Processing: Looping and Conditionals

The while Loop:

A while loop is a control structure that allows you to repeat a task a certain number of times.

Syntax:

The syntax of a while loop is:

```
while(Boolean_expression) {     //Statements }
```

The do...while Loop:

A do...while loop is similar to a while loop, except that a do...while loop is guaranteed to execute at least one time.

Syntax:

The syntax of a do...while loop is:

```
do {     //Statements }while(Boolean_expression);
```

The for Loop:

A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

A for loop is useful when you know how many times a task is to be repeated.

Syntax:

The syntax of a for loop is:

```
for(initialization; Boolean_expression; update) {     //Statements }
```

Enhanced for loop in Java:

As of java 5 the enhanced for loop was introduced. This is mainly used for Arrays.

Syntax:

The syntax of enhanced for loop is:

```
for(declaration : expression) {     //Statements }
```

The break Keyword:

The *break* keyword is used to stop the entire loop. The break keyword must be used inside any loop or a switch statement.

The break keyword will stop the execution of the innermost loop and start executing the next line of code after the block.

The continue Keyword:

The *continue* keyword can be used in any of the loop control structures. It causes the loop to immediately jump to the next iteration of the loop.

- In a for loop, the continue keyword causes flow of control to immediately jump to the update statement.
- In a while loop or do/while loop, flow of control immediately jumps to the Boolean expression.

Syntax:

The syntax of a continue is a single statement inside any loop:

```
continue;
```

The if Statement:

An if statement consists of a Boolean expression followed by one or more statements.

Syntax:

The syntax of an if statement is:

```
if(Boolean_expression) {    //Statements will execute if the Boolean expression
is true }
```

The if...else Statement:

An if statement can be followed by an optional *else* statement, which executes when the Boolean expression is false.

Syntax:

The syntax of a if...else is:

```
if(Boolean_expression){    //Executes when the Boolean expression is true
}else{    //Executes when the Boolean expression is false }
```

The if...else if...else Statement:

An if statement can be followed by an optional *else if...else* statement, which is very usefull to test various conditions using single if...else if statement.

Syntax:

The syntax of a if...else is:

```
if(Boolean_expression 1){    //Executes when the Boolean expression 1 is true
}else if(Boolean_expression 2){    //Executes when the Boolean expression 2 is
true }else if(Boolean_expression 3){    //Executes when the Boolean expression
3 is true }else {    //Executes when the one of the above condition is true. }
```

Nested if...else Statement:

It is always legal to nest if-else statements. When using if , else if , else statements there are few points to keep in mind.

- An if can have zero or one else's and it must come after any else if's.
- An if can have zero to many else if's and they must come before the else.
- Once an else if succeeds, none of he remaining else if's or else's will be tested.

Syntax:

The syntax for a nested if...else is as follows:

```
if(Boolean_expression 1){    //Executes when the Boolean expression 1 is true
if(Boolean_expression 2){       //Executes when the Boolean expression 2 is
true    } }
```

The switch Statement:

A *switch* statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.

Syntax:

The syntax of enhanced for loop is:

```
switch(expression){     case value :        //Statements        break;
//optional      case value :        //Statements        break; //optional
//You can have any number of case statements.     default : //Optional
//Statements }
```