 3:
          System.out.println("i is three.");
          break;
        default:
          System.out.println("i is greater than 3.");
      }
  }
}
```

Example 2.13. Season program using switch-statement

```
class Switch {
    public static void main(String args[]) {
        int month = 4;
        String season;
        switch (month) {
          case 12:
          case 1:
          case 2:
            season = "Winter";
            break;
          case 3:
          case 4:
          case 5:
            season = "Spring";
            break;
          case 6:
          case 7:
          case 8:
            season = "Summer";
            break;
          case 9:
          case 10:
          case 11:
            season = "Autumn";
            break;
          default:
            season = "Bogus Month";
        }
        System.out.println("April is in the " + season + ".");
    }
}
```

- Exercise 2.2: Write a program that finds out the Chinese Zodiac sign for a given year. The Chinese Zodiac is based on a twelve-year cycle, with each year represented by an animal monkey, rooster, dog, pig, rat, ox, tiger, rabbit, dragon, snake, horse, or sheep—in this cycle, as shown in Figure.

year % 12 =

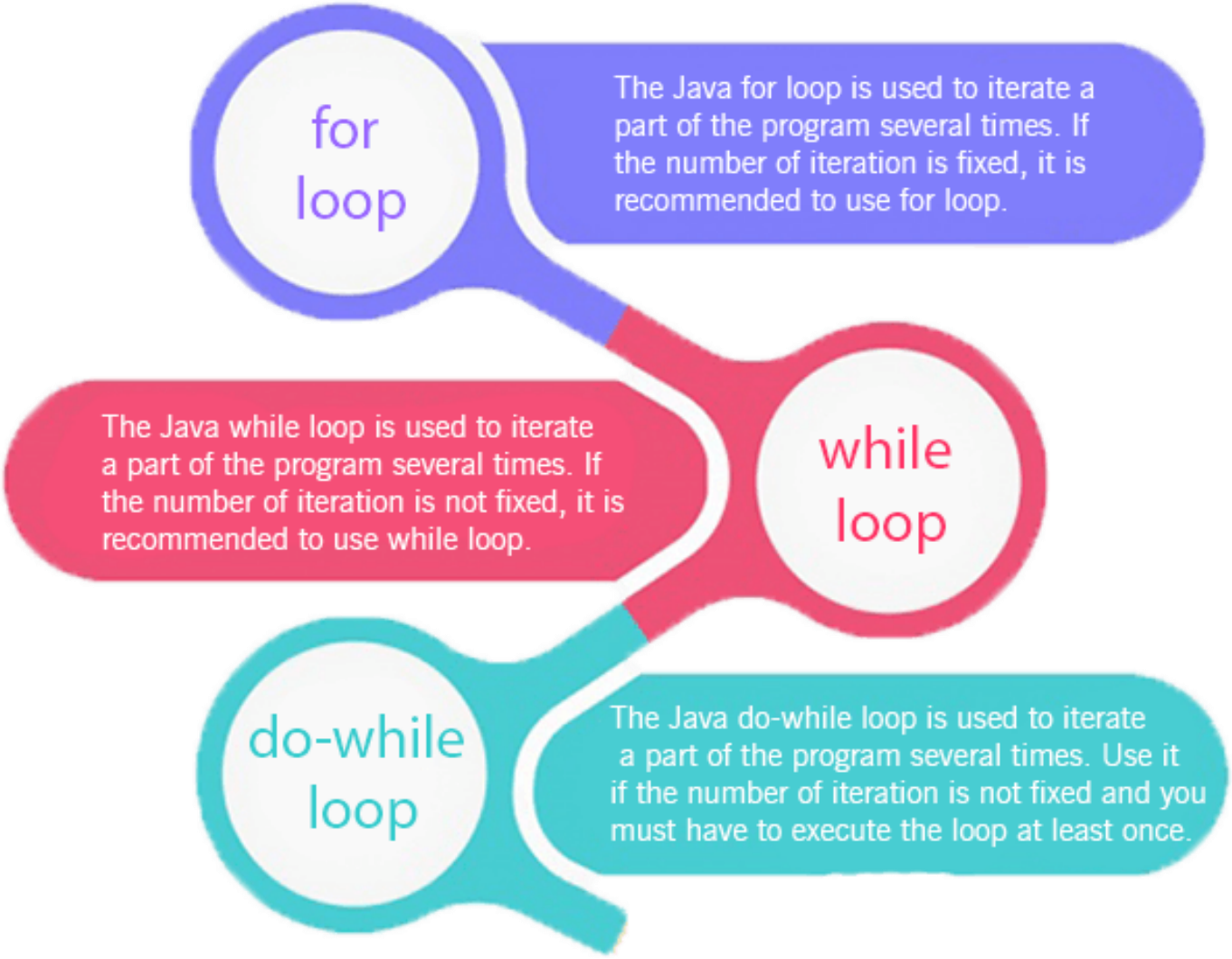
0: monkey
1: rooster
2: dog
3: pig
4: rat
5: ox
6: tiger
7: rabbit
8: dragon
9: snake
10: horse
11: sheep

```java
import java.util.Scanner;

public class ChineseZodiac {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

    System.out.print("Enter a year: ");
    int year = input.nextInt();

    switch (year % 12) {
      case 0: System.out.println("monkey"); break;
      case 1: System.out.println("rooster"); break;
      case 2: System.out.println("dog"); break;
      case 3: System.out.println("pig"); break;
      case 4: System.out.println("rat"); break;
      case 5: System.out.println("ox"); break;
      case 6: System.out.println("tiger"); break;
      case 7: System.out.println("rabbit"); break;
      case 8: System.out.println("dragon"); break;
      case 9: System.out.println("snake"); break;
      case 10: System.out.println("horse"); break;
      case 11: System.out.println("sheep");
    }
  }
}
```

Output

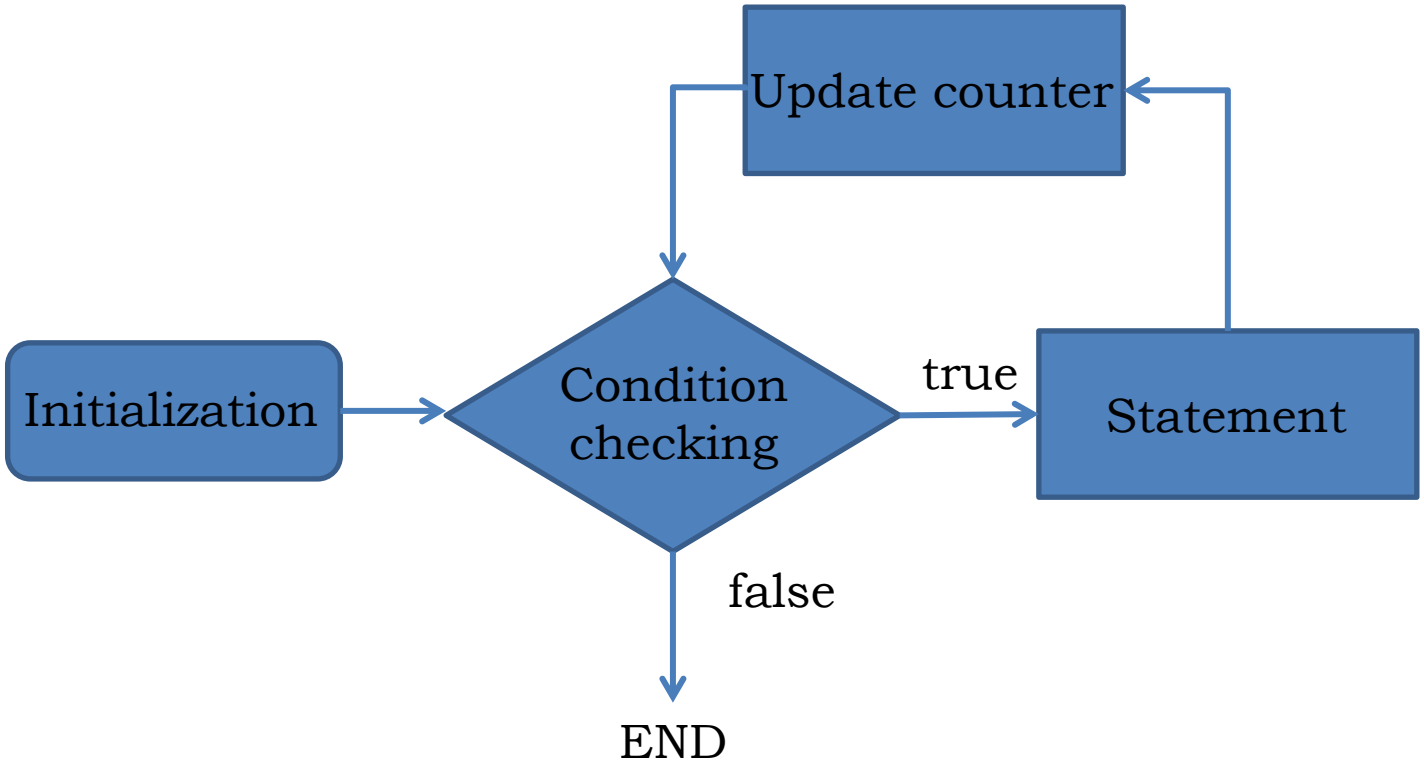
```
Enter a year: 1984
rat
```

2) Iteration statements/ Loops



2) Loops - for

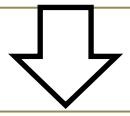
```
for(initialization; condition; iteration) {
   // body
}
```



2) Loops: for-each

for(*type itr-var : collection*) *statement-block*

```
int nums[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
int sum = 0;

for(int i=0; i < 10; i++) sum += nums[i];
```



```
int nums[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
int sum = 0;

for(int x: nums) sum += x;
```

Example 2.14. Demonstrate for-each

```
public class forEach
{
        public static void main(String args[])
        {
        String array[] = {"Ron", "Harry", "Hermoine"};
        //enhanced for loop
        for (String x:array)
        {
        System.out.println(x);
        }

        }
}
```

```
        /*for loop for same function
        for (int i = 0; i < array.length; i++)
        {
        System.out.println(array[i]);
        }
        */
```

Output

Ron
Harry
Hermoine

Example 2.15. Demonstrate the use of for-each for two dimensional array

```
// Use for-each style for on a two-dimensional array.
class ForEach3 {
  public static void main(String args[]) {
    int sum = 0;
    int nums[][] = new int[3][5];

    // give nums some values
    for(int i = 0; i < 3; i++)

     for(int j=0; j < 5; j++)
       nums[i][j] = (i+1)*(j+1);

   // use for-each for to display and sum the values
   for(int x[] : nums) {
     for(int y : x) {
       System.out.println("Value is: " + y);
       sum += y;
     }
   }
   System.out.println("Summation: " + sum);
  }
}
```

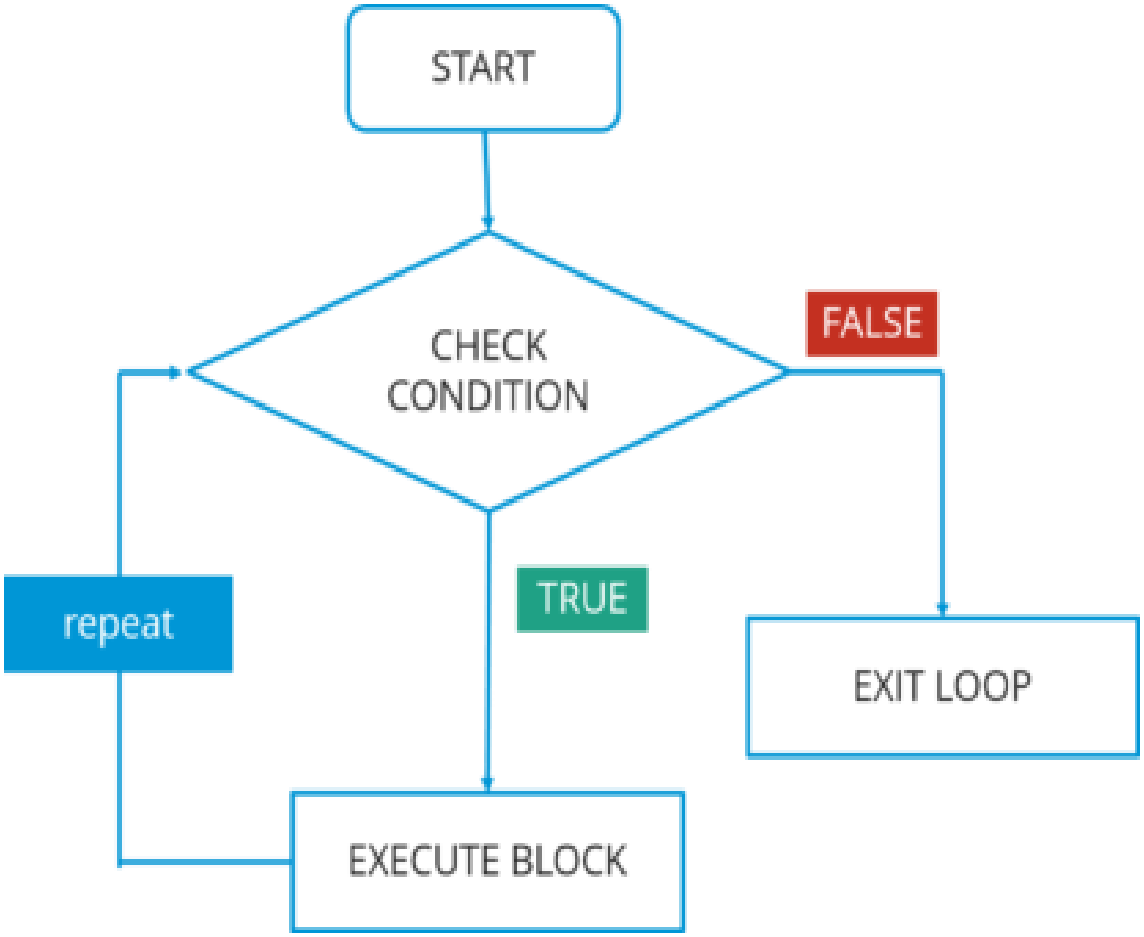
Output

```
Value is: 1
Value is: 2
Value is: 3
Value is: 4
Value is: 5
Value is: 2
Value is: 4
Value is: 6
Value is: 8
Value is: 10
Value is: 3
Value is: 6
Value is: 9
Value is: 12
Value is: 15
Summation: 90
```

Nested For Loop

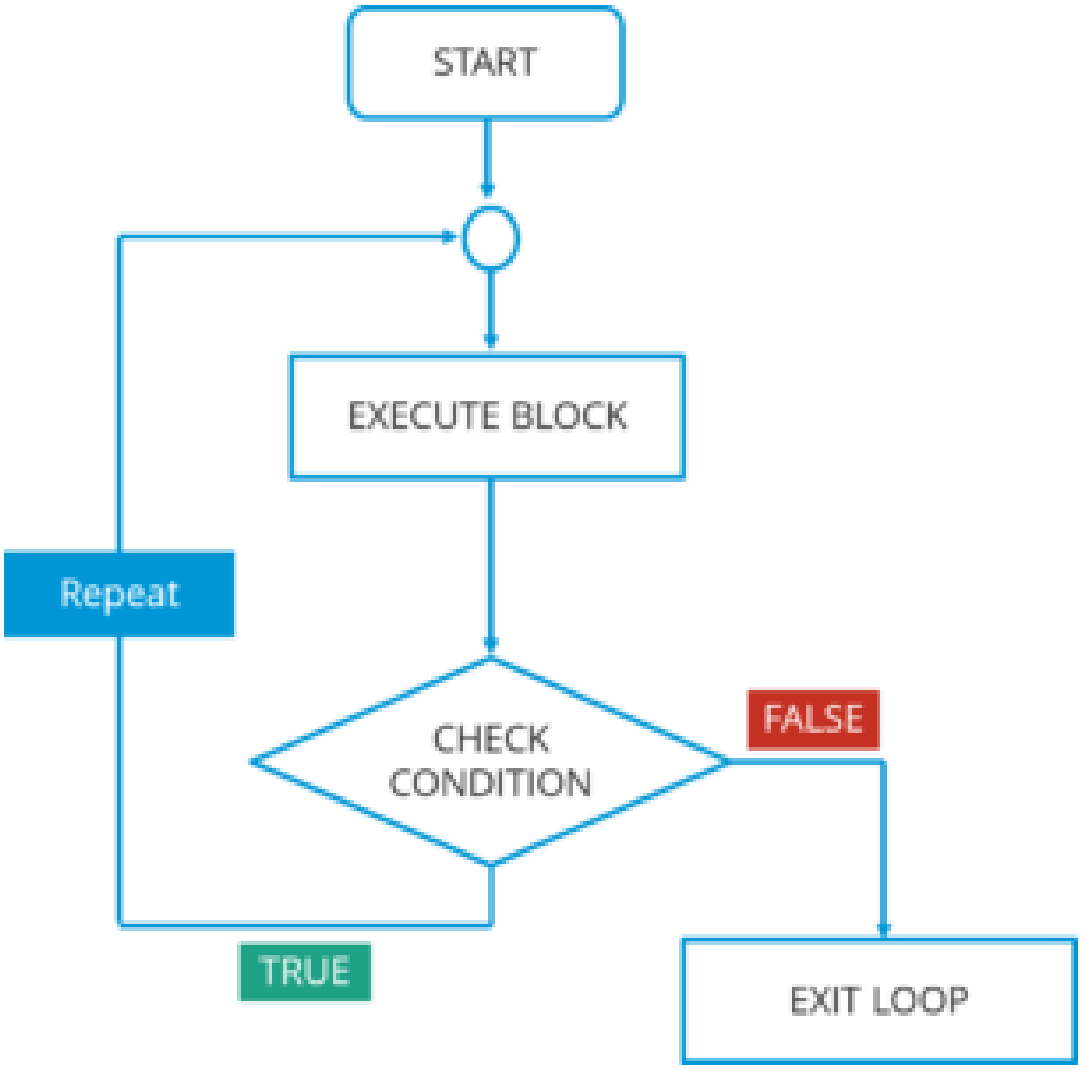
```
// Loops may be nested.
class Nested {
  public static void main(String args[]) {
    int i, j;

    for(i=0; i<10; i++) {
      for(j=i; j<10; j++)
        System.out.print(".");
      System.out.println();
    }
  }
}
```

2) Loops - while




```
while (condition) {
    // body of loop
}
```

2) Loops: do-while




```
do {
  // body of loop
} while (condition);
```

Exercise 2.3: Write a program that generates multiplication table of numbers 1 to 9.

```
                Multiplication Table
          1   2   3   4   5   6   7   8   9
-------------------------------------------
1 |       1   2   3   4   5   6   7   8   9
2 |       2   4   6   8  10  12  14  16  18
3 |       3   6   9  12  15  18  21  24  27
4 |       4   8  12  16  20  24  28  32  36
5 |       5  10  15  20  25  30  35  40  45
6 |       6  12  18  24  30  36  42  48  54
7 |       7  14  21  28  35  42  49  56  63
8 |       8  16  24  32  40  48  56  64  72
9 |       9  18  27  36  45  54  63  72  81
```

```
1  public class MultiplicationTable {
2    /** Main method */
3    public static void main(String[] args) {
4      // Display the table heading
5      System.out.println("           Multiplication Table");
6
7      // Display the number title
8      System.out.print("    ");
9      for (int j = 1; j <= 9; j++)
10       System.out.print("   " + j);
11
12     System.out.println("\n----------------------------------------");
13
14     // Display table body
15     for (int i = 1; i <= 9; i++) {
16       System.out.print(i + " | ");
17       for (int j = 1; j <= 9; j++) {
18         // Display the product and align properly
19         System.out.printf("%4d", i * j);
20       }
21       System.out.println();
22     }
23   }
24 }
```

- Exercise 2.4: Write a program that prompts the user to enter an answer for a question on addition of two single digits and let the user repeatedly enter a new answer until it is correct.

```
import java.util.Scanner;

public class repeatanswer {
public static void main(String[] args) {
int number1 = 8679;
int number2 = 43445;

// Create a Scanner
Scanner input = new Scanner(System.in);
System.out.print(
"What is " + number1 + " + " + number2 + "? ");
int answer = input.nextInt();
while(answer != number1+number2) {
System.out.print("Wrong answer. Try again. What is "
+ number1 + " + " + number2 + "? ");
answer = input.nextInt();
}
System.out.println("You got it!");
}
}
```

Example 2.15. Demonstrate do-while loop

```
// Demonstrate the do-while loop.
class DoWhile {
  public static void main(String args[]) {
    int n = 10;

    do {
      System.out.println("tick " + n);
      n--;
    } while(n > 0);
  }
}
```

Output

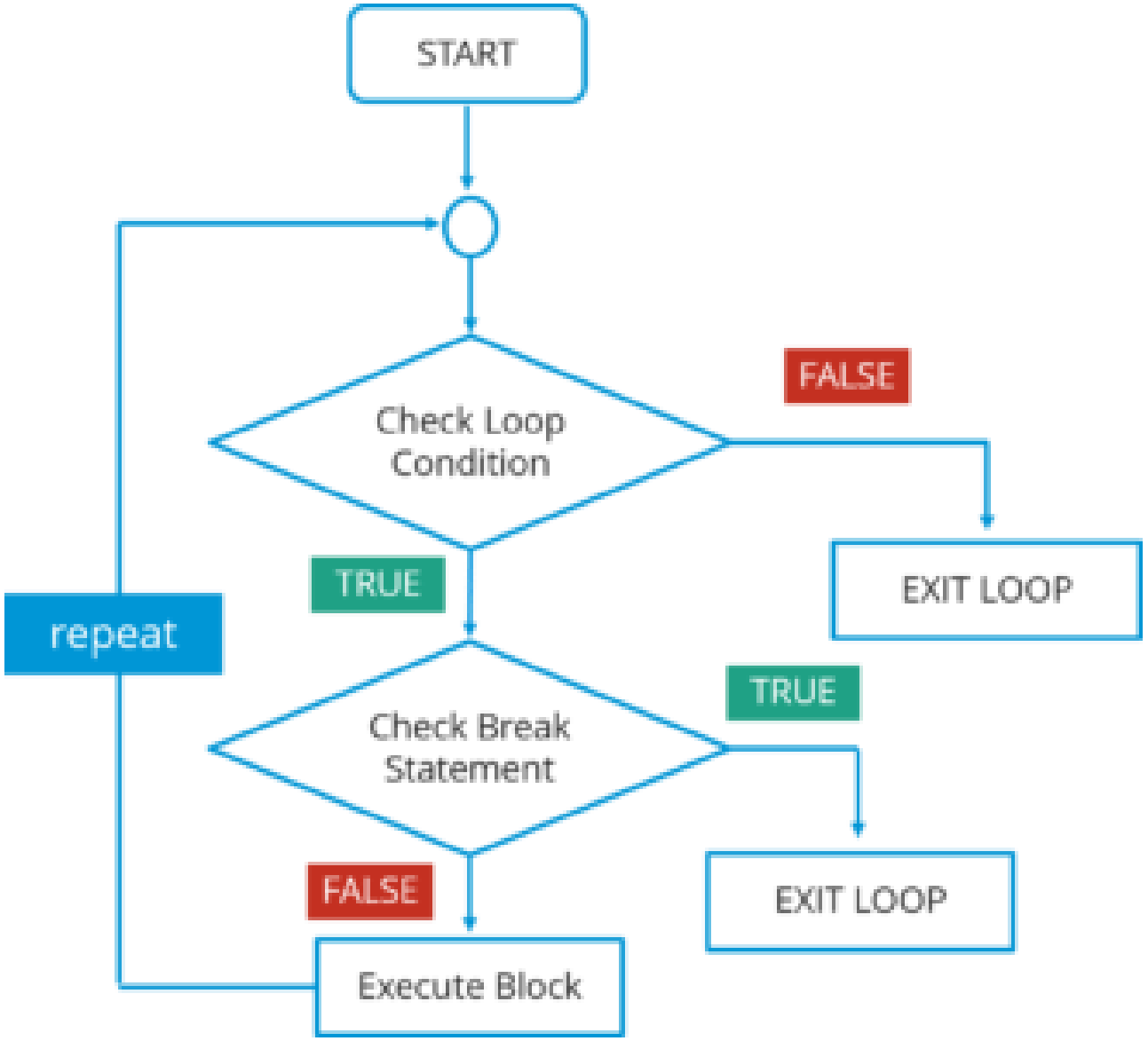
```
tick 10
tick 9
tick 8
tick 7
tick 6
tick 5
tick 4
tick 3
tick 2
tick 1
```

3) Jump Statements

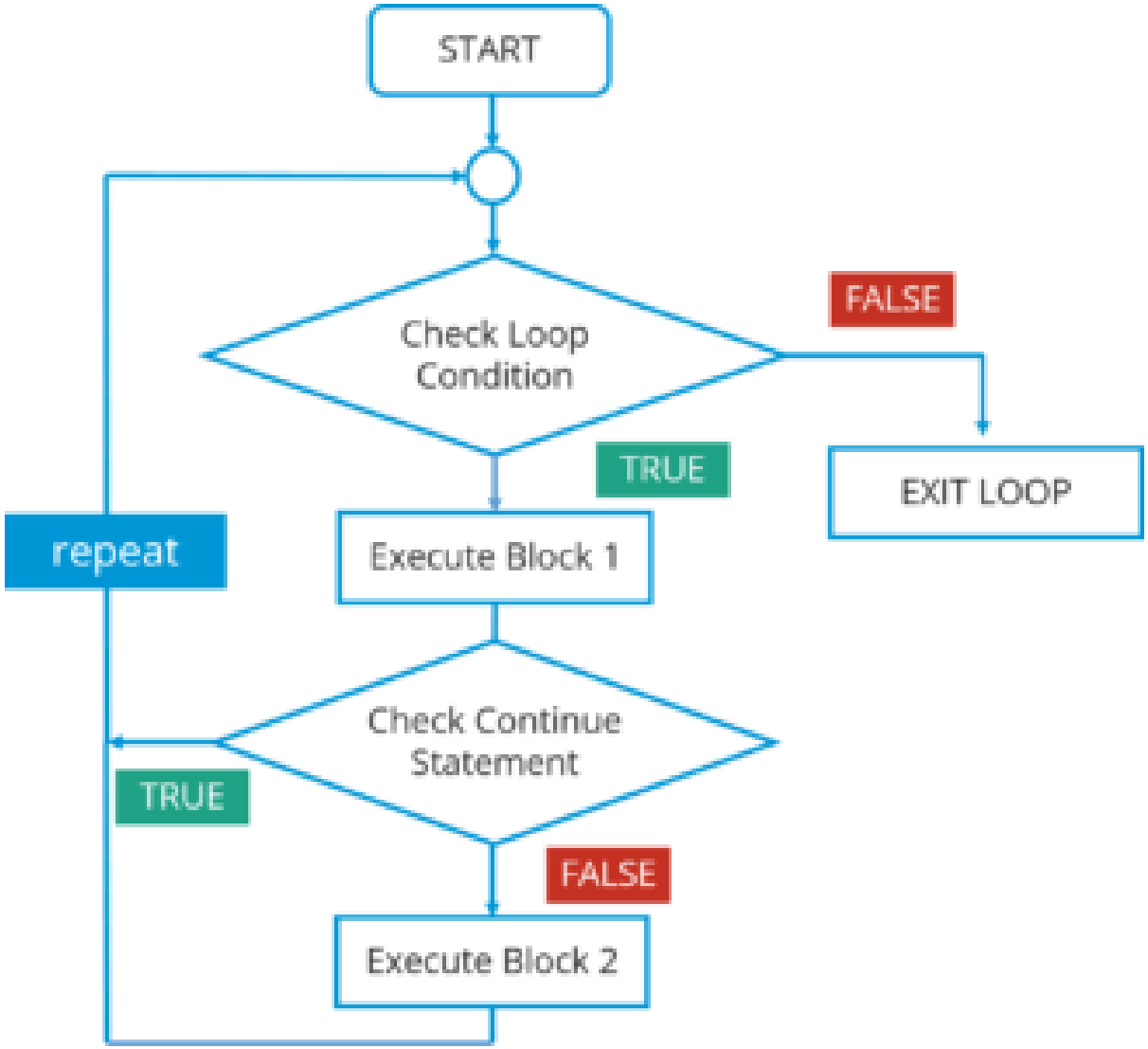
- **break**
  - To terminates a statement sequence in a **switch** statement
  - To be used to exit a loop.
  - To use as a “civilized” form of goto

    e.g., break *Label*; // exit from the *Label* block
- **continue**
  - To continue running the loop but stop processing the remainder of the code in its body for the particular iteration
- **return**
  - To explicitly return from a method

3) Jump Statements - Break



3) Jump Statements - Continue



- **Example 2.16. Using break to exit a loop**

```
class BreakLoopDemo
{
    public static void main(String args[])
    {
        // Initially loop is set to run from 0-9
        for (int i = 0; i < 10; i++)
        {
        // terminate loop when i is 5.
        if (i == 5)
        break;
        System.out.println("i: " + i);
        }
        System.out.println("Loop complete.");
    }
}
```

Output

```
i: 0
i: 1
i: 2
i: 3
i: 4
Loop complete.
```

- **Example 2.17. Using break to exit a nested loop**

```
// Using break with nested loops.
class BreakLoop3 {
  public static void main(String args[]) {
    for(int i=0; i<3; i++) {
      System.out.print("Pass " + i + ": ");
      for(int j=0; j<100; j++) {
        if(j == 10) break; // terminate loop if j is 10
        System.out.print(j + " ");
      }
      System.out.println();
    }
    System.out.println("Loops complete.");
  }
}
```

Output

```
Pass 0: 0 1 2 3 4 5 6 7 8 9
Pass 1: 0 1 2 3 4 5 6 7 8 9
Pass 2: 0 1 2 3 4 5 6 7 8 9
Loops complete.
```

Example 2.18. Using break a Goto

```
// Using break as a civilized form of goto.
class Break {
  public static void main(String args[]) {
    boolean t = true;

    first: {
      second: {
        third: {
          System.out.println("Before the break.");
          if(t) break second; // break out of second block
          System.out.println("This won't execute");
        }
        System.out.println("This won't execute");
      }
      System.out.println("This is after second block.");
    }
  }
}
```

Output

```
Before the break.
This is after second block.
```

- **Example 2.19. Using break to exit from nested loops**

```
// Using break to exit from nested loops
class BreakLoop4 {
  public static void main(String args[]) {
    outer: for(int i=0; i<3; i++) {
      System.out.print("Pass " + i + ": ");
      for(int j=0; j<100; j++) {
        if(j == 10) break outer; // exit both loops
        System.out.print(j + " ");
      }
      System.out.println("This will not print");
    }
    System.out.println("Loops complete.");
  }
}
```

Output

Pass 0: 0 1 2 3 4 5 6 7 8 9 Loops complete.

- **Example 2.20. Demonstrate the use of continue**

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

Output

```
0 1
2 3
4 5
6 7
8 9
```

- **Example 2.21. Demonstrate the use of continue**

```
class ContinueLabel {
  public static void main(String args[]) {
outer: for (int i=0; i<10; i++) {
         for(int j=0; j<10; j++) {
           if(j > i) {
             System.out.println();
             continue outer;
           }
           System.out.print(" " + (i * j));
         }
       }
       System.out.println();
  }
}
```

```
0
0 1
0 2 4
0 3 6 9
0 4 8 12 16
0 5 10 15 20 25
0 6 12 18 24 30 36
0 7 14 21 28 35 42 49
0 8 16 24 32 40 48 56 64
0 9 18 27 36 45 54 63 72 81
```

- **Example 2.22. Demonstrate the use of return**

```
class Return
{
    public static void main(String args[])
    {
        boolean t = true;
        System.out.println("Before the return.");
        if (t)
        return;
        // Compiler will bypass every statement after return
        System.out.println("This won't execute.");
    }
}
```

Output

Before the return.

Practical Assignments

1. Write a program that lets the user enter a year and checks whether it is a leap year. A year is a leap year if it is divisible by 4 but not by 100 , or if it is divisible by 400.
2. Write a program that randomly generates a lottery of a two-digit number, prompts the user to enter a two-digit number, and determines whether the user wins according to the following rules:

   1. If the user input matches the lottery number in the exact order, the award is $10,000.

   2. If all the digits in the user input match all the digits in the lottery number, the award is $3,000.

   3. If one digit in the user input matches a digit in the lottery number, the award is $1,000.
3. Use nested loops that display the following patterns in four separate programs.

```
Pattern A          Pattern B          Pattern C          Pattern D
1                  1 2 3 4 5 6                1          1 2 3 4 5 6
1 2                1 2 3 4 5                2 1            1 2 3 4 5
1 2 3              1 2 3 4                3 2 1              1 2 3 4
1 2 3 4            1 2 3                4 3 2 1                1 2 3
1 2 3 4 5          1 2                5 4 3 2 1                  1 2
1 2 3 4 5 6        1                6 5 4 3 2 1                    1
```

Next Week Lecture

Classes and A closer look at Methods and Classes

Thank you!