Control structures and loops

Control structures for decision making

- **Decision Making in java is done using if, if-else, switch-case**
- **Selection statements:**
  - if
  - if-else
  - nested-if
  - if-else-if ladder
  - switch-case
- **Jump Statements**
  - break
  - continue
  - Return

if statement

- **if syntax:**

```
if (condition)   {
 // Statements to execute if condition is true
}
```

- **if-else syntax:**

```
if  (condition) {
// Executes this block if condition is true
} else {
// Executes this block if condition is false
}
```

- **Nested if syntax:**

```
if (condition1)  {
// Executes when condition1 is true
        if (condition2)  {
          // Executes when condition2 is true
        }
}
```

if-else-if ladder

If-else-if syntax :

```
if (condition1) {
   statements;
} else if (condition2) {
   statement;
} else if (condition3) {
   statement;
} else {
   statement;
}
```

Switch- case

switch-case: The switch statement is a multiway branch statement. It provides an easy way to dispatch execution to different parts of code based on the value of the expression.

```
switch (expression) {
 case value1:
  statements;
  break;
 case value2:
  statements;
  break;
  case valueN:
  statements;
  break;
 default:
  statements;
}
```

- The expression can be of type byte, short, int char, or an enumeration. Beginning with JDK7, *expression* can also be of type String.
- Duplicate case values are not allowed.
- The default statement is optional.
- The break statement is used inside the switch to terminate a statement sequence.
- The break statement is optional. If omitted, execution will continue on into the next case.

Jump statements

- Java supports three jump statements: **break, continue** and **return**. These three statements transfer control to another part of the program.
  - **break:** In Java, a break is majorly used for:

    Terminate a sequence in a switch statement

    To exit/break a loop.
  - **continue:** It is useful to continue the loop skipping statements below continue statement.
  - **return** : It is used to return value or control to calling function.

Demo programs/ assignments

- Program to find maximum number of three numbers
- Print if number is odd or even
- Print passing class for Student based on marks entered

  40 to < 50 - Pass class

  50 to < 60 - Second class

  60 to < 75 - First class

  75+ - Distinction

- Print each number in words from 1 to 10 using switch case.

Loops

- **Loops are used to perform repetitive tasks by executing statement written inside loop**

while loop:

- Its entry controlled loop
- A while loop is a control flow statement that allows code to be executed repeatedly based on a given boolean condition.

Syntax:

```
while (condition) {
  loop statements...
}
```

Loops

- **for loop:**
  - Its entry controlled loop
  - for loop provides a concise way of writing the loop structure.
  - Unlike a while loop, a for statement consumes the initialization, condition and increment/decrement in one line thereby providing a shorter, easy to debug structure of looping.

Syntax:

```
for (initialization; testing condition; increment/decrement) {
    statement(s)
}
```

Loops

- **do while:**
  - Exit controlled Loop
  - do while loop is similar to while loop with only difference that it checks for condition after executing the statements

Syntax:

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

Recusion

- Calling a method within itself
- Recursive methods can replace loops
- Recursive function should have halting condition otherwise the methods will be called infinite time.

Ex.

```
//calling a method within itself
     public static int factorial(int no) {
      if(no == 1) {
          return 1;
      }
      return no * factorial(no - 1);
     }
```

Demo programs/ assignments

- Print n numbers using loops
- Print Fibonacci series up to n numbers.
- Print factorial of input number with loop and recursion
- Print below patterns