Chapter 3
Program control statements

Based on the course literature:

Java: A beginner's guide

Sixth Edition

Herbert Schildt

What we'll cover

- Keyboard input
- if, switch, for, while, do-while
- nested loops
- break, continue

Input from

```
char ch  = (char) System.in.read();
```

Demo 1

Statements

selection statements:
if, switch

iteration statements:
for, while, do-while

jump statements:
break, continue, return

The return of the if

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

if (condtion)
{
        statements;
}
else
{
        statements;
}
```

Exercise

1) **Think of a character between a-z.**
2) **Save that character in a suitable variable.**
3) **Prompt the user to guess the character you are thinking of.**
4) **Read in the users guess and save it to a variable.**
5) **If the user guessed correctly display “correct”.**
6) **Otherwise display “wrong”.**

Nested ifs

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

Exercise

1) **If the guess was wrong:**
2) **Check that the guess was between a and z.**
   If it wasn't within that range you print tell them.
3) **If the guess was in the correct range let the player no if they guessed higher than the character you thought of or lower.**

if-else-if ladder

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

Demo 2

switch

```
switch (expression) {
        case constant1:
                statements;
                break;
        case constant2:
                statements;
                break;
        default:
                statements;
}
```

Special switch in Java

- Switches don’t work with all variable types in Java.
- Supported types:
  - byte, short, int, char and enumerator
  - String (As of JDK 7)

Demo 3 & 4

nested switches

```
switch (ch1) {
        case ‘A’:
                switch (ch2) {
                        case ‘A’:
                        break;
                }
                break;
}
```

ARCTICTIGER

The for loop

The for loop is best used when you know the number of iterations to perform. I.e. looping through an array etc.

for(intialization; condition; iteration) statement;

for (count = 10; count < 5; count++) statement;

// The above statement will never be triggered.

// as the condition is never met;

for

```
for (x = 100; x > -100; x -= 5) statement;
// The above is totally ok

for ( i = 0, j = 0; i < j; i++, j- -) statement;
// It's also ok to initialise and increment more
// than one variable

j=0;
for (i = 0; i < j; i++){
        j- -;
}
```

for

```
for (i = 0; (char) System.in.read() != 'S'; i++)
        System.out.println("Pass #" + i);

for (i = 0; i < 10;) {
        System.out.println("Pass #" + i);
        i++; // increment loop control
}

int i;
for (; i < 10; ){
        i++;
}
```

ARCTICTIGER

Demo 5

for

```
for(;;;){}

for(i = 1; i <= 5; sum += i++);

vs

for(i = 1; i <=5; i++){
        sum += i;
}
```

for

```
for(;;;){}

for(i = 1; i <= 5; sum += i++);

vs

for(i = 1; i <=5; i++){
        sum += i;
}
```

while

When the loop will loop an unknown number of times.

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

The condition can be any valid Boolean expression.

The loop repeats while the condition is true.

Demo 6

do-while

A loop that will always perform at least one iteration.

```
do {
        System.out.print(“Press a key followed by enter: “);
        ch = (char) System.in.read();
} while(ch != ‘q’);
```

Exercise

1) **Rewrite the code from the earlier exercise so that the user will be able to guess until they have guessed correctly. (hint do-while loop)**

break

```
for (int i = 0; i < users.length; i++){
        if(users[i].getStatus() == “super”){
                currentUser = users[i];
                break;
        }
}
```

In the example above when we have found a super user we use the break statement to prematurely exit the loop.

break

```
for(intialization; condition; increment){
        while(condition){
                …
                break;
        }
}
```

break

```
main : for(intialization; condition; increment){
        while(condition){
                ...
                break main;
        }
}
```

Labels can be used to identify a loop.

Demo 7

continue

Continue is used when we want to ignore the rest of the code in our current iteration and force the next iteration.

```
for(i = 0; i <= 100; i++){
        if((i%2) !=0) continue;
        System.out.println(i);
}
```

break

```
main : for(intialization; condition; increment){
        while(condition){
                …
                continue main;
        }
}
```

Labels can be used to identify a loop.

Demo 8