JAVA PROGRAMMING BASICS

Module 1: Java Overview

Training program

1. Java Fundamentals
2. Start programming with Java, create simple console application
3. Classification of Data Types
4. Primitive types in java
5. Control Flow Statements
6. Arrays

Module contents

- Control Flow Statements
  - Identifiers and Literals
  - Local variables: initialization and lifetime
  - Declaring a Variable as a Constant
  - The if-then and if-then-else statements
  - The switch statement
  - Loops: the while, do-while and for statements
  - The break and continue statements
  - The goto keyword
  - Program exit

Keywords in the Java programing language

abstract	double	int	provides…with♦♦	throws
assert***	else	interface	public	transitive♦♦
boolean	enum♦	long	record■	transient
break	extends	module♦♦	requires♦♦	true
byte	false	native	return	try
case	final	non-sealed■■	sealed■■	uses♦♦
catch	finally	null	short	var♦♦
char	float	new	static	void
class	for	open♦♦	strictfp**+*	volatile
const*	goto*	opens…to♦♦	super	while
continue	if	package	switch	yield■
exports♦♦	implements	permits■■	synchronized	
default	import	private	this	
do	instanceof	protected	throw	

* not used, ** 1.2 added, *** 1.4 added, ♦ 5 added, ♦ ♦ 9 added, ■14 added, ■ ■15 added

Module contents

- Control Flow Statements
  - Identifiers and Literals
  - Local variables: initialization and lifetime
  - Declaring a Variable as a Constant
  - The if-then and if-then-else statements
  - The switch statement
  - Loops: the while, do-while and for statements
  - The break and continue statements
  - The goto keyword
  - Program exit

Declaring a Variable as a Constant

- Constant represent permanent data that will never change
- To declare a constant need to use the **final** keyword
- Java constants should be named using uppercase letters with underscore characters as separators
- As a rule, the Java constants are declared with public and static modifiers also at class level

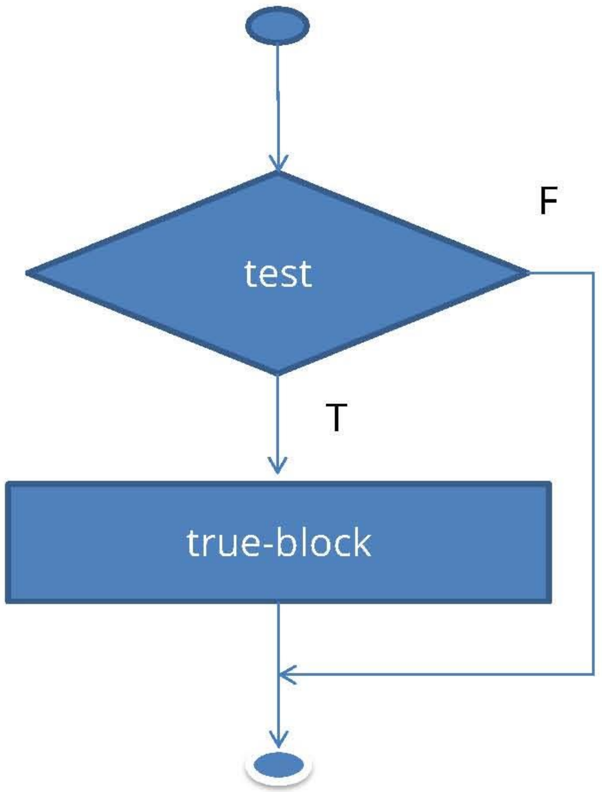
For example:

```
public static final double PI = 3.14159;
public static final int DAY_HOURS = 24;
```

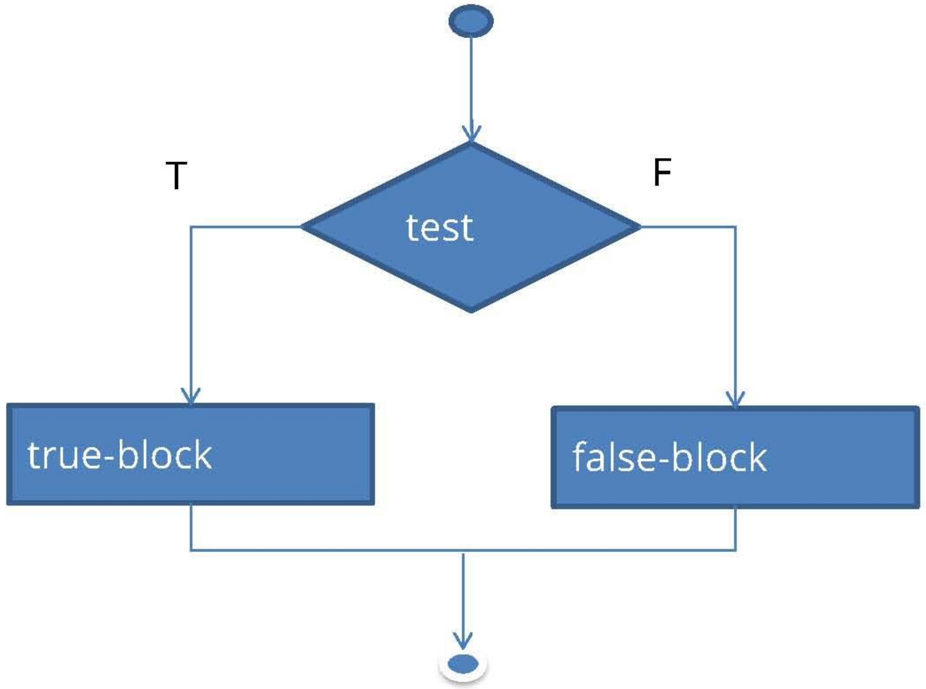
Module contents

- Control Flow Statements
  - Identifiers and Literals
  - Local variables: initialization and lifetime
  - Declaring a Variable as a Constant
  - The if-then and if-then-else statements
  - The switch statement
  - Loops: the while, do-while and for statements
  - The break and continue statements
  - The goto keyword
  - Program exit

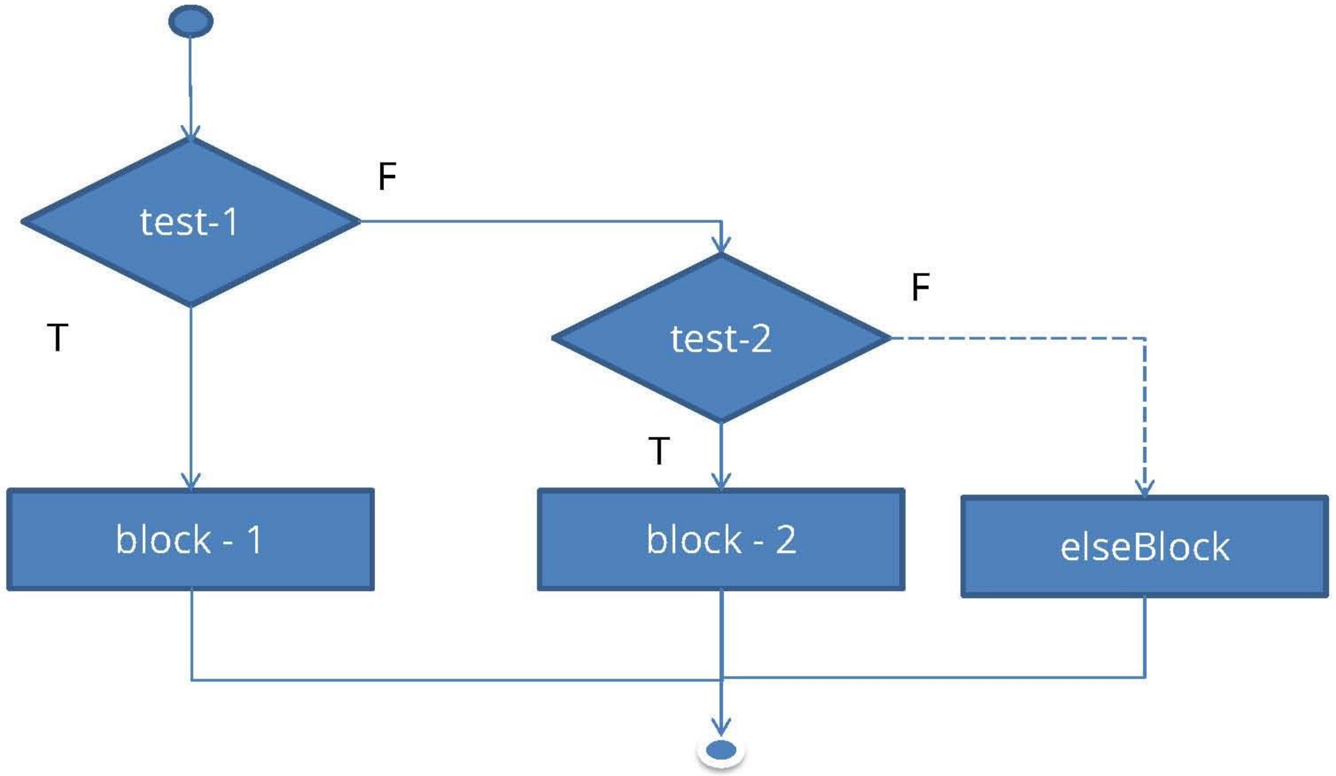
The if Statement 1/2



The if-else Statements 1/3



The if-else Statements 3/3

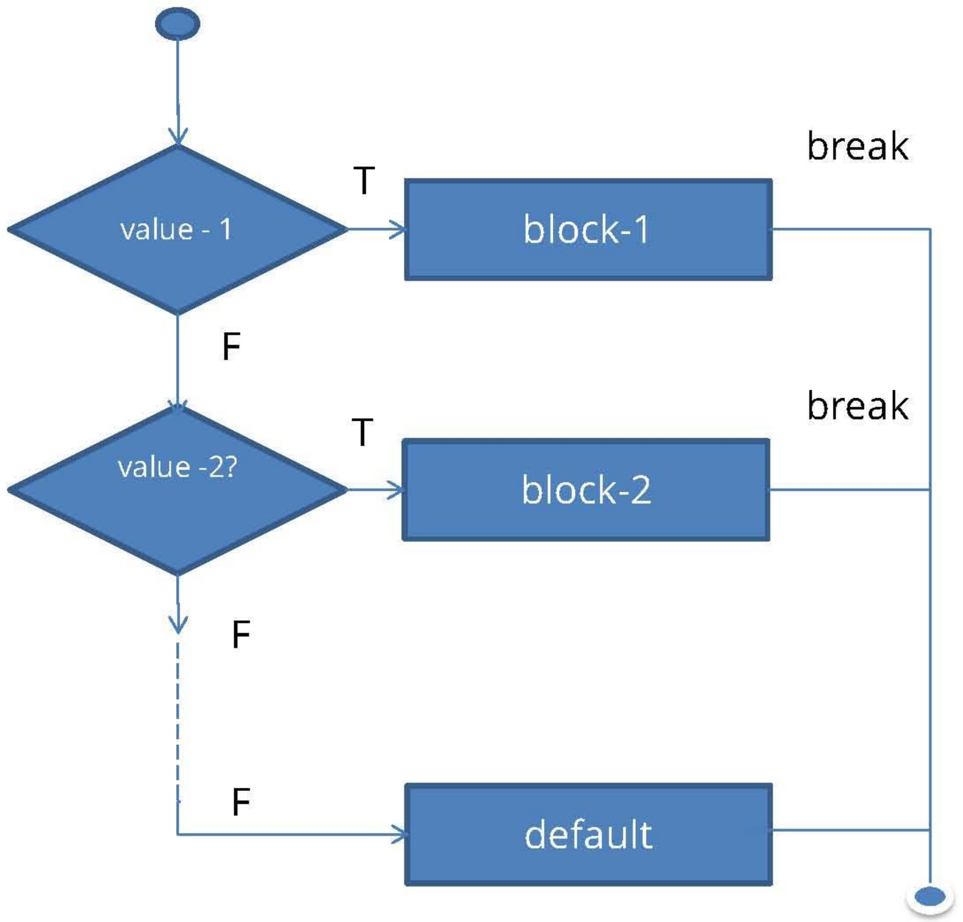


IfSwitchDemo.java

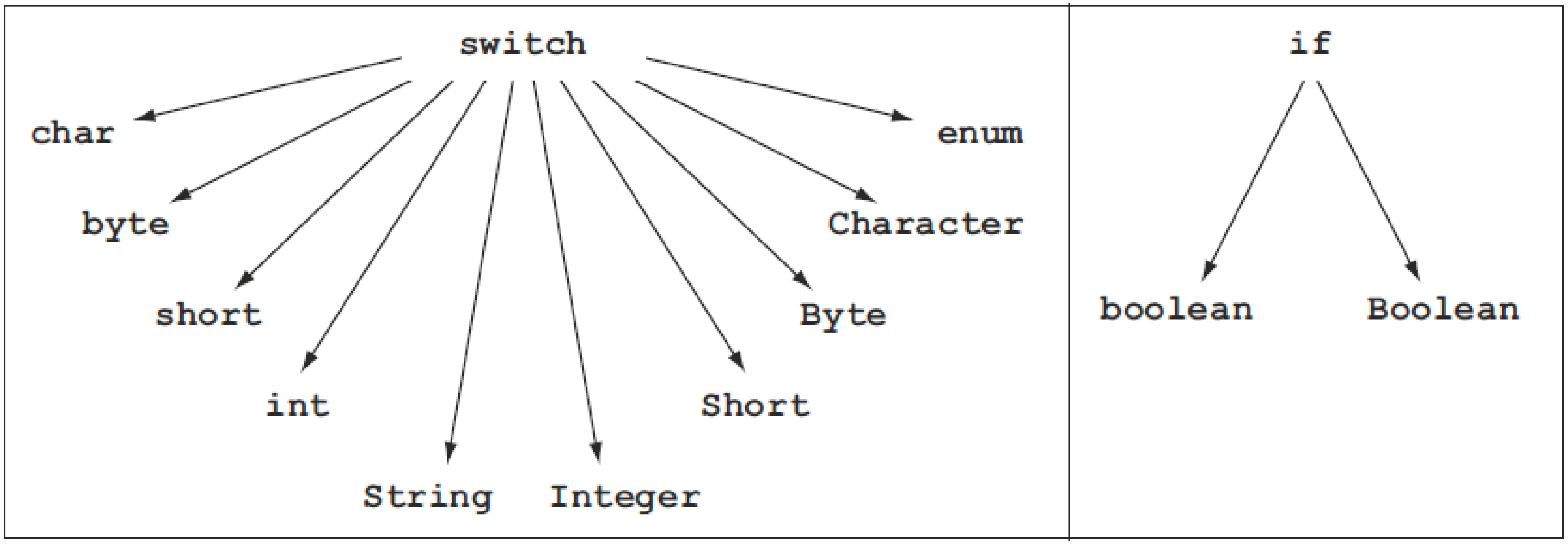
Module contents

- Control Flow Statements
  - Identifiers and Literals
  - Local variables: initialization and lifetime
  - Declaring a Variable as a Constant
  - The if-then and if-then-else statements
  - The switch statement
  - Loops: the while, do-while and for statements
  - The break and continue statements
  - The goto keyword
  - Program exit

The switch Statement 1/2

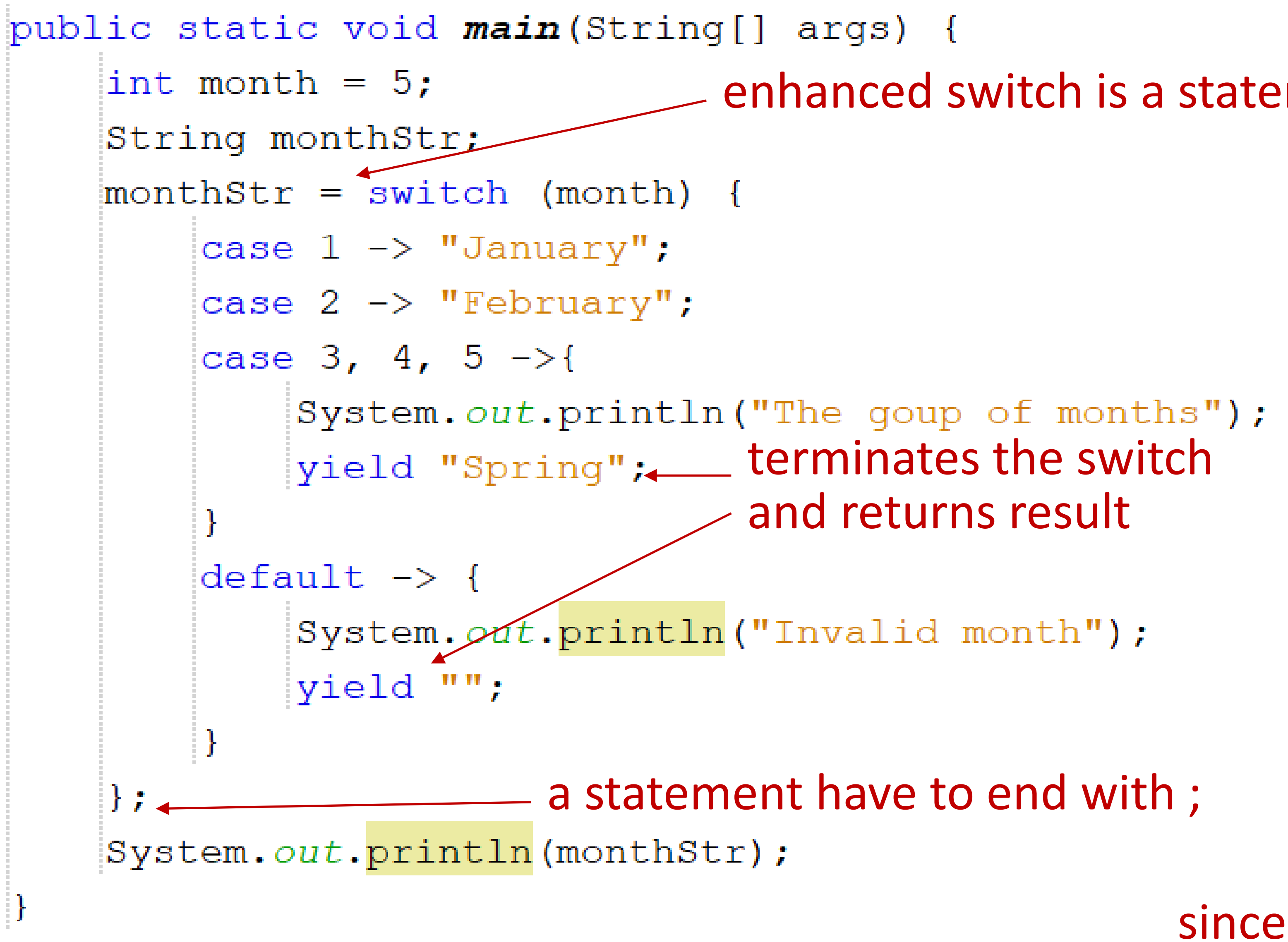


Argument types passed to a switch and if statements



IfSwitchDemo.java

The enhanced switch



```
public static void main(String[] args) {
    int month = 5;
    String monthStr;
    monthStr = switch (month) {
        case 1 -> "January";
        case 2 -> "February";
        case 3, 4, 5 ->{
            System.out.println("The goup of months");
            yield "Spring";
        }
        default -> {
            System.out.println("Invalid month");
            yield "";
        }
    };
    System.out.println(monthStr);
}
```

enhanced switch is a statement

terminates the switch
and returns result

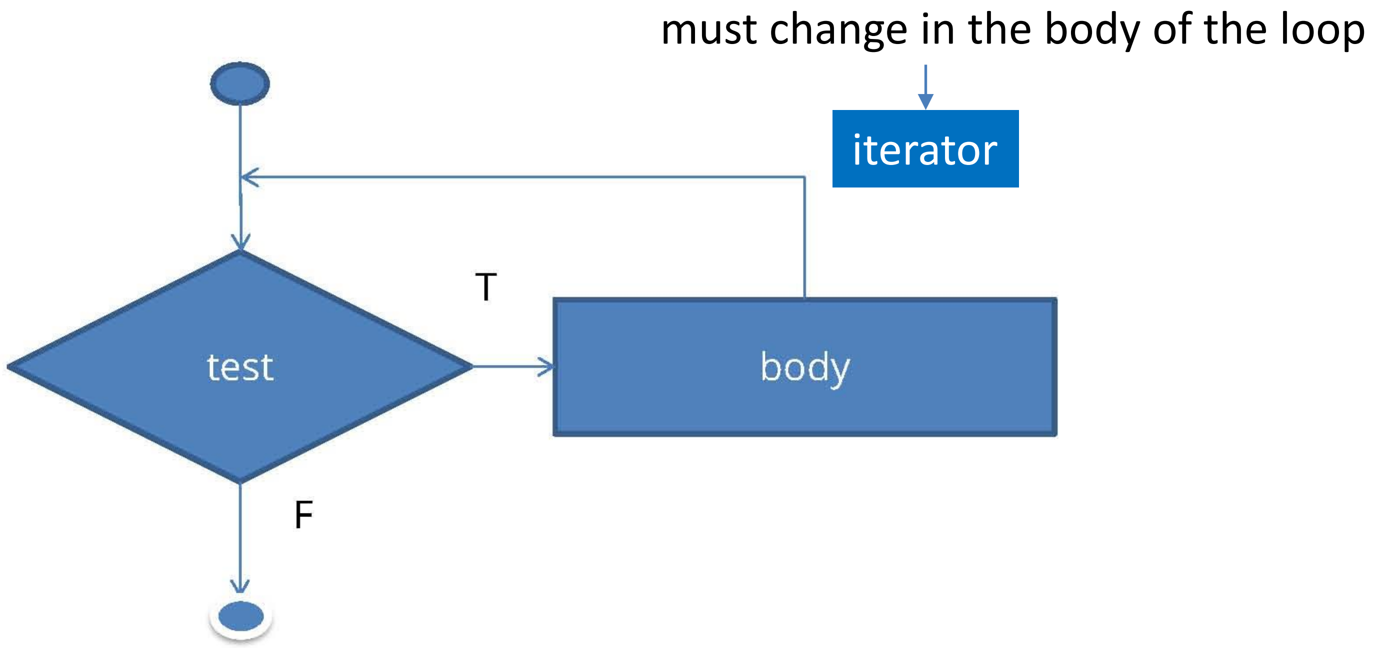
a statement have to end with ;

since JDK 14

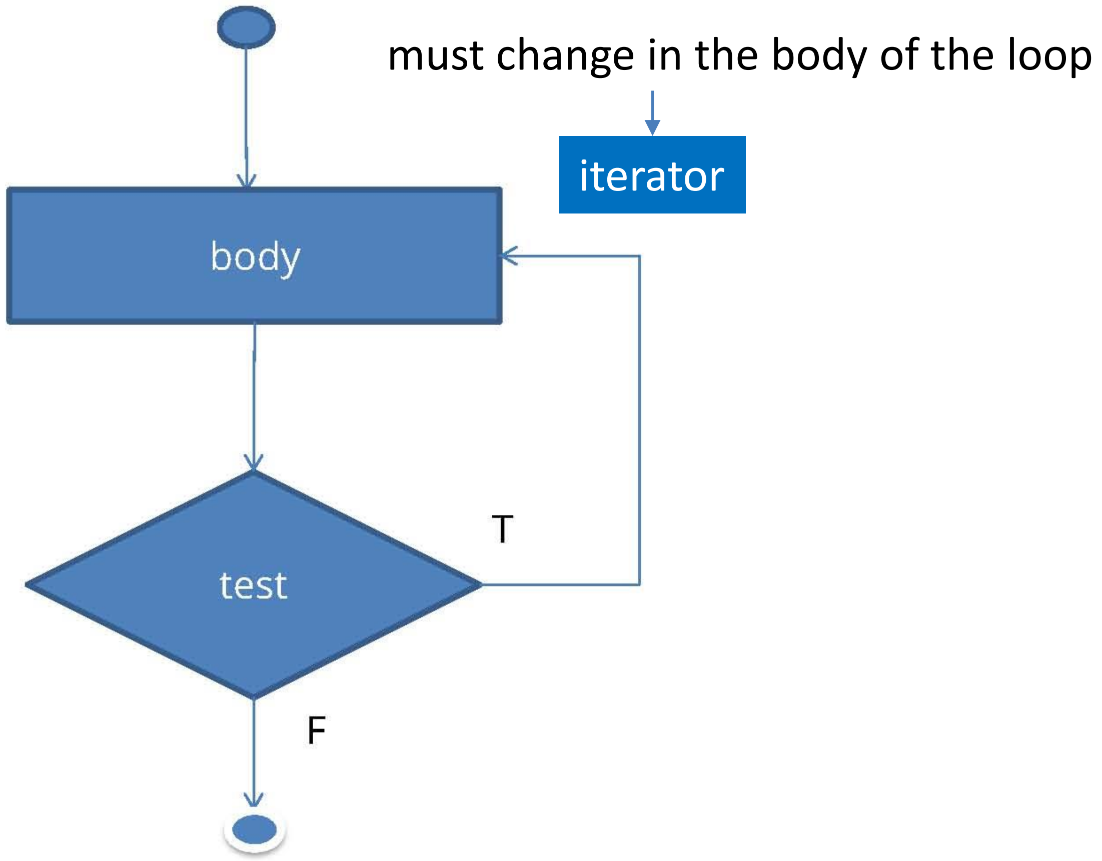
Module contents

- Control Flow Statements
  - Identifiers and Literals
  - Local variables: initialization and lifetime
  - Declaring a Variable as a Constant
  - The if-then and if-then-else statements
  - The switch statement
  - Loops: the while, do-while and for statements
  - The break and continue statements
  - The goto keyword
  - Program exit

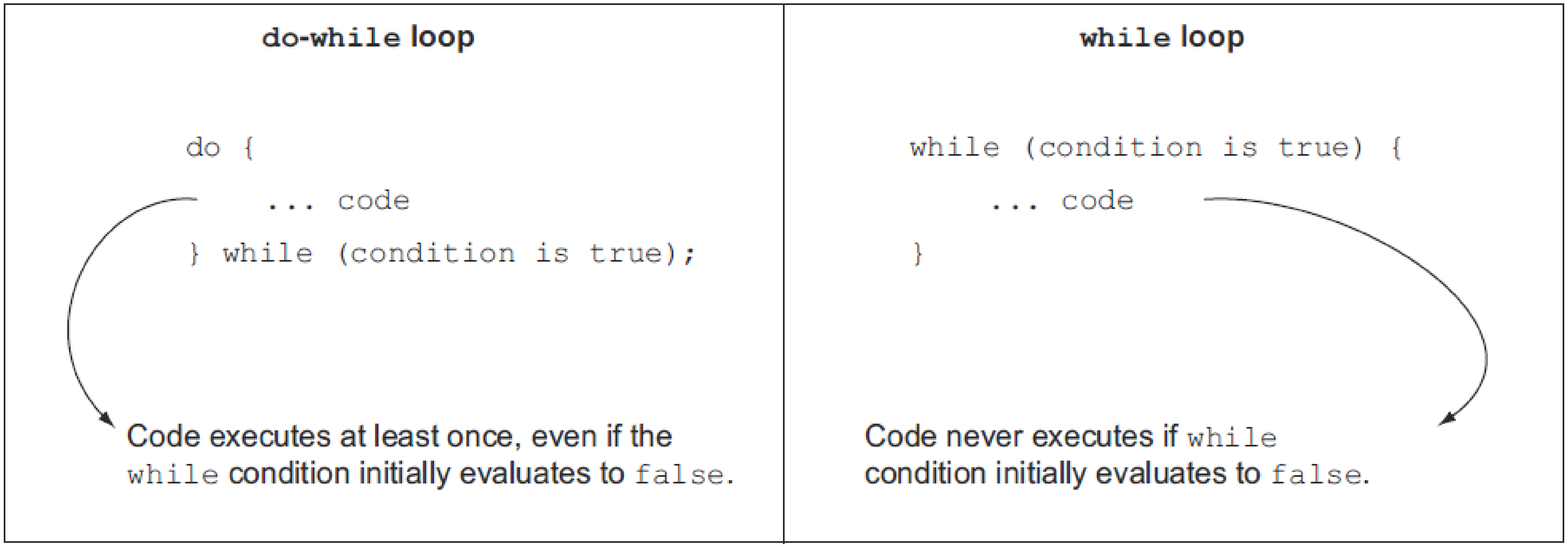
Loops: The while Statements



Do while 1/3



while vs do-while loop

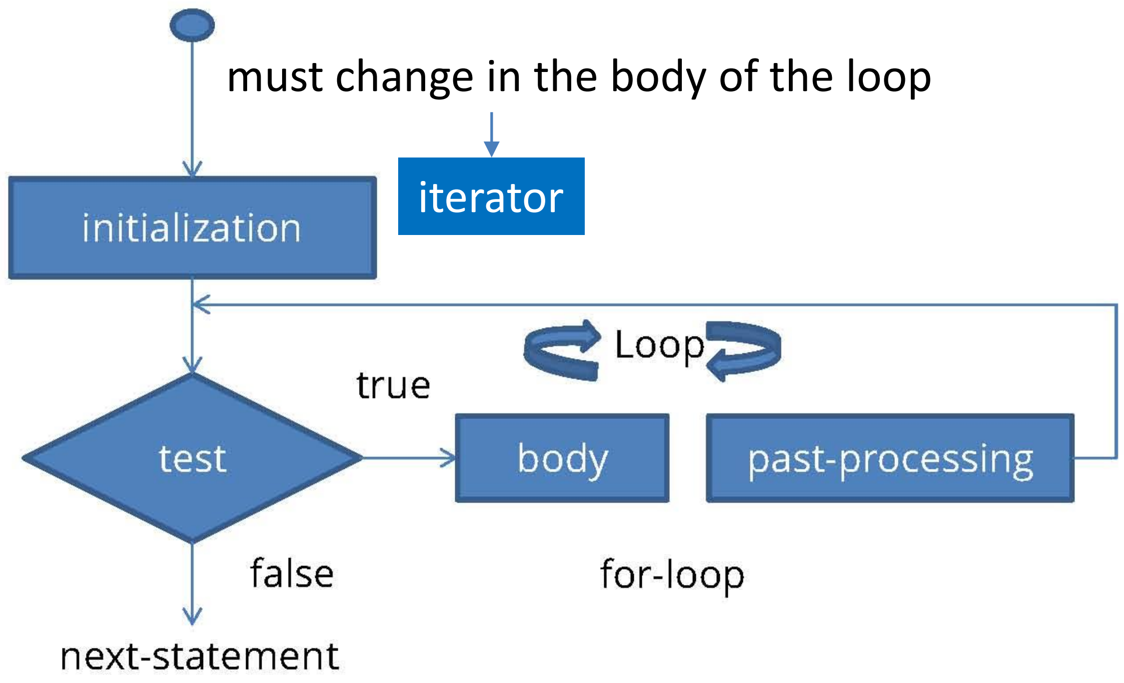


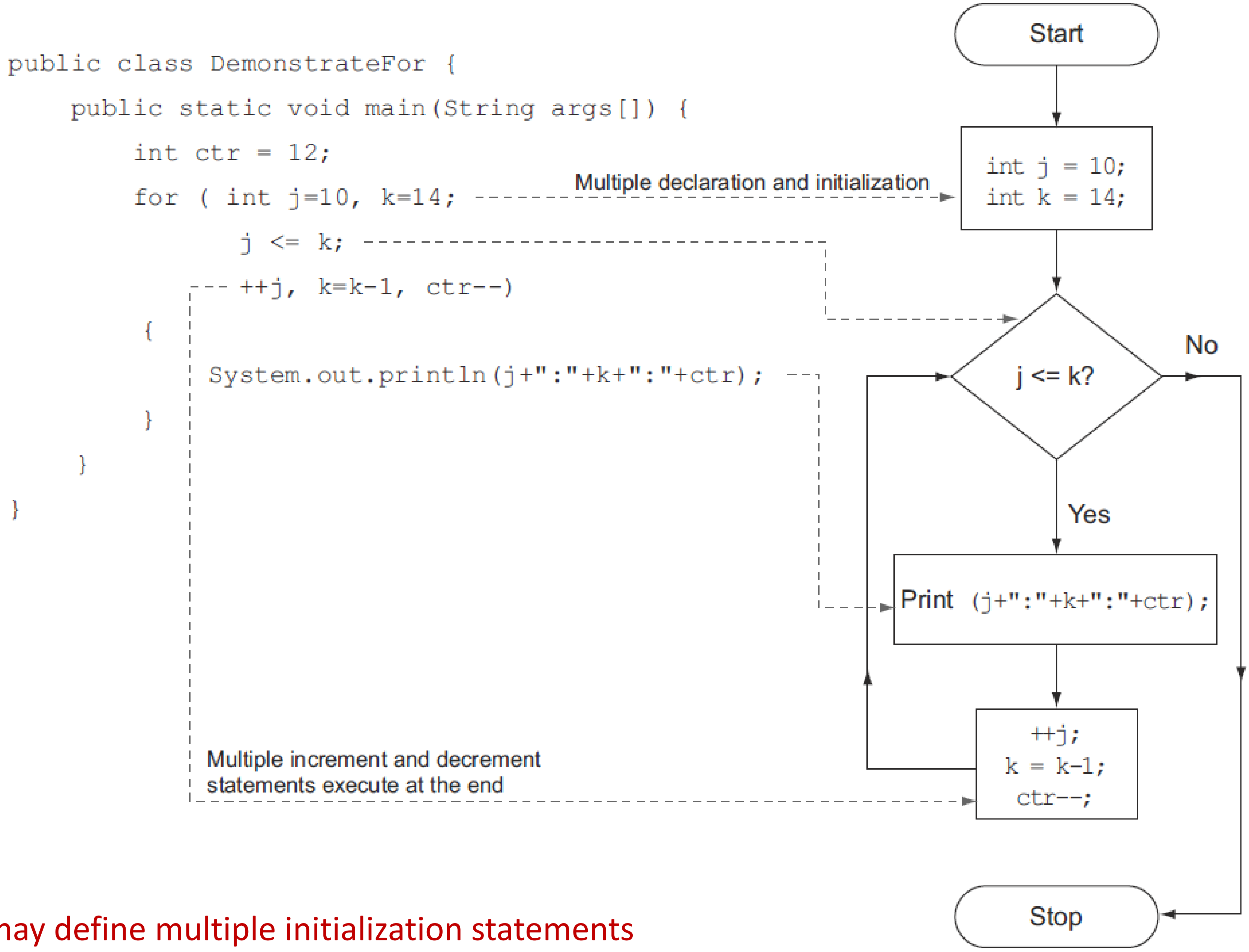
`do-while` loop	**`while` loop**

```
do {
    ... code
} while (condition is true);
```
	```
	while (condition is true) {
	    ... code
	}
	```

Code executes at least once, even if the `while` condition initially evaluates to `false`.	Code never executes if `while` condition initially evaluates to `false`.

for 1/3


must change in the body of the loop
iterator
initialization
Loop
true
test
body
past-processing
false
for-loop
next-statement



```
public class DemonstrateFor {
    public static void main(String args[]) {
        int ctr = 12;
        for ( int j=10, k=14;
                j <= k;
                ++j, k=k-1, ctr--)
            {
                System.out.println(j+":"+k+":"+ctr);
            }
    }
}
```

Multiple declaration and initialization
Multiple increment and decrement statements execute at the end
Start
int j = 10;
int k = 14;
j <= k?
No
Yes
Print (j+":"+k+":"+ctr);
++j;
k = k-1;
ctr--;
Stop

You may define multiple initialization statements and/or multiple update clause. But there can be only one termination condition for a for loop.

Module contents

- Control Flow Statements
  - Identifiers and Literals
  - Local variables: initialization and lifetime
  - Declaring a Variable as a Constant
  - The if-then and if-then-else statements
  - The switch statement
  - Loops: the while, do-while and for statements
  - The break and continue statements
  - The goto keyword
  - Program exit

The break and continue statements

break terminates: for, while, or do-while loop and switch case

BreakDemo.java

continue terminates a current iteration for, while, or do-while loop

ContinueDemo.java

The break and continue with labels

LabeledBreak.java

LabeledContinue.java

The return statement

return terminates a method

ReturnDemo.java

Module contents

- Control Flow Statements
  - Identifiers and Literals
  - Local variables: initialization and lifetime
  - Declaring a Variable as a Constant
  - The if-then and if-then-else statements
  - The switch statement
  - Loops: the while, do-while and for statements
  - The break and continue statements
  - The goto keyword
  - Program exit

The goto keyword

- Java has no goto statement .
- The Java keyword list specifies the goto keyword, but it is marked as "not used"
- Studies illustrated that goto is (mis)used more often than not simply "because it's there"
- Multi-level break and continue remove most of the need for goto statements

Studies on approximately 100,000 lines of C code determined that roughly 90 percent of the goto statements were used purely to obtain the effect of breaking out of nested loops

Module contents

- Control Flow Statements
  - Identifiers and Literals
  - Local variables: initialization and lifetime
  - Declaring a Variable as a Constant
  - The if-then and if-then-else statements
  - The switch statement
  - Loops: the while, do-while and for statements
  - The break and continue statements
  - The goto keyword
  - Program exit

Program Exit

A program terminates all its activity and exits when one of two things happens:

- All the threads that are not daemon threads terminate.

- Some thread invokes the exit method of class Runtime or class System, and the exit operation is not forbidden by the security manager.
- You can use System.exit(0) to close the program

ProgramExitDemo.java