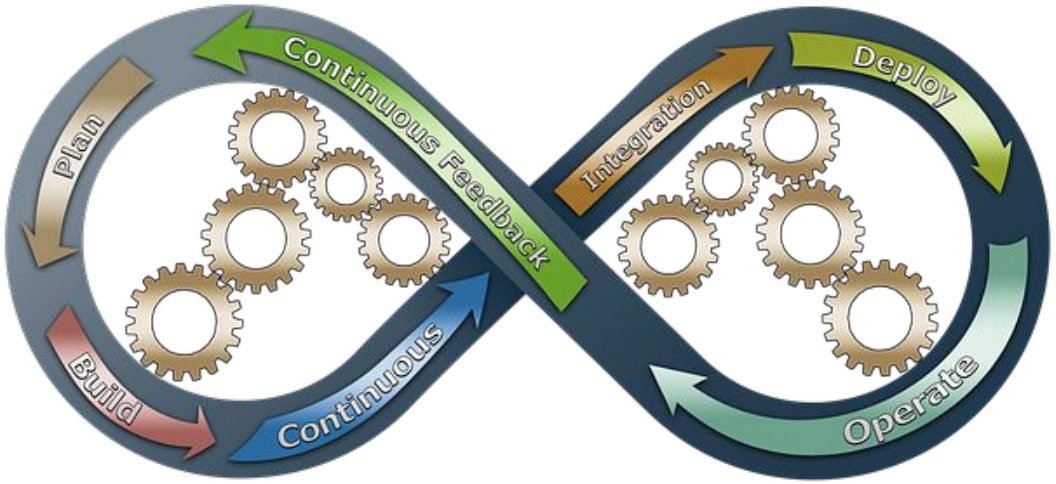


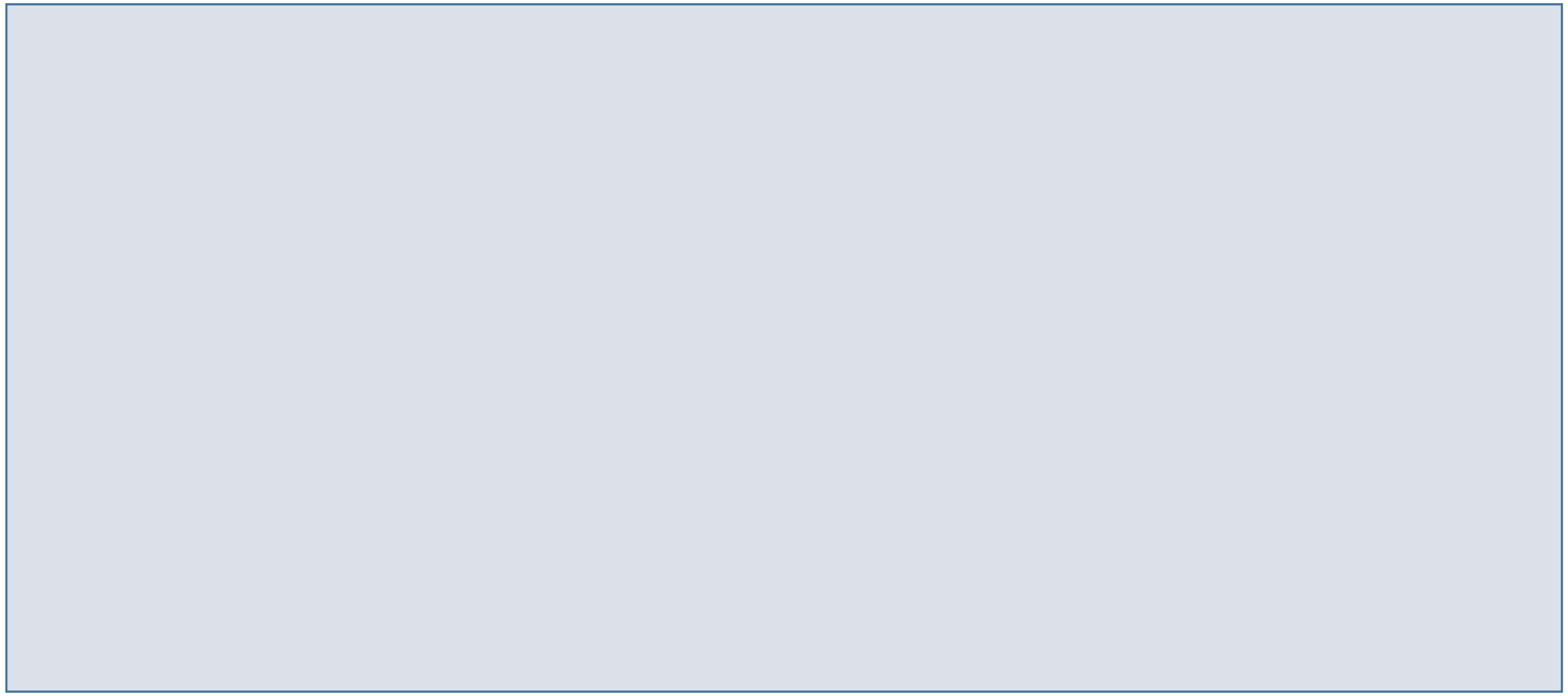
Session 3.4

Operators
&
Control Flow

AN INITIATIVE BY

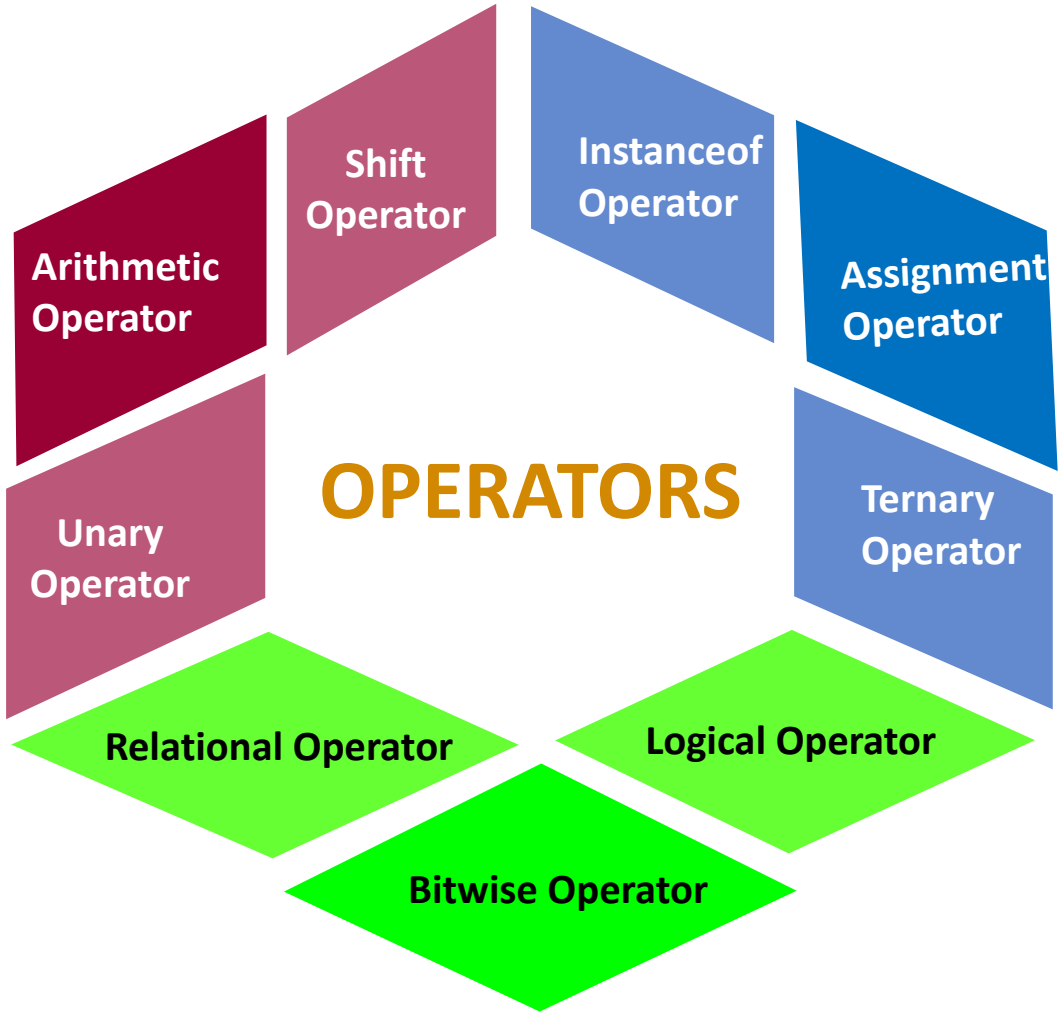
UNICAL ACADEMY

Introduction



Let’s go!!!

Operators



UNICAL ACADEMY

Arithmetic Operator

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo Operation (Remainder after division)

Example:

```
class Main {
        public static void main(String[] args) {
          // declare variables
           int a = 12, b = 5;
            // addition operator
            System.out.println("a + b = " + (a + b));
           // subtraction operator
            System.out.println("a - b = " + (a - b));
          // multiplication operator
           System.out.println("a * b = " + (a * b));
          // division operator
           System.out.println("a / b = " + (a / b));
          // modulo operator
          System.out.println("a % b = " + (a % b));
                }
            }
```

UNICAL ACADEMY

Unary Operators

Operator	Description
+	It is used to represent the **positive** value
-	It is used to represent the **negative** value
++var	Pre-increment
var++	Post-increment
--var	Pre-decrement
var--	Post-decrement

Example:

```
class Operator  {
    public static void main(String[] args)  {
                int var1 = 5, var2 = 5;
                // 5 is displayed
                // Then, var1 is increased to 6.
                System.out.println(var1++);
                // var2 is increased to 6
                // Then, var2 is displayed
                System.out.println(++var2);
                  }
              }
```

UNICAL ACADEMY

Equality and Relational Operators

Operator	Description
==	Is Equal To
!=	Not Equal To
>	Greater Than
<	Less Than
>=	Greater Than or Equal To
<=	Less Than or Equal To

Example:

```
class Main {
    public static void main(String[] args) {
      // create variables
      int a = 7, b = 11;
     // value of a and b
      System.out.println("a is " + a + " and b is " + b);
      // == operator
      System.out.println(a == b); // false
      // != operator
      System.out.println(a != b); // true
     // > operator
      System.out.println(a > b); // false
     // < operator
      System.out.println(a < b); // true
     // >= operator
      System.out.println(a >= b); // false
     // <= operator
      System.out.println(a <= b); // true
            }
       }
```

UNICAL ACADEMY

Logical Operators

Operator	Description
&& (Logical AND)	**true** only if both expression1 and expression2 are true
|| (Logical OR)	**true** if either expression1 or expression2 is true
! (Logical NOT)	**true** if expression is false and vice versa

Example:

```
class Main {
      public static void main(String[] args) {
       // && operator
        System.out.println((5 > 3) && (8 > 5)); // true
        System.out.println((5 > 3) && (8 < 5)); // false
       // || operator
       System.out.println((5 < 3) || (8 > 5)); // true
       System.out.println((5 > 3) || (8 < 5)); // true
       System.out.println((5 < 3) || (8 < 5)); // false
       // ! operator
       System.out.println(!(5 == 3)); // true
       System.out.println(!(5 > 3)); // false
            }
       }
```

UNICAL ACADEMY

Bitwise Operators

Operator	Description
|	Bitwise OR
&	Bitwise AND
^	Bitwise XOR
~	Bitwise complement

Examples:

Bitwise OR

```
class Main {
public static void main(String[] args) {
int number1 = 12, number2 = 25, result; //
bitwise OR between 12 and 25
result = number1 | number2;
System.out.println(result); // prints 29
}
}
```

Bitwise AND

```
class Main {
 public static void main(String[] args) {
 int number1 = 12, number2 = 25, result;
// bitwise AND between 12 and 25
 result = number1 & number2;
System.out.println(result); // prints 8
 }
}
```

Bitwise XOR

```
class Main {
public static void main(String[] args) {
 int number1 = 12, number2 = 25, result;
// bitwise XOR between 12 and 25
result = number1 ^ number2;
System.out.println(result); // prints 21
 }
}
```

Bitwise complement

```
class Main {
 public static void main(String[] args) {
 int number = 35, result;
// bitwise complement of 35
result = ~number;
System.out.println(result); // prints -36
 }
}
```

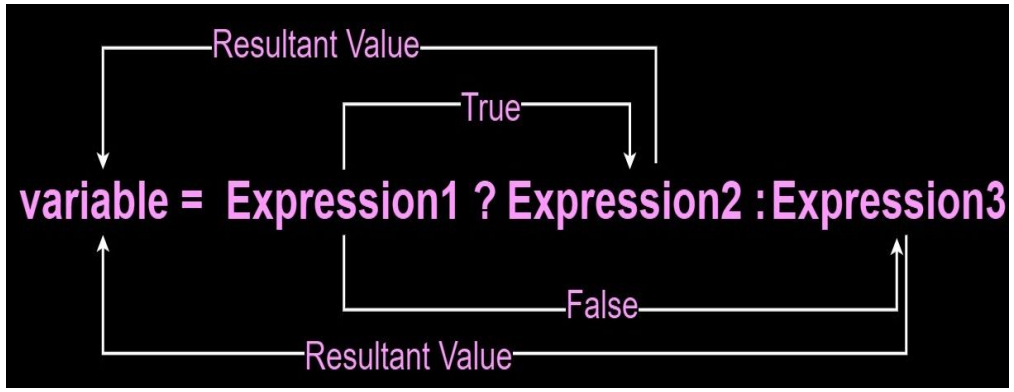
UNICAL ACADEMY

Ternary Operators

- if Expression1 is true, Expression2 is executed.
- And, if Expression1 is false, Expression3 is executed

Syntax:

```
variable = Expression1 ? Expression2: Expression3
```



Example:

```
import java.util.Scanner;

class Main {
    public static void main(String[] args) {
     // take input from users
     Scanner input = new Scanner(System.in);
     System.out.println("Enter your marks: ");
     double marks = input.nextDouble();
     // ternary operator checks if
    // marks is greater than 40
     String result = (marks > 40) ? "pass" : "fail";
     System.out.println("You " + result + " the exam.");
     input.close();
        }
     }
```

UNICAL ACADEMY

Assignment Operators

Operator	Description
=	a = b
+=	a = a + b;
-=	a = a - b;
*=	a = a * b;
/=	a = a / b;
%=	a = a % b;

Example:

```
class Main {
        public static void main(String[] args) {
        // create variables
        int a = 4; int var;
        // assign value using =
        var = a; System.out.println("var using =: " + var);
        // assign value using =+
        var += a; System.out.println("var using +=: " + var);
        // assign value using =*
        var *= a; System.out.println("var using *=: " + var);
                }
         }
```

Instanceof Operator

Syntax:

```
objectName instanceOf className;
```

Examples:

```
class Main {
        public static void main(String[] args) {
          // create a variable of string type
          String name = “UnicalAcademy";
          // checks if name is instance of String
          boolean result1 = name instanceof String;
          System.out.println("name is an instance of String: " + result1);
          // create an object of Main
          Main obj = new Main();
          // checks if obj is an instance of Main
          boolean result2 = obj instanceof Main;
          System.out.println("obj is an instance of Main: " + result2);
                }
            }
```

UNICAL ACADEMY

Shift Operators

Operator	Description
<<	Left Shift
>>	Signed Right Shift
>>>	Unsigned Right Shift

Examples:

Left Shift Operator

```
class Main {
public static void main(String[] args) {
int number = 2; // 2 bit left shift operation
 int result = number << 2;
System.out.println(result); // prints 8
 }
}
```

Signed Right Shift Operator

```
class Main {
public static void main(String[] args) {
 int number1 = 8;
int number2 = -8; // 2 bit signed right shift
System.out.println(number1 >> 2); // prints 2
System.out.println(number2 >> 2); // prints -2
}
 }
```

Unsigned Right Shift Operator

```
class Main {
public static void main(String[] args) {
 int number1 = 8;
int number2 = -8; // 2 bit signed right shift
System.out.println(number1 >>> 2); // prints 2
System.out.println(number2 >>> 2); // prints 1073741822
}
 }
```

Type Casting

- **Widening Casting** (automatically) - converting a smaller type to a larger type size
  byte -> short -> char -> int -> long -> float -> double

- **Narrowing Casting** (manually) - converting a larger type to a smaller size type
  double -> float -> long -> int -> char -> short -> byte

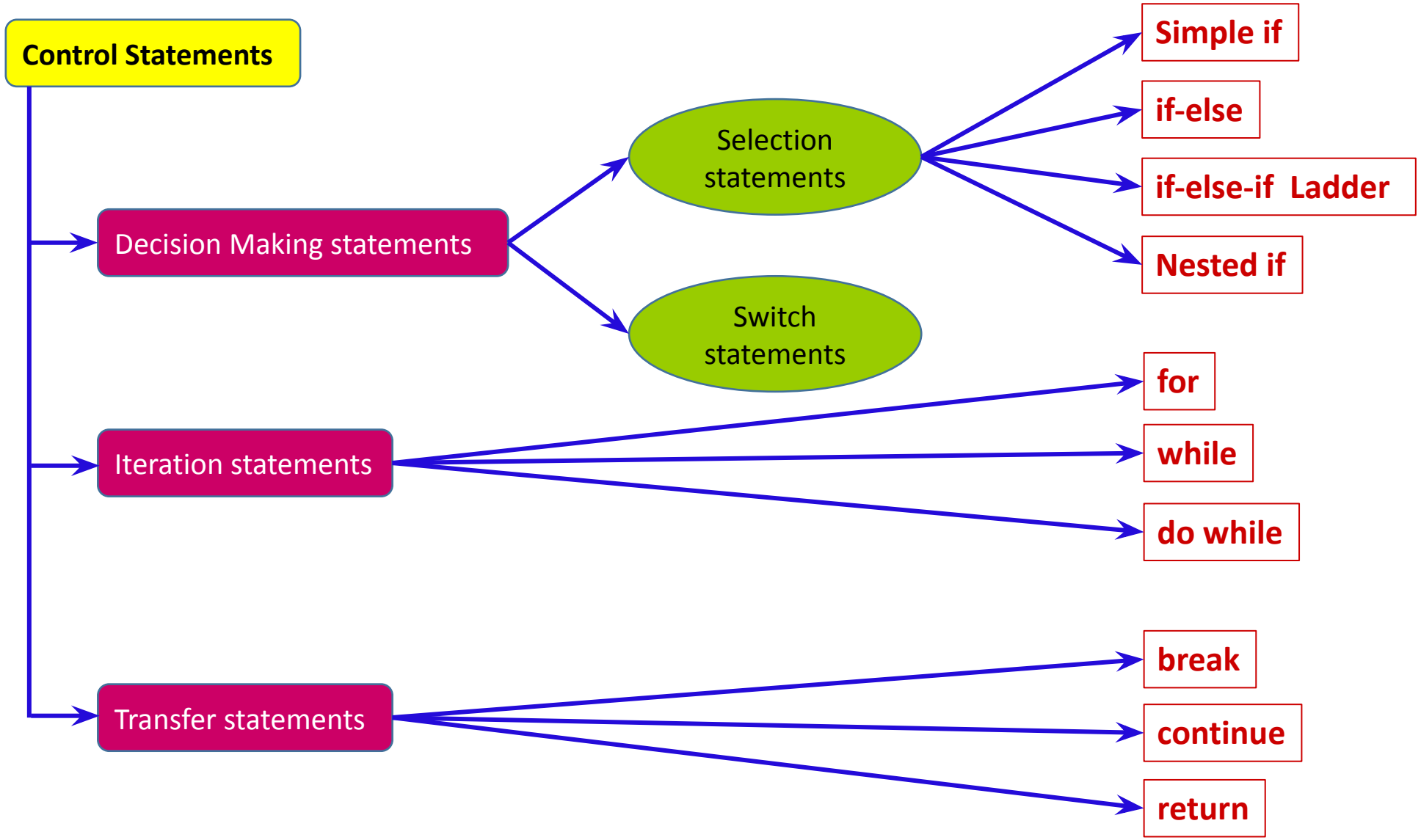
Widening Casting Example:

```
public class Main {
public static void main(String[] args) {
int myInt = 9; double myDouble = myInt;
// Automatic casting: int to double
System.out.println(myInt);  // Outputs 9
System.out.println(myDouble);  // Outputs 9.0
}
}
```

Narrowing Casting Example:

```
public class Main {
 public static void main(String[] args) {
double myDouble = 9.78;
int myInt = (int) myDouble;
// Manual casting: double to int
System.out.println(myDouble);  // Outputs 9.78
System.out.println(myInt);  // Outputs 9
 }
}
```

Control Flow



UNICAL ACADEMY

Selection Statements

Simple if Syntax:

```
if (condition) {
// statements
 }
```

if-else Syntax:

```
if (condition) {
// statements in if block
 }
 else {
 // statements in else block
}
```

if-else-if Ladder Syntax:

```
if (condition1) {
// statements}
else if(condition2) {
// statements
}
else if (condition3) {
 // statements
}
.
 .
else {
//Statements
}
```

Nested if Syntax:

```
if (condition1)
  {
  if (condition2)
    {
    // statements
    }
   else
    {
     // statements
    }
    }
else
 {
  if (condition3)
    {
     // statements
    }
   else
   {
    // statements
   }
}
```

Switch Syntax:

```
switch (expression) {
case value1:
// code
break;
case value2:
// code
break;
...
...
default:
// default statements
 }
```

UNICAL ACADEMY

Iteration Statements

for Syntax:

```
for (initialExpression; testExpression; updateExpression)
{
// body of the loop
}
```

while Syntax:

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

do while Syntax:

```
do {
 // body of loop
}
while(textExpression)
```

UNICAL ACADEMY

Transfer Statements

break Syntax:

```
while (testExpression) {
    // codes
    if (condition to break) {
        break;
    }
    // codes
}
```

```
do {
    // codes
    if (condition to break) {
        break;
    }
    // codes
}
while (testExpression);
```

```
for (init; testExpression; update) {
    // codes
    if (condition to break) {
        break;
    }
    // codes
}
```

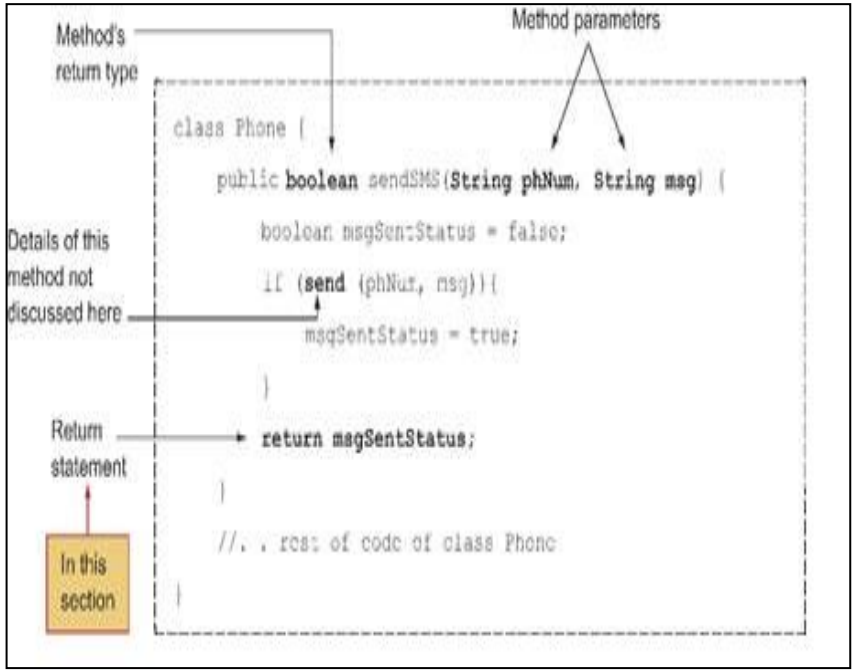
continue Syntax:

```
while (testExpression) {
    // codes
    if (testExpression) {
        continue;
    }
    // codes
}
```

```
do {
    // codes
    if (testExpression) {
        continue;
    }
    // codes
}
while (testExpression);
```

```
for (init; testExpression; update) {
    // codes
    if (testExpression) {
        continue;
    }
    // codes
}
```

return Syntax:



UNICAL ACADEMY

Session Recap