

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)



دانشکده مهندسی کامپیوتر
و فناوری اطلاعات

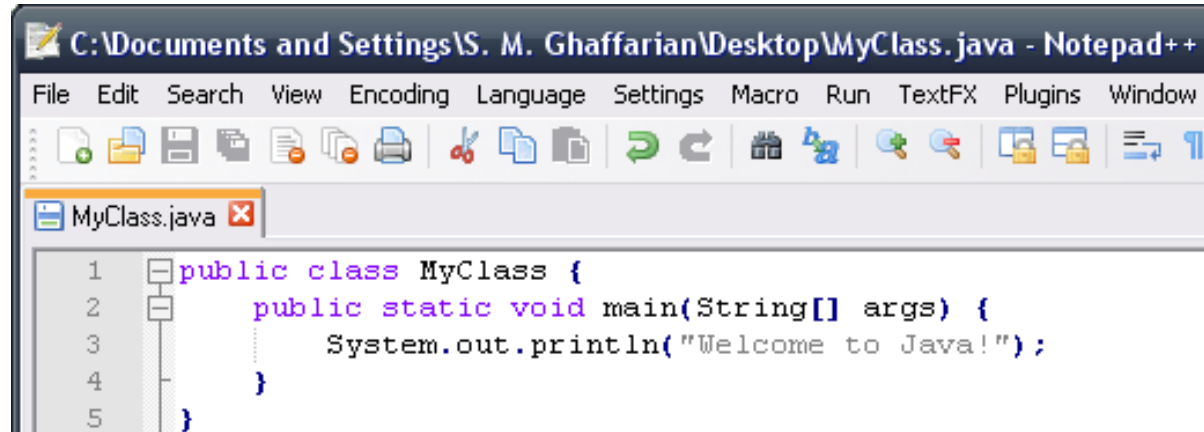
برنامه نویسی پیشرفته

برنامه نویسی ساخت یافته با Java

نیم سال اول ۹۹-۱۴۰۰

Java Basics

- Each Java file includes a **public class** with the same name as the file-name:



```
C:\Documents and Settings\S. M. Ghaffarian\Desktop\MyClass.java - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window
MyClass.java
1 public class MyClass {
2     public static void main(String[] args) {
3         System.out.println("Welcome to Java!");
4     }
5 }
```

- Just like the C language, the **main method** is the program's starting point.

Java Basics (continued ...)

- ❑ Java is a C based language
 - very similar to the syntax of C / C++

- ❑ The Java primitive data types:
 - byte, int, short, long, float, double, boolean, char

- ❑ Control statements are also mostly the same:
 - if, else, switch-case, while, for, do-while, continue, break

- ❑ Syntax of Java methods is also similar to C functions

print and println Methods

```
1  // Fig. 2.3: Welcome2.java
2  // Printing a line of text with multiple statements.
3
4  public class Welcome2
5  {
6      // main method begins execution of Java application
7      public static void main( String args[] )
8      {
9          System.out.print( "Welcome to " );
10         System.out.println( "Java Programming!" );
11
12     } // end method main
13
14 } // end class Welcome2
```

Good Old printf !!

```
1  // Fig. 2.6: Welcome4.java
2  // Printing multiple lines in a dialog box.
3
4  public class Welcome4
5  {
6      // main method begins execution of Java application
7      public static void main( String args[] )
8      {
9          System.out.printf( "%s\n%s\n",
10                          "Welcome to", "Java Programming!" );
11
12      } // end method main
13
14  } // end class Welcome4
```

```
Welcome to
Java Programming!
```

Simple Arithmetic Example

```
1  // Addition program
2  public class Addition {
3
4      // The main method
5      public static void main(String[] args) {
6
7          int num1 = 5;          // 1st integer
8
9          int num2 = 15;         // 2nd integer
10
11         int sum;
12         sum = num1 + num2; // sum of 2 integers
13
14         System.out.printf("Sum is %d", sum);
15     }
16 }
```

Arithmetic Operators

Java operation	Arithmetic operator	Algebraic expression	Java expression
Addition	+	$f + 7$	<code>f + 7</code>
Subtraction	-	$p - c$	<code>p - c</code>
Multiplication	*	bm	<code>b * m</code>
Division	/	x / y or $\frac{x}{y}$ or $x \div y$	<code>x / y</code>
Remainder	%	$r \bmod s$	<code>r % s</code>

Arithmetic Operators: Precedence

Operator(s)	Operation(s)	Order of evaluation (precedence)
* / %	Multiplication Division Remainder	Evaluated first. If there are several operators of this type, they are evaluated from left to right.
+ -	Addition Subtraction	Evaluated next. If there are several operators of this type, they are evaluated from left to right.

Arithmetic Operators (continued ...)

- Examples of operator precedence:

`z = p * r % q + w / x - y;`

`y = a * x * x + b * x + c;`

Relational Operators

Standard algebraic equality or relational operator	Java equality or relational operator	Sample Java condition	Meaning of Java condition
<i>Equality operators</i>			
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y
<i>Relational operators</i>			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than or equal to y
≤	<=	x <= y	x is less than or equal to y

Precedence & Associativity of Operators

Operators				Associativity	Type
*	/	%		left to right	multiplicative
+	-			left to right	additive
<	<=	>	>=	left to right	relational
==	!=			left to right	equality
=				right to left	assignment

Simple Example Program

```
public class IfElse0 {  
    public static void main(String[] args) {  
        int num1 = 18; // 1st integer  
        int num2 = 15; // 2nd integer  
        if (num1 == num2)  
            System.out.printf("%d == %d\n", num1, num2);  
        if (num1 != num2)  
            System.out.printf("%d != %d\n", num1, num2);  
        if (num1 > num2)  
            System.out.printf("%d > %d\n", num1, num2);  
        if (num1 < num2)  
            System.out.printf("%d < %d\n", num1, num2);  
        if (num1 >= num2)  
            System.out.printf("%d >= %d\n", num1, num2);  
        if (num1 <= num2)  
            System.out.printf("%d <= %d\n", num1, num2);  
    }  
}
```

if-else Control Statements

```
char gradeRank;  
float studentGrade = 18.0f;  
if (studentGrade >= 17) {  
    gradeRank = 'A';  
    System.out.println("Student Grade is A!");  
} else if (studentGrade >= 15) {  
    gradeRank = 'B';  
    System.out.println("Student Grade is B!");  
} else if (studentGrade >= 12) {  
    gradeRank = 'C';  
    System.out.println("Student Grade is C!");  
} else if (studentGrade >= 10) {  
    gradeRank = 'D';  
    System.out.println("Student Grade is D!");  
} else {  
    System.out.println("Student Failed!");  
}
```

if-else Control Statements

```
char gradeRank;
float studentGrade = 18.0f;
if (studentGrade >= 17) {
    gradeRank = 'A';
} else if (studentGrade >= 15) {
    gradeRank = 'B';
} else if (studentGrade >= 12) {
    gradeRank = 'C';
} else if (studentGrade >= 10) {
    gradeRank = 'D';
} else {
    gradeRank = 'F';
}
System.out.println("Student's grade is " + gradeRank + "!");
```

More on print

Output?

```
System.out.println(1 + 2);  
System.out.println(1 + 2 + " = 1 + 2");  
System.out.println("1 + 2 = " + 1 + 2);  
System.out.println("" + 1 + 2);  
System.out.println("1 + 2 = " + (1 + 2));
```

Increment & Decrement Operators

Operator	Operator name	Sample expression	Explanation
++	prefix increment	++a	Increment a by 1, then use the new value of a in the expression in which a resides.
++	postfix increment	a++	Use the current value of a in the expression in which a resides, then increment a by 1.
--	prefix decrement	--b	Decrement b by 1, then use the new value of b in the expression in which b resides.
--	postfix decrement	b--	Use the current value of b in the expression in which b resides, then decrement b by 1.

The Difference ...

```
public class PostVsPrefix {  
  
    public static void main(String[] args) {  
  
        int number1 = 5;  
        System.out.println("number1 is: " + number1);  
        System.out.println("number1 is: " + number1++);  
        System.out.println("number1 is: " + number1);  
  
        int number2 = 5;  
        System.out.println("number2 is: " + number2);  
        System.out.println("number2 is: " + ++number2);  
        System.out.println("number2 is: " + number2);  
    }  
}
```

Arithmetic Compound Assignment Operators

Assignment operator	Sample expression	Explanation	Assigns
<i>Assume:</i> <code>int c = 3, d = 5, e = 4, f = 6, g = 12;</code>			
<code>+=</code>	<code>c += 7</code>	<code>c = c + 7</code>	10 to c
<code>-=</code>	<code>d -= 4</code>	<code>d = d - 4</code>	1 to d
<code>*=</code>	<code>e *= 5</code>	<code>e = e * 5</code>	20 to e
<code>/=</code>	<code>f /= 3</code>	<code>f = f / 3</code>	2 to f
<code>%=</code>	<code>g %= 9</code>	<code>g = g % 9</code>	3 to g

The Conditional Operator

- The Conditional Operator (?:)

```
double studentGrade = 15.75;
```

```
System.out.println(studentGrade >= 10 ? "Passed!" : "Failed!");
```

- is equal to ...

```
double studentGrade = 15.75;
```

```
if (studentGrade >= 10)
    System.out.println("Passed!");
else
    System.out.println("Failed!");
```

Precedence & Associativity of Operators

Operators						Associativity	Type
++	--					right to left	unary postfix
++	--	+	-	(type)		right to left	unary prefix
*	/	%				left to right	multiplicative
+	-					left to right	additive
<	<=	>	>=			left to right	relational
==	!=					left to right	equality
?:						right to left	conditional
=	+=	-=	*=	/=	%=	right to left	assignment

Repetition Control Statements

- while Repetition Statement

```
int counter = 0;

while (counter < 10)
    counter++;

while (counter >= 0) {
    System.out.println(counter);
    counter--;
}
```

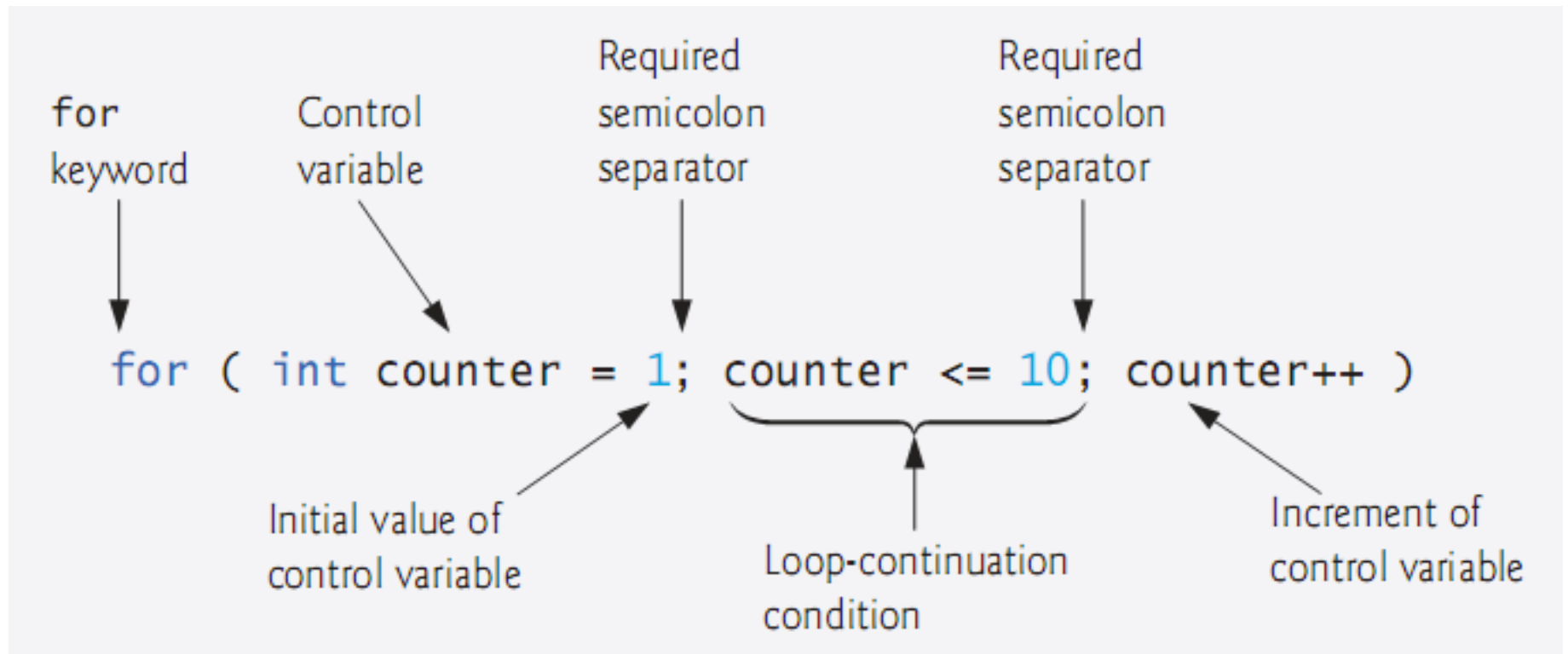
Repetition Control Statements (continued ...)

- for Repetition Statement

```
1  // Fig. 5.2: ForCounter.java
2  // Counter-controlled repetition with the for repetition statement.
3
4  public class ForCounter
5  {
6      public static void main( String args[] )
7      {
8          // for statement header includes initialization,
9          // loop-continuation condition and increment
10         for ( int counter = 1; counter <= 10; counter++ )
11             System.out.printf( "%d  ", counter );
12
13         System.out.println(); // output a newline
14     } // end main
15 } // end class ForCounter
```

1 2 3 4 5 6 7 8 9 10

Repetition Control Statements (continued ...)



Repetition Control Statements (continued ...)

```
for ( initialization; loopContinuationCondition; increment )  
    statement
```

□ is equal to ...

```
{  
    initialization;  
  
    while ( loopContinuationCondition )  
    {  
        statement  
        increment;  
    }  
}
```


Repetition Control Statements

(continued ...)

Vary the control variable from 7 to 77 in increments of 7.

```
for ( int i = 7; i <= 77; i += 7 )
```

Vary the control variable from 20 to 2 in decrements of 2.

```
for ( int i = 20; i >= 2; i -= 2 )
```

Vary the control variable over the following sequence of values: 2, 5, 8, 11, 14, 17, 20.

```
for ( int i = 2; i <= 20; i += 3 )
```

Vary the control variable over the following sequence of values: 99, 88, 77, 66, 55, 44, 33, 22, 11, 0.

```
for ( int i = 99; i >= 0; i -= 11 )
```

Repetition Control Statements

(continued ...)

- Summation of even numbers in the range of 2 to 20:

```
for ( int number = 2; number <= 20; total += number, number += 2 )  
    ; // empty statement
```

Repetition Control Statements (continued ...)

- do...while Repetition Statement

```
1 // Fig. 5.7: DoWhileTest.java
2 // do...while repetition statement.
3
4 public class DoWhileTest
5 {
6     public static void main( String args[] )
7     {
8         int counter = 1; // initialize counter
9
10        do
11        {
12            System.out.printf( "%d ", counter );
13            ++counter;
14        } while ( counter <= 10 ); // end do...while
15
16        System.out.println(); // outputs a newline
17    } // end main
18 } // end class DoWhileTest
```

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

Write A Simple Program!

- ❑ Assume: `getNum()` is a method that reads a number (integer) from the user and returns it. You don't need to know how it works for now.
- ❑ Using `getNum()` write a program that reads integer values and adds the values as long as the entered number is not -1. After the user enters -1, the program shows the sum of the numbers (not including the last -1).

break Statement

```
1 // Fig. 5.12: BreakTest.java
2 // break statement exiting a for statement.
3 public class BreakTest
4 {
5     public static void main( String args[] )
6     {
7         int count; // control variable also used after loop terminates
8
9         for ( count = 1; count <= 10; count++ ) // loop 10 times
10        {
11            if ( count == 5 ) // if count is 5,
12                break;       // terminate loop
13
14            System.out.printf( "%d ", count );
15        } // end for
16
17        System.out.printf( "\nBroke out of loop at count = %d\n", count );
18    } // end main
19 } // end class BreakTest
```

```
1 2 3 4
Broke out of loop at count = 5
```

continue Statement

```
1 // Fig. 5.13: ContinueTest.java
2 // continue statement terminating an iteration of a for statement.
3 public class ContinueTest
4 {
5     public static void main( String args[] )
6     {
7         for ( int count = 1; count <= 10; count++ ) // loop 10 times
8         {
9             if ( count == 5 ) // if count is 5,
10                 continue; // skip remaining code in loop
11
12             System.out.printf( "%d ", count );
13         } // end for
14
15         System.out.println( "\nUsed continue to skip printing 5" );
16     } // end main
17 } // end class ContinueTest
```

```
1 2 3 4 6 7 8 9 10
Used continue to skip printing 5
```

Logical Operators

- Conditional AND and OR operators

```
double studentGrade = 16.25;

if (17 <= studentGrade && studentGrade <= 20)
    System.out.println("Student Grade is A");

int integer = 3;

if (integer == 3 || integer == 5 || integer == 7)
    System.out.println("integer is and odd number");
```

Write Another Program!

- ❑ Write a program that given a year shows if the year is a leap year or not.

■ چنانچه باقی مانده حاصل تقسیم سال مورد نظر (سال‌های ۱۳۴۳ تا ۱۴۷۲) بر عدد ۳۳، یکی از اعداد (۱، ۵، ۹، ۱۳، ۱۷، ۲۲، ۲۶ و ۳۰) باشد، آن سال کبیسه است

○ از صفحه سال کبیسه ویکی‌پدیا

switch Multiple-Selection Statement

```
char character = 'A';

switch (character) {
    case 'A':
        System.out.println('A');
        break;
    case 'B':
    case 'C':
        System.out.println("B or C");
        break;
    case 'D':
        System.out.println('D');
        break;
    default:
        System.out.println("Any character except: A, B, C and D");
}
```

Logical Operators (continued ...)

- Logical Negation Operator

```
char c = 'b';  
if (!(c == 'a'))  
    System.out.println("character isn't 'a'");
```

Precedence & Associativity of Operators

Operators	Associativity	Type
++ --	right to left	unary postfix
++ -- + - ! (type)	right to left	unary prefix
* / %	left to right	multiplicative
+ -	left to right	additive
< <= > >=	left to right	relational
== !=	left to right	equality
&	left to right	boolean logical AND
^	left to right	boolean logical exclusive OR
	left to right	boolean logical inclusive OR
&&	left to right	conditional AND
	left to right	conditional OR
?:	right to left	conditional
= += -= *= /= %=	right to left	assignment

Primitive Data-Types

Type	Size in bits	Values	Standard
boolean		true or false [Note: A boolean's representation is specific to the Java Virtual Machine on each platform.]	
char	16	'\u0000' to '\uFFFF' (0 to 65535)	(ISO Unicode character set)
byte	8	-128 to +127 (-2^7 to $2^7 - 1$)	
short	16	-32,768 to +32,767 (-2^{15} to $2^{15} - 1$)	
int	32	-2,147,483,648 to +2,147,483,647 (-2^{31} to $2^{31} - 1$)	
long	64	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807 (-2^{63} to $2^{63} - 1$)	
float	32	Negative range: -3.4028234663852886E+38 to -1.40129846432481707e-45 Positive range: 1.40129846432481707e-45 to 3.4028234663852886E+38	(IEEE 754 floating point)
double	64	Negative range: -1.7976931348623157E+308 to -4.94065645841246544e-324 Positive range: 4.94065645841246544e-324 to 1.7976931348623157E+308	(IEEE 754 floating point)

Code Aesthetics

- Indent the code inside a block (4x spaces or 1x tab)
- Put a space on both sides of every operator
- Start the name of every variable with lower-case letters
- Start the name of every class with upper-case letters
- Use Camel-case letters for all names

Java Coding Conventions

- **Sun Microsystems original Java coding conventions:**
 - www.oracle.com/technetwork/java/codeconventions-150003.pdf
- **Google's Java coding conventions:**
 - <https://google.github.io/styleguide/javaguide.html>
- **Twitter's Java coding conventions:**
 - github.com/twitter/commons/blob/master/src/java/com/twitter/common/styleguide.md

References

- **Deitel's Java How to Program (7th Edition)**

- ❑ Chapter 2

- ❑ Chapter 4

- ❑ Chapter 5