JAVA PROGRAMMING

Week 2: Program Control Statements

Lecturer:

- Hồ Tuấn Thanh, M.Sc.

Plan

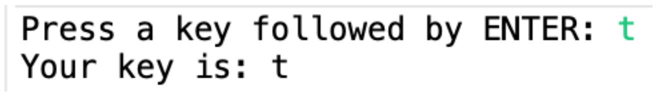
1. Input characters from the keyboard
2. The if statement
3. The switch statement
4. The complete form of the for loop
5. The while loop
6. The dowhile loop
7. Use break to exit a loop
8. Use break as a form of goto
9. Apply continue
10. Nest loops

Input characters from the keyboard

- To read a character from the keyboard, we will use System.in.read().
- System.in is the complement to System.out.
- The read() method waits until the user presses a key and then returns the result.
- The character is returned as an integer, so it must be cast into a char to assign it to a char variable.
- By default: console input is line buffered. The buffer holds a complete line of text → the program waits for ENTER pressed by the user.

Example

```
// Read a character from the keyboard.
public class KbIn {
        public static void main(String[] args)
                                                throws java.io.IOException{
                char ch;

                System.out.print("Press a key followed by ENTER: ");
                ch = (char) System.in.read(); // get a char
                System.out.println("Your key is: " + ch);
        }
}
```



```
Press a key followed by ENTER: t
Your key is: t
```

if statement [1]

```
if(condition)
    statement;
else
    statement;
```

- The targets of the if and else are single statements.
- The else clause is optional.
- The targets of both the if and else can be blocks of statements.

if statement [2]

```
if(condition)
{
    statement sequence
}
else
{
    statement sequence
}
```

Example: Guess a letter (V1)

```
// Guess the letter game.
public class Guess {
    public static void main(String[] args)
        throws java.io.IOException{

        char ch, answer = 'K';
        System.out.println("I'm thinking of a letter
                                        between A and Z.");
        System.out.print("Can you guess it: ");
        // read a char from the keyboard
        ch = (char)System.in.read();
        if(ch == answer) System.out.println("** Right **");
    }
}
```

Example: Guess a letter (V2)

```java
// Guess the letter game.
public class Guess2 {
        public static void main(String[] args)
                                        throws java.io.IOException{
                char ch, answer = 'K';
                System.out.println("I'm thinking of a letter
                                                between A and Z.");
                System.out.print("Can you guess it: ");
                // read a char from the keyboard
                ch = (char)System.in.read();
                if(ch == answer) System.out.println("** Right **");
                else System.out.println("...Sorry, you're wrong.");
        }
}
```

Nested ifs

- A nested if is an if statement that is the target of another if or else.
- Nested ifs are very common in programming.
- Main thing to remember: an else statement always refers to the nearest if statement that is within the same block as the else and not already associated with an else.

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

```
// Guess the letter game.
public class Guess3 {
    public static void main(String[] args)
        throws java.io.IOException{
        char ch, answer = 'K';
        System.out.println("I'm thinking of a letter between A
                                                            and Z.");
        System.out.print("Can you guess it: ");
        ch = (char)System.in.read();
        if(ch == answer) System.out.println("** Right **");
        else {
            System.out.print("...Sorry, you're ");
            // a nested if
            if (ch < answer) System.out.println("too low");
            else System.out.println("too high");
        }
    }
}
```

The if-else-if ladder

```
if(condition)
        statement;
else if(condition)
        statement;
else if(condition)
        statement;
.
.
.
else
        statement;
```

```java
// Demonstrate an if-else-if ladder.
public class Ladder {
    public static void main(String[] args) {
        int x;
        for(x = 0; x < 6; x++) {
            if(x == 1)
                System.out.println("x is one");
            else if(x == 2)
                System.out.println("x is two");
            else if(x == 3)
                System.out.println("x is three");
            else if(x == 4)
                System.out.println("x is four");
            else // this is the default statement
                System.out.println("x is not between 1 and 4");
        }
    }
}
```

```
x is not between 1 and 4
x is one
x is two
x is three
x is four
x is not between 1 and 4
```

The switch statement

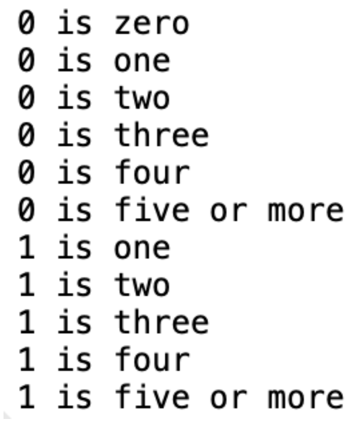
```
switch(expression) {
        case constant1:
                statement sequence
                break;
        case constant2:
                statement sequence
                break;
        case constant3:
                statement sequence
                break;
        ...
        default:
                statement sequence
}
```

Prior to JDK 7:
The expression controlling the switch must resolve to type byte, short, int, char, or an enumeration.
From JDK 7:
Expression can also be of type String.

```java
// Demonstrate the switch
public class SwitchDemo {
        public static void main(String[] args) {
                int i;
                for(i=0; i<10; 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;
                                case 4:
                                        System.out.println(i + " is four"); break;
                                default:
                                        System.out.println(i + " is five or more");
                        }
        }
}
```

```
1 is one
2 is two
3 is three
4 is four
5 is five or more
6 is five or more
7 is five or more
8 is five or more
9 is five or more
```

```java
// Demonstrate the switch without break statements
public class NoBreak {
    public static void main(String[] args) {
        int i;
        for(i=0; i<5; i++)
            switch(i) {
                case 0:
                    System.out.println(i + " is zero");
                case 1:
                    System.out.println(i + " is one");
                case 2:
                    System.out.println(i + " is two");
                case 3:
                    System.out.println(i + " is three");
                case 4:
                    System.out.println(i + " is four");
                default:
                    System.out.println(i + " is five or more");
            }
    }
}
```



```
0 is zero
0 is one
0 is two
0 is three
0 is four
0 is five or more
1 is one
1 is two
1 is three
1 is four
1 is five or more
```

```
switch(i) {
        case 1:
        case 2:
        case 3: System.out.println("i is 1, 2 or 3");
                break;
        case 4: System.out.println("i is 4");
                break;
}
```

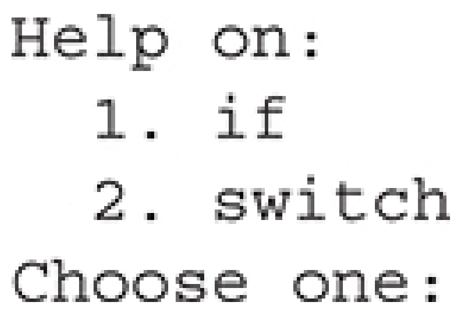
Nested switch statements

- It is possible to have a switch as part of the statement sequence of an outer switch. This is called a nested switch.

```
switch(ch1) {
        case 'A’:
                System.out.println("This A is part of outer switch.");
                switch(ch2) {
                        case 'A’:
                                System.out.println("This A is part of inner
                                                                switch");
                                break;
                        case 'B': // ...
                }// end of inner switch
        case 'B': //...
        }
}
```

Exercise: Building a Java Help System

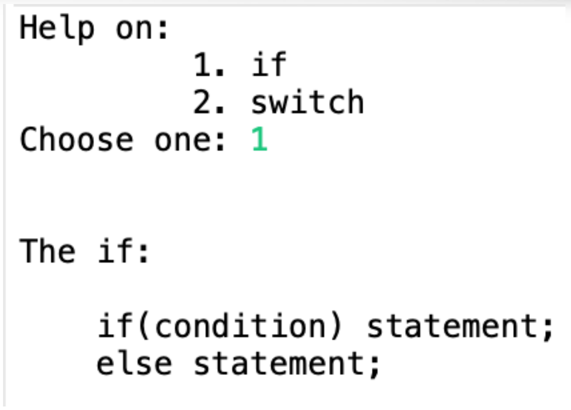
1. Create a file called Help.java.
2. The program begins by displaying the following menu:



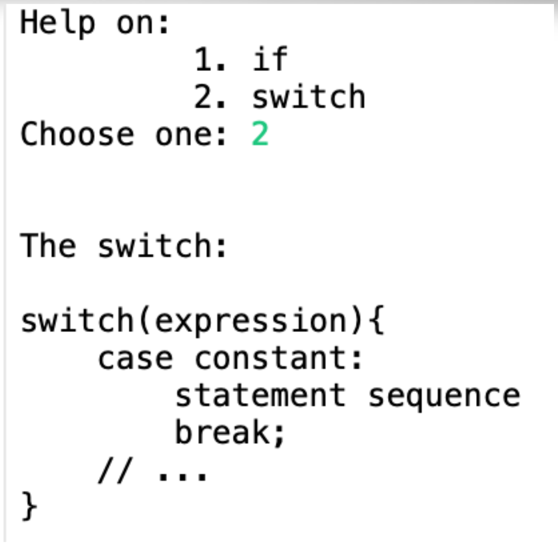
```
Help on:
   1. if
   2. switch
Choose one:
```

1. Next, the program obtains the user's selection by calling System.in.read().
2. Once the selection has been obtained, the program uses the switch statement to display the syntax for the selected statement.
3. Compile and run.

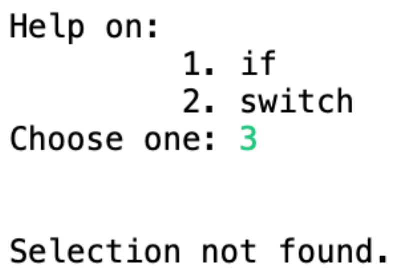
Result



```
Help on:
        1. if
        2. switch
Choose one: 1


The if:

    if(condition) statement;
    else statement;
```



```
Help on:
        1. if
        2. switch
Choose one: 2


The switch:

switch(expression){
    case constant:
        statement sequence
        break;
    // ...
}
```



```
Help on:
        1. if
        2. switch
Choose one: 3


Selection not found.
```

The for loop

- General form:

```
        for(initialization; condition; iteration) statement;


for(initialization; condition; iteration) {
        statement sequence
}
```

- initialization: sets the initial value of the loop control variable
- condition: is a Boolean expression that determines whether or not the loop will repeat.
- iteration expression defines the amount by which the loop control variable will change each time the loop is repeated.

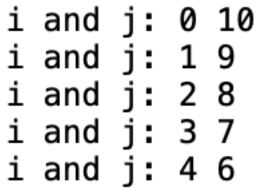
Example

```
// Show quare roots of 1 to 99 and the rounding error.
public class SqrRoot {
    public static void main(String[] args) {
        double num, sroot, rerr;
        for(num = 1.0; num < 100.0; num++) {
            sroot = Math.sqrt(num);
            System.out.println("Square root of " + num +
                                        " is " + sroot);
            rerr = num - (sroot * sroot);
            System.out.println("Rounding error is " + rerr);
            System.out.println();
        }
    }
}
```

```
// A negatively running for loop.
public class DecrFor {
        public static void main(String[] args) {
        int x;
        for(x = 100; x > -100; x -=5)
                System.out.println(x);
        }
}

for(count = 10; count < 5; count++)
                x += count; // this statement will not execute
```

Some variations on the for loop

```java
// Use commas in a for statement
public class Comma {
        public static void main(String[] args) {
                int i, j;
                for(i = 0, j = 10; i < j; i++, j--)
                        System.out.println("i and j: " + i + " " + j);
        }
}
```



```
i and j: 0 10
i and j: 1 9
i and j: 2 8
i and j: 3 7
i and j: 4 6
```

```java
//Loop until an S is typed.
public class ForTest {
	public static void main(String[] args)
		throws java.io.IOException{
		int i;
		System.out.println("Press S to stop.");
		for(i = 0; (char)System.in.read() != 'S'; i++)
			System.out.println("Pass #" + i);
	}
}
```

Missing pieces [1]

```
// Parts of the for can be empty
public class Empty {
	public static void main(String[] args) {
		int i;
		for(i = 0; i < 10;) {
			System.out.println("Pass #" + i);
			i++; // increment loop control var
		}
	}
}
```

Missing pieces [2]

```
// Move more out of the for loop
public class Empty2 {
    public static void main(String[] args) {
        int i = 0; // move intinitalization out of loop
        for(; i < 10;) {
            System.out.println("Pass #" + i);
            i++; // increment loop control var
        }
    }
}
```

The Infinite Loop

Consider the following code:

```
for(;;)
{
		//...
}
```

This loop will run forever → infinite loop.

Loops with no body

```java
// The body of the loop can be empty
public class Empty3 {
	public static void main(String[] args) {
		int i = 0;
		int sum = 0;

		// sum the number through 5
		for(i = 0; i <= 5; sum += i++);

		System.out.println("Sum is: " + sum);
	}
}
```

```
Sum is: 15
```

Declaring loop control variables inside the for loop

```
// Declare loop control variable inside the for
public class ForVar {
        public static void main(String[] args) {
                int sum = 0;
                int fact = 1;
                // compute the factorial of the numbers through 5
                for(int i = 1; i <= 5; i++) {
                        sum += i;// i is known throughout the loop
                        fact *= i;
                }
                //but, is is not known here
                System.out.println("Sum is " + sum);
                System.out.println("Factorial is " + fact);
        }
}
```

Plan

Syntax

```
while(condition) statement;
```

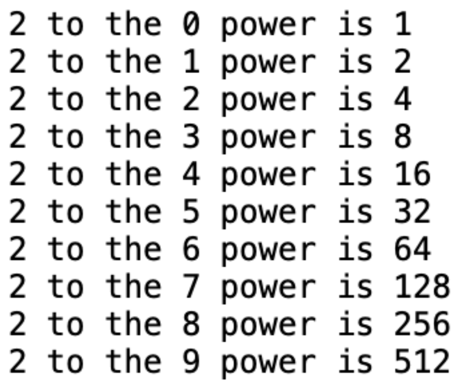
- statement may be a single statement or a block of statements
- condition defines the condition that controls the loop.
  - The condition may be any valid Boolean expression.
  - The loop repeats while the condition is true.
  - When the condition becomes false, program control passes to the line immediately following the loop.

Example

```
// Demonstrate the while loop
public class WhileDemo {
    public static void main(String[] args) {
        char ch;
        // print the alphabet using a while loop
        ch = 'a';
        while (ch <= 'z') {
            System.out.print(ch);
            ch++;
        }
    }
}
```

Exercise:

- Compute 2^i, for each i ∈ [0,10)



```
2 to the 0 power is 1
2 to the 1 power is 2
2 to the 2 power is 4
2 to the 3 power is 8
2 to the 4 power is 16
2 to the 5 power is 32
2 to the 6 power is 64
2 to the 7 power is 128
2 to the 8 power is 256
2 to the 9 power is 512
```

Plan

Syntax

```
do{
        statements;
}while (condition);
```

- Difference between while loop and do … while loop?

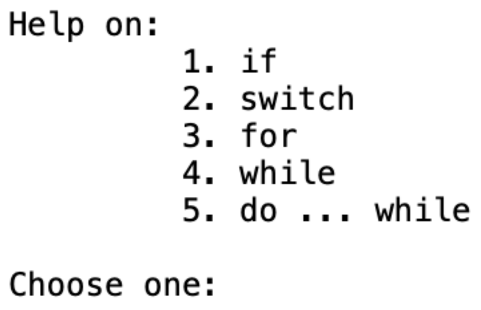
Example

```java
// Demonstrate the do-while loop
public class DWDemo {
	public static void main(String[] args)
		throws java.io.IOException{
		char ch;
		do {
			System.out.print("Press a key followed by ENTER: ");
		ch = (char) System.in.read(); // get a char
		}while(ch !='q’);
	}
}
```

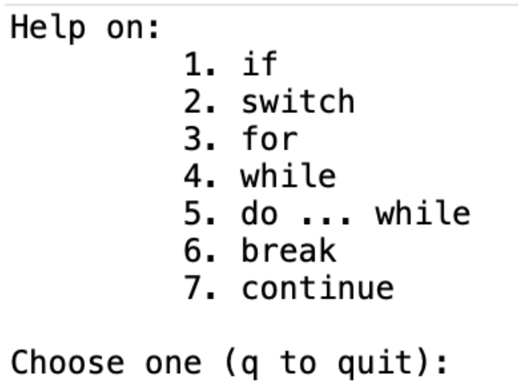
```
// Guess the letter game.
public class Guess4 {
	public static void main(String[] args)
		throws java.io.IOException{
		char ch, ignore, answer = 'K';
		do {
			System.out.println("I'm thinking of a letter between
			                                                  A and Z.");
			System.out.print("Can you guess it: ");
			ch = (char)System.in.read();
			do { ignore = (char)System.in.read();
			}while(ignore != '\n');
			if(ch == answer) System.out.println("** Right **");
			else {
				System.out.print("...Sorry, you're ");
				if (ch < answer) System.out.println("too low");
				else System.out.println("too high");
				System.out.println("Try again!\n");
			}//end of else
		}while(answer !=ch);
	}
}
```

Exercise: Improve the Java Help System

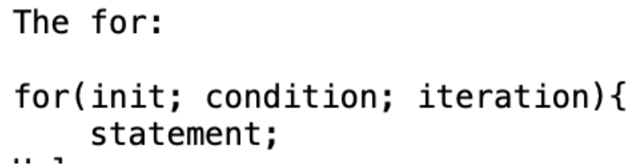
- This exercise expands on the Java help system.
- This version adds the syntax for the for, while, and do ... while loops. It also checks the user's menu selection, looping until a valid response is entered.



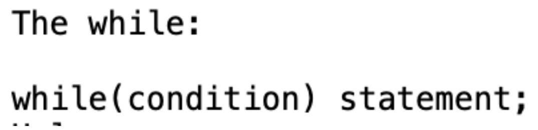
```
Help on:
        1. if
        2. switch
        3. for
        4. while
        5. do ... while

Choose one:
```



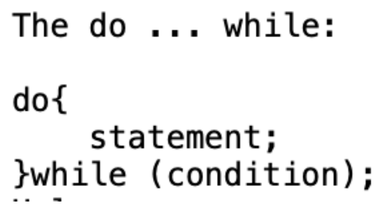
```
Help on:
        1. if
        2. switch
        3. for
        4. while
        5. do ... while
        6. break
        7. continue

Choose one (q to quit):
```



```
The for:

for(init; condition; iteration){
    statement;
```



```
The while:

while(condition) statement;
```



```
The do ... while:

do{
    statement;
}while (condition);
```

Plan

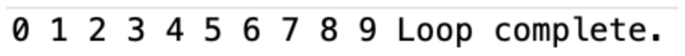
1. Input characters from the keyboard
2. The if statement
3. The switch statement
4. The complete form of the for loop
5. The while loop
6. The dowhile loop
7. Use break to exit a loop
8. Use break as a form of goto
9. Apply continue
10. Nest loops

break statement

- Use the break statement to force an immediate exit from a loop, bypassing any remaining code in the body of the loop and the loop's conditional test.
- When a break statement is encountered inside a loop:
  - the loop is terminated and
  - program control resumes at the next statement following the loop.

Example

```java
//Using break to exit a loop.
public class BreakDemo {
    public static void main(String[] args) {
        int num = 100;
        //loop while i-square is less than num
        for(int i = 0; i < num; i++) {
            // terminate loop if i*i >= 100;
            if (i*i >= num) break;
            System.out.print(i + " ");
        }
        System.out.println("Loop complete.");
    }
}
```



```
0 1 2 3 4 5 6 7 8 9 Loop complete.
```

```java
// Read input until a q is received.
public class Break2 {
          public static void main(String[] args)
                    throws java.io.IOException{
                    char ch;
                    for(;;) {
                              ch = (char)System.in.read(); // get a char
                              if(ch == 'q') break;
                    }
                    System.out.println("Your pressed q!");
          }
}
```

```java
// Using break with nested loops.
public class Break3 {
	public static void main(String[] args) {
		for(int i = 0; i < 3; i++) {
			System.out.println("Outer loop count: " + i);
			System.out.print("    Inner loop count:");
			int t = 0;
			while( t < 100) {
				if( t == 10) break;
				System.out.print(t + " ");
				t++;
			}
			System.out.println();
		}
		System.out.println("Loops complete.");
	}
}
```

Plan

Syntax

```
break label;
```

- label is the name of a label that identifies a block of code.

```java
// Using break with a label.
public class Break4 {
    public static void main(String[] args) {
        int i;
        for(i = 1; i < 4; i++) {
            one:{
            two:  {
            three:  {
                        System.out.println("\n i is " + i);
                        if(i == 1) break one;
                        if(i == 2) break two;
                        if(i == 3) break three;
                        // this is never reached
                        System.out.println("won't reach.");
                    } System.out.println("After block
                                                        three.");
                } System.out.println("After block two.");
            } System.out.println("After block one.");
        } System.out.println("After for.");
    }
}
```

```
 i is 1
After block one.

 i is 2
After block two.
After block one.

 i is 3
After block three.
After block two.
After block one.
After for.
```

```java
// This program contains an error.
public class BreakErr {
        public static void main(String[] args) {
                one: for(int i = 0; i < 3; i++) {
                                System.out.print("Pass " + i + ": ");
                          }
                for(int j =0; j < 100; j++) {
                        if(j == 10) break one; // WRONG
                        System.out.print(j + " ");
                }
        }
}
```

Plan

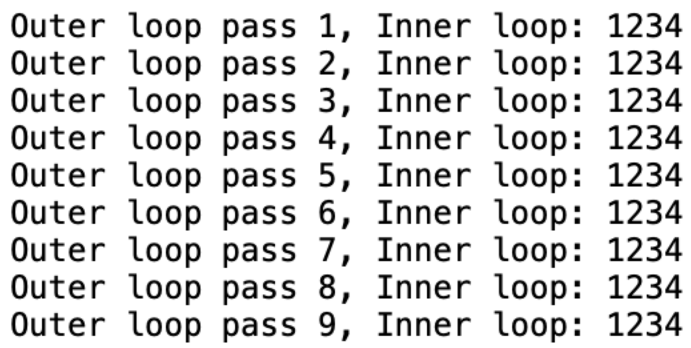
continue statement

- Forces the next iteration of the loop to take place, skipping any code between itself and the conditional expression that controls the loop.

Example

```
// Use continue.
public class ContDemo {
    public static void main(String[] args) {
        int i;
        // print even numbers between 0 and 100
        for(i = 0; i <= 100; i++) {
            if ((i%2) != 0) continue; // iterate
            System.out.println(i);
        }
    }
}
```

```java
// Use continue with a label.
public class ContToLabel {
    public static void main(String[] args) {
    outerloop:
        for(int i = 1; i < 10; i++) {
            System.out.print("\nOuter loop pass " + i +
            ", Inner loop: ");;
            for(int j = 1; j < 10; j++) {
                // continue outer loop
                if(j == 5) continue outerloop;
                System.out.print(j);
            }
        }
    }
}
```



```
Outer loop pass 1, Inner loop: 1234
Outer loop pass 2, Inner loop: 1234
Outer loop pass 3, Inner loop: 1234
Outer loop pass 4, Inner loop: 1234
Outer loop pass 5, Inner loop: 1234
Outer loop pass 6, Inner loop: 1234
Outer loop pass 7, Inner loop: 1234
Outer loop pass 8, Inner loop: 1234
Outer loop pass 9, Inner loop: 1234
```

Plan

- One loop can be nested inside of another.
- Nested loops are used to solve a wide variety of programming problems and are an essential part of programming.

Example

```java
/* Use nested loops to find factors of numbers
 * between 2 and 100.
 */
public class FindFac {
	public static void main(String[] args) {
		for(int i = 2; i <= 100; i++) {
			System.out.print("Factors of " + i + ": ");
			for(int j = 2; j < i; j++)
				if((i%j) == 0) System.out.print(j + " ");
			System.out.println();
		}
	}
}
```

QUESTION ?