The Program Flow

Session Objectives

- Upon completing this session you should be able to:
  - Develop code that implements if and switch statement; and identify legal argument types for these statements.
  - Develop code that implements all forms of loops and iterators, including the use of for, the enhanced for loop (for-each), do, while, labels, break, and continue; and explain the values taken by loop counter variables during and after loop execution.

Introduction

- The execution of the statements in a program takes place from top to bottom.
- It is possible to alter this flow, so that different sections of our programs can be ignored while other sections are repeated again and again.
- Different kinds of statement have different effects in looping like decision-making statements (if-then, if-then-else, switch), the looping statements (for, while, do-while), and the branching statements (break, continue, return)

Flow Control

- Specify order in which statements are to executed
- A control structure is a control statement and the statements whose execution it controls
- Three structures control program execution:
  - Sequence
  - Selection or decision structure
  - Iteration or looping structure

a) Sequence

- These consist of execution of statements one at a time in the sequences they appear
- In sequential programming code is executed from top to bottom.
- It's very linear, in that each and every line of code will be read, starting with the first line of code you write and ending at the last line.

b) Selection

- In a conditional or branching or selection control structure, a portion of the program is executed once depending on what conditions occur in the code.
- A logical test using logical and relational operators may require to be used in order to determine which actions to take (subsequent statements to be executed) depending on the outcome of the test.

a) The if statement

- **Syntax of a simple if statement**

```
if (condition) statement1;
```

- The general form of the statement is:

```
if (condition) statement1;
else statement2;
```

- The above format shows that an else statement will be executed whenever an if statement evaluates to false. For instance

```
if (choice==’m’)
answer=num1*num2;
else
answer=num1/num2;
```

- A complex if statement would allow us to be more specific and execute multiple statement, depending on the condition , this statement is known as the nested if statement

```
if (choice==’m’)
answer=num1*num2;
else if(choice==’d’)
answer=num1/num2;
else if(choice===’a’)
answer=num1+num2;
else
answer=num1-num2;
```

Example

```
package salaryprogram;
import java.util.Scanner;
public class SalaryProgram {
public static void main(String[] args) {
Scanner userInput = new Scanner(System.in);
System.out.println("Enter the amount:");
int sal_amt = userInput.nextInt();
Double Disc;
if(sal_amt>=10000)
System.out.println ("The discount is"+ sal_amt*0.10) //*ten percent/
else if (sal_amt>=5000&&sal_amt<10000 )
System.out.println ("The discount is"+sal_amt*0.07); /*seven percent */
else if((sal_amt>=3000)&&(sal_amt<=5000)){
Disc = sal_amt * 0.05; /* five percent */
System.out.println ("The discount is"+Disc);
}
else
System.out.println ("The discount is 0") ;}
```

- }

Exercises

1. Write that accepts students marks and prints out the corresponding letter grade.
2. Write a program that accepts user input from the console. The programme should take a number and then test for the following age ranges: 0 to 10, 11 to 20, 21 to 30, 30 and over. Display a message in the Output window in the following format: user_age + " is between 21 and 30"

3. Write a program to calculate the monthly telephone bills as per the following rule:

- Minimum ksh 200 for upto 100 calls.
- Plus ksh. 0.60 per call for next 50 calls.
- Plus ksh 0.50 per call for next 50 calls.
- Plus ksh 0.40 per call for any call beyond 200 calls.

b)The switch statement

- Sometimes it becomes cumbersome to write lengthy programs using if and if-else statements. To avoid this we can use Switch statements in Java.
- The switch statement in Java is the best way to test a single expression against a series of possible values and executing the code.
- Here is the general form of switch statement:

```
switch (expression){
case 1: code block1
case 2: code block2
.
.
 default: code default; }
```

- The expression to the switch must be of a type byte, short, char, or int. Then there is a code block following the switch statement that comprises of multiple case statements and an optional default statement.
- The execution of the switch statement takes place by comparing the value of the expression with each of the constants. The comparison of the values of the expression with each of the constants occurs after the case statements. Otherwise, the statements after the default statement will be executed. Now, to terminate a statement following a switch statement use break statement within the code block.
- if we won't use break statement the computer will go ahead to execute the statements associated with the next case after executing the first statement.

Example

```
package Days;
import java.util.Scanner;
public class Days {
public static void main(String args[]){
Scanner input = new Scanner(System.in);
System.out.println("Enter a number 1-7: ");
int day = input.nextInt();
switch(day){
case 1:System.out.println("Monday!");break;
case 2: System.out.println("Tuesday!");break;
case 3:System.out.println("Wednesday");break;
case 4 :System.out.println("Thursday");break;
case 5:System.out.println("Friday");break;
case 6 :System.out.println("Saturday");break;
case 7 :System.out.println("Sunday");break;
 default :System.out.println("Invalid Number. Please enter 1-7");}
}
}
```

Example

```
package exam;
import java.util.Scanner;
public class Exam {
public static void main(String args[]){
Scanner input = new Scanner(System.in);
System.out.println("Enter the a grade:");
//char grade = input.nextInt(); //this wouldn’t work
char grade = input.next().charAt(0);//notice use of input.next().charAt(0);//
switch(grade){
case 'A' :System.out.println("Excellent!");break;
case 'B' :System.out.println("Well done");break;
case 'C' :System.out.println("Good Trial");break;
case 'D' :System.out.println("You passed");break;
case 'F' :System.out.println("Better try again");break;
default :System.out.println("Invalid grade");
}
System.out.println("Your grade is " + grade);
}
}
```

- **NB: There is no classical nextChar() method in Java Scanner class. Hence the use of the next().charAt(0).**

Exercise

1. With the help of a switch statement, write a JAVA code that prints the month name corresponding to a month number.
2. Write a program that asks a user to choose between four colours: white, red, or blue. Use Switch statements to display one of the following messages, depending on which colour was chosen:

- WHITE "You are a Spurs fan"
- RED "You are a Red Devil"
- BLUE "You belong to Chelsea"

Flow Control -Iteration

- These are a group of statements in a program that may have to be executed repeatedly until some condition is satisfied.
- In a looping control structure, series of statements are repeated. This is known as looping. Suppose you were asked to display "Hello World!" five times on the screen. How would you write the above program?
- All looping involves repeatedly executing one or more statements until a certain condition is met.
- In a pre-test loop, the condition is checked before the loop statements are executed.
- In a post-test loop, the condition is checked after the statements in the loop have been executed.
- Java supports three loop versions:
  - while loop
  - do while loop
  - for loop.

The for loop

The for Statement

- It is used to execute a block of code continuously to accomplish a particular condition. For statement consists of three parts i.e. initialization, condition, and iteration.
  - **Initialization**: It is an expression that sets the value of the loop control variable. It executes only once.
  - **Condition**: This must be a boolean expression. It tests the loop control variable against a target value and hence works as a loop terminator.
  - **Iteration**: It is an expression that increments or decrements the loop control variable. Here is the form of the for loop:

```
for(initialization; condition; iteration)
{ //body of the loop }
```

For example, a sample for loop may appear as follows:

```
    int i;
    for (i=0; i<10; i++)
    System.out.println("i = " +i);
```

Example

- The program to print even numbers between 1 and 20

```
public class myTest {
public static void main(String args[]) {
 for(int x = 1; x < 20; x++) {
if(x%2==0)
System.out.print(x );
System.out.print("\n");
}
}
}
```

The while Statement

- In a simpler language, the while statement continually executes a block of statements while a particular condition is true.
- The while statement is used to carry out looping instructions where a group of instructions executed repeatedly until some conditions are satisfied.
- To write a while statement use the following form:

```
while (expression)
{ statement(s) }
```

Example

- The program below displays numbers 1-5 using while do:

```
class myNumbers
{
public static void main (String[] args){
int number = 1;
while (number < = 5){
System.out.println("The value of number is : " + number); number =
number + 1;
}
}
}
```

While Example2

- The program below displays numbers 1-5 using while do:

```
class TestClass
{
public static void main (String[] args)
{
int number = 1;
while (number < = 5)
{
System.out.println("The value of number is : " + number);
number = number + 1;
}
}
}
```

The ‘do .. while’ loop

- It is used when the loop condition is executed at the end of each loop pass.
- The do-while loop does not check until the loop body is executed once.
- It takes the form:

do statement;

while(expression);

- The statement (simple or compound) will be executed repeatedly as long as the value of the expression is true. (i.e. non zero).
- do … while evaluates the variable choice at the bottom, the loop has to at-least execute once before realizing that choice is false

Example

- Program prints numbers between 0 and 5 using do while

```
public class DoWhileExample {
public static void main(String[] args) {
int i = 0;
do {
System.out.println(“The number is: " + i);
i++;
}
while (i < 5);
}
}
```

Comparing for and while

- **While loop**

```
int x = 1; // initial statement
while (x <= 10) { // conditional
System.out.println(x); //body
x = x + 1; // increment
}
```

- **Corresponding for loop**

```
//for (initial stmt; conditional; increment)
for (int x = 1; x <= 10; x = x + 1){
System.out.println(x); //body
}
```

- Both for and while loops check the condition before entering the loop for the first time

Jumping

- Using jumping statements like break and continue it is easier to jump out of loops to control other areas of program flow.
- **The break Statement**
- We use break statement to terminate the loop once the condition gets satisfied.
- **The continue statement**
- The continue statement is used in many programming languages such as C, C++, java etc. Sometimes we do not need to execute some statements under the loop then we use the continue statement that stops the normal flow of the control and control returns to the loop without executing the statements written after the continue statement.
- There is the difference between break and continue statement that the break statement exit control from the loop but continue statement keeps continuity in loop without executing the statement written after the continue statement according to the conditions.
- do … while evaluates the variable choice at the bottom, the loop has to at-least execute once before realizing that choice is false

- **Break statement**
- Causes immediate exit from control structure
- o Used in while, for, do…while or switch statements

- **Continue statement**
- Skips remaining statements in loop body
- Proceeds to next iteration
- o Used in while, for or do…while statements

```
for (int x = 0; x < 5; x++) //continue goes to here
{
System.out.println(" x < 10 : " + x);
if (x > 5) {continue;} //goes back to top of loop
if (x > 10) {break;} //breaks out of loop
}
//break goes to here
for (int x = 0; x < 5; x++) //outer loop
{
System.out.println("Outer: " + x); 9
for (int y = 0; y < 5; y++) //inner loop - continue goes back to here
{
System.out.println("Inner: " + y);
if (x > 2) {continue;} //goes back to top of inner loop
 if (x > 3) {break;} //breaks out of inner loop
}
//breaks goes back to here, remains in outer loop
}
```

Exercises

1. Using a nested if statement, write a program that prompts the user for a number and then reports if the number is positive, zero or negative.
2. Write a while loop that will calculate the sum of every fourth integer, beginning with the integer 3 (that is calculate the sum 3 + 7 +11 + 15 + ...) for all integers that are less than 30.
3. Write a program that prints only the odd numbers between 1 and 100. Use a for loop.
4. Write an application that reads an integer and determines and prints whether it's odd or even.
5. Write an application that reads five integers and determines and prints the largest and smallest integers in the group.