

## Control Flow

---

---

---

---

---

---

---

---

## Control Flow

- Sequential Flow: Statements (review), blocks of statements
- Selection
  - if statement (one way)
  - if/else (two ways)
  - Nested if/else
  - Multi-way if/else
  - switch/case (multi-way)

---

---

---

---

---

---

---

---

## Control Flow

- Repetition (loop)
  - while loop
  - for loop
  - do/while loop
  - break and continue
  - break / continue with label
- Combination of statement blocks

---

---

---

---

---

---

---

---

## Statements

- Sequence of execution of a program is controlled by *statements*
- A sequence of statements and local variable declarations can be combined as a single statement by a *block* of { and } braces
- Declarations can be made anywhere within a block as long as they are *declared before use*.

---

---

---

---

---

---

---

---

## Sequence of statements

```
class Celsius {  
    public static void main(String[] args) {  
        double f;  
        f = 67.3;  
        double c;  
        c = ((double)5)/9 * (f-32);  
        System.out.println("F "  
            + f + " = "  
            + c + " degree Celsius");  
    }  
}
```

SEQUENCE CONTROL STRUCTURE



---

---

---

---

---

---

---

---

## Selection: if statement

- Syntax:  
if (conditional\_expression)  
 statement\_to\_be\_executed\_if\_true;  
rest\_of\_code;
- Or  
if (conditional\_expression) {  
 statement1;  
 ...  
 statementn;  
}

---

---

---

---

---

---

---

---

## statement;

- First, *conditional\_expression* is evaluated
- If expression evaluates to true then *Statement* is executed
- If expression evaluates to false then no further action is taken and *Statement* is not executed
- The expression must be a boolean
  - C style: `if (someValue)` and `if (1)` is harmful

---

---

---

---

---

---

---

---

## if statement example

```
int score = 8;           # score = 8
char gpa = 'D';         # gpa = D
if (score >= 7)         # true
    gpa = 'C';         # gpa = C
if (score >= 8)         # true
    gpa = 'B';         # gpa = B
if (score >= 9)         # false
    gpa = 'A';         # false
scr.print(gpa);        # display gpa as B
// value of gpa?      # is it the best way?
                       # What if we reverse
                       # order?
```

---

---

---

---

---

---

---

---

## if statement example2

```
int score = 8;           # score = 8
char gpa = 'D';         # gpa = D
if (score >= 9)         # false
    gpa = 'A';         #
if (score >= 8)         # true
    gpa = 'B';         # gpa = B
if (score >= 7)         # true
    gpa = 'C';         # gpa = C
scr.print(gpa);        # display gpa=C: WRONG!
// value of gpa?
```

---

---

---

---

---

---

---

---

## Warning

- What's wrong with this code?

```
double PI=Math.PI;
if (PI == 3.1415927) {
    System.out.println("I got a Pie!");
}
```

---

---

---

---

---

---

---

---

## Selection: if/else

- Syntax:

```
if (conditional_expression)
    statement_to_be_executed_if_true;
else
    statement_to_be_executed_if_false;
rest_of_code;
```

- Or

```
if (conditional_expression) {
    true_block;
} else {
    false_block;
}
rest_of_code;
```

---

---

---

---

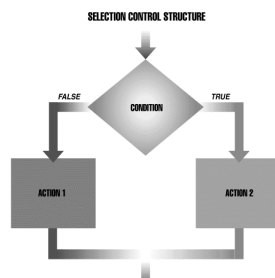
---

---

---

---

## if (exp) statement1; else statement2;



- First, *expression* is evaluated
- If expression evaluates to true then *Statement1* is executed
- If expression evaluates to false then *Statement2* is executed

---

---

---

---

---

---

---

---

## if/ else statement example

```
int score = 76;           # score = 76
String grade;            # grade declared
if (score < 69) {        # false
    grade=new String("NC");  #
    scr.print("Failed");    #
} else {                 # false block start here
    grade=new String("CR");  # assign grade
    scr.print("Passed");    # display "Passed"
}                        # end block
scr.print(" with "+grade); # display " with CR"
```

---

---

---

---

---

---

---

---

## Nested if, if/else

```
String grade;
... //declaration and init. of finalscore & midterm
if (finalscore>=90)
    if (midterm>=90)
        grade="A+"; // same as grade=new String("A+");
    else
        grade="A";
else if (finalscore>=80)
    grade="B";
else if (finalscore>=70)
    grade="C";
else
    grade="F";

scr.print("grade="+grade);
```

---

---

---

---

---

---

---

---

## Who owns the else?

```
if (a>=5)                if (a>=5) {
    c = c+1;              c = c+1;
    if (b<20)             if (b<20)
        b = 2;           b = 2;
else                       } else {
    d = 5;                d = 5;
                          } // why the second set
                          of {} braces
```

---

---

---

---

---

---

---

---

## Nested if, if/else

```
String grade;
... //declaration and init. of finalscode & midterm
if (finalscore>=90) {
  if (midterm>=90) {
    grade="A+"; // same as grade=new String("A+");
  } else {
    grade="A";
  }
} else {
  if (finalscore>=80) {
    grade="B";
  } else if (finalscore>=70) {
    grade="C";
  } else {
    grade="F";
  }
}
scr.print("grade="+grade);
```

---

---

---

---

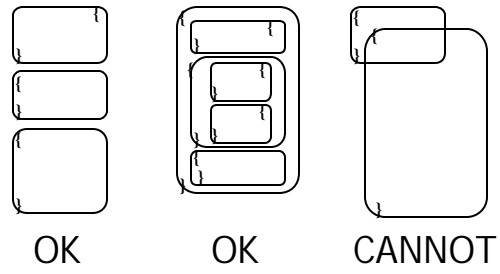
---

---

---

---

## Block rules



---

---

---

---

---

---

---

---

## Better Formatting: multi-way if/else if

```
String grade;
... //declaration and init. of finalscode & midterm
if (finalscore>=90) {
  if (midterm>=90) {
    grade="A+"; // same as grade=new String("A+");
  } else {
    grade="A";
  }
} else if (finalscore>=80) {
  grade="B";
} else if (finalscore>=70) {
  grade="C";
} else {
  grade="F";
}
scr.print("grade="+grade);
```

---

---

---

---

---

---

---

---

## multi-way if/ else example

```
int score = 8;           # score = 8
char gpa = ' ';         # gpa declared, initialized
if (score >= 9) {       # false
    gpa = 'A';          ↵
} else if (score >= 8) { # true
    gpa = 'B';         # gpa = B
} else if (score >= 7) { ↵
    gpa = 'C';        ↵
} else {               ↵
    gpa = 'F';        ↵
}
scr.print("grade="+gpa); # display gpa as B
// value of gpa?        # What is advantage of the
                        # additional elses?
```

---

---

---

---

---

---

---

---

## Multi-way if/else advantage

```
int score = 8;           ✓
char gpa = ' ';         ✓
if (score >= 9) {       ✓
    gpa = 'A';          ✓
} else if (score >= 8) { ✓
    gpa = 'B';         ✓
} else if (score >= 7) { ○
    gpa = 'C';        ○
} else {               ○
    gpa = 'F';        ○
}
scr.print("grade="+gpa); ✓
// value of gpa?      ○
```

```
int score = 8;           ✓
char gpa = 'D';        ✓
if (score >= 9) {       ✓
    gpa = 'A';          ○
} if (score >= 8) {    ✓
    gpa = 'B';         ✓
} if (score >= 7) {   ✓
    gpa = 'C';        ○
} scr.print(gpa);     ✓
// value of gpa?     ○
```

---

---

---

---

---

---

---

---

## multi-way if/else example

```
int score = 8;           ✓ score = 8
char gpa = ' ';         ✓ gpa declared, initialized
if (score >= 7) {       ✓ true
    gpa = 'C';         ✓ gpa = 'C'
} else if (score >= 8) { ✓ skip the else block
    gpa = 'B';        ○
} else if (score >= 9) { ○
    gpa = 'A';        ○
} else {               ○
    gpa = 'F';        ○
}                      ○
scr.print("grade="+gpa); ○
// value of gpa?      ✓ display gpa=C still WRONG
```

---

---

---

---

---

---

---

---

## selection: if/ else example

- IfStatementEx.java

---

---

---

---

---

---

---

---

## switch statement

- Syntax:

```
switch ( value ) {  
  case value1:   
    statements;  
    ...  
    break;  
  case valueN:   
    statements;  
    ...  
    break;  
  default:   
    statements;  
    ...  
}
```

---

---

---

---

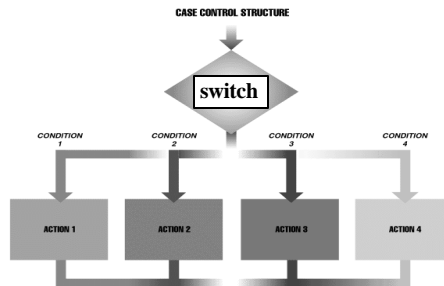
---

---

---

---

## switch/case statement



---

---

---

---

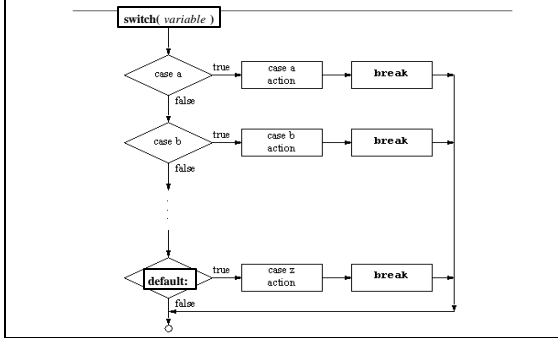
---

---

---

---

# Switch Statement



---

---

---

---

---

---

---

---

## Switch Example

### ■ SwitchStatementEx.java

```
switch( grade ) {  
  case 'A':  
    message = "You got an A!\nYou're gonna get paid(???)!";  
    break;  
  case 'B':  
    message = "You got a B!\nNot bad!";  
    break;  
  case 'C':  
    message = "You got an C!\nStudy a little more!";  
    break;  
  default:  
    message = "You failed!\n"; // message string  
} // end switch
```

---

---

---

---

---

---

---

---

## while statement syntax

```
while (<continuation condition expression>)  
  <statement>;
```

```
while (<continuation condition expression>) {  
  <statement>;...<statement>;  
}
```

---

---

---

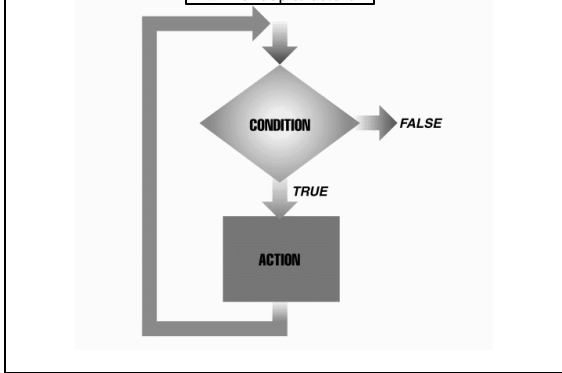
---

---

---

---

---



---

---

---

---

---

---

---

---

## Live Code example: while

---

---

---

---

---

---

---

---

## Counter-Control Loop

- Count-controlled loops are those loops that are executed a fixed number of times. The number of iterations is known before commencing the loop.

```
i = 1;
while (i <= 5) {
    System.out.println("Square of " + i + " is " + (i * i));
    ++i;
}
```

---

---

---

---

---

---

---

---

# Sentinel-Control Loop

- Event-controlled loops are those loops that are executed an indefinite number of times until some condition occurs.

```
EasyIn easy = new EasyIn ();
final int SENTINEL = -1;
int posInt; // input: positive integer value read
int sum = 0; // output: sum of the integers read in
System.out.println("Enter number: "); posInt = easy.readInt();
while (posInt != SENTINEL) {
    sum += posInt;
    System.out.println("Enter number: ");
    posInt = easy.readInt();
}
```

---

---

---

---

---

---

---

---

## do/while statement syntax

```
do {
    <statement>;...<statement>;
} while (<continuation condition expression>;);
```

---

---

---

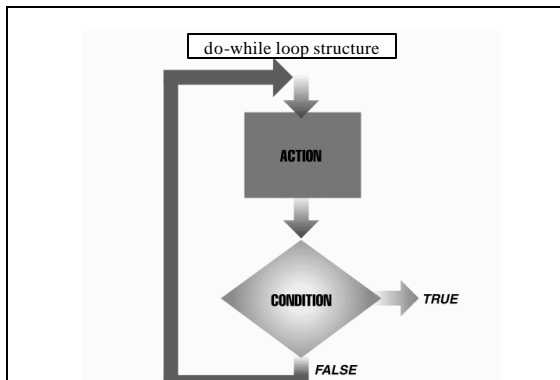
---

---

---

---

---



---

---

---

---

---

---

---

---

## Live Code example: do/ while

---

---

---

---

---

---

---

---

## for loop structure syntax

```
for (<initializing statement>,...;  
    <continuation assertion>;  
    <increment statement> )  
{  
    <body statement>;  
    ...<body statement>;  
}
```

---

---

---

---

---

---

---

---

## Live Code example: for

```
■ SampleCode.java  
do {  
    System.out.println("Display #" + i);  
    i += 2;           // #2 change of counter  
} while ( i <= 10 ); // #3 checking of counter  
  
int j;  
for ( j = 0; j <= 10; j += 2 ) {  
    System.out.println("For Loop Display #" + j);  
}
```

---

---

---

---

---

---

---

---

## Break and Continue

- **break** quits the loop without executing the rest of the statements in the loop.
  - Used in switch statements
  - Can be used in loops, but not common
  - interrupts the current loop
- **continue** stops the execution of the current iteration and goes back to the beginning of the loop to begin the next iteration. Executes if continuation condition is true.
  - used with loops, but not common
  - interrupts the current loop

---

---

---

---

---

---

---

---

## Break Example

```
for(int i = 0; i < 10; i++) {  
    if( i == 5 )  
        break;  
    System.out.println("i is " + i );  
}
```

} i only lives in the for loop. This is known as variable scope.

Output:  
i is 0  
i is 1  
i is 2  
i is 3  
i is 4

---

---

---

---

---

---

---

---

## Break/Continue with Label Example

```
top:  
for(int j = 0; j < 10; j++) {  
    System.out.println("outer: j is " + j);  
    for(int i = 0; i < 10; i++) {  
        if( i == 3 )  
            continue top;  
        if( j == 2 && i == 2 )  
            break top;  
        System.out.println("inner: i is " + i );  
    }  
}
```

Output:  
outer: j is 0  
 inner: i is 0  
 inner: i is 1  
 inner: i is 2  
outer: j is 1  
 inner: i is 0  
 inner: i is 1  
 inner: i is 2

---

---

---

---

---

---

---

---

## Continue Example

```
for(int i = 0; i < 10; i++) {  
    if( i < 5 )  
        continue;  
    System.out.println("i is " + i );  
}
```

i only lives in the for loop. This is known as variable scope.

Output:  
i is 5  
i is 6  
i is 7  
i is 8  
i is 9

---

---

---

---

---

---

---

---

## Keywords - continue and break

- continue
  - Causes the program to abandon the current iteration and start the next iteration of a loop.
  - Use continue to skip input values that do not need processing.
- break
  - Causes the program to continue control at the statement just beyond the end of the innermost block.
  - Use break to leave the execution of a particular iteration upon an unusual error condition.

---

---

---

---

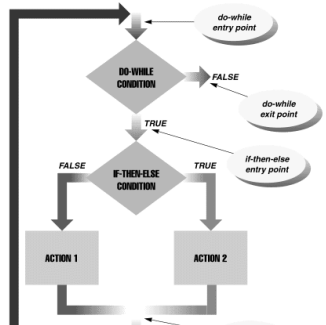
---

---

---

---

## Compound Structure



---

---

---

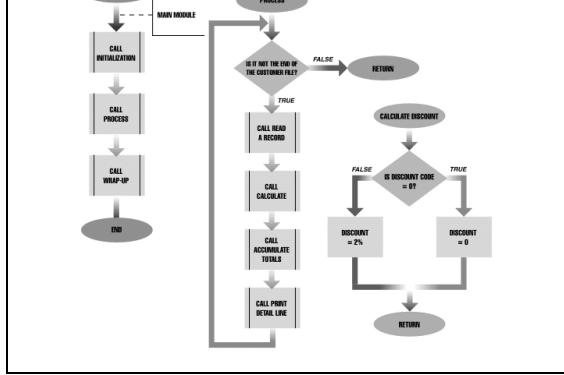
---

---

---

---

---




---



---



---



---



---



---

## More Examples

- NestedIfStatementEx.java
- LoopEx.java
- Loop2Ex.java
- FiboEx.java

---



---



---



---



---



---