JAVA

1. Install Java

a. Install JDK (https://www.oracle.com/in/java/technologies/javase-downloads.html)

b. Install IntelliJ (https://www.jetbrains.com/idea/download/#section=mac)

2. Sample Code

Functions

A function is a block of code which takes some input, performs some operations and returns some output.

The functions stored inside classes are called methods.

The function we have used is called main.

Class

A class is a group of objects which have common properties. A class can have some properties and functions (called methods).

The class we have used is Main.

3. Our 1st Program

```
package com.apnacollege;

public class Main {

    public static void main(String[] args) {
   // Our 1st Program
        System.out.println("Hello World");
    }
}
```

4. Variables

A variable is a container (storage area) used to hold data.

Each variable should be given a unique name (identifier).

```
package com.apnacollege;

public class Main {

    public static void main(String[] args) {
        // Variables
        String name = "Aman";
        int age = 30;
```

```
        String neighbour = "Akku";
        String friend = neighbour;
    }
}
```

5. Data Types

Data types are declarations for variables. This determines the type and size of data associated with variables which is essential to know since different data types occupy different sizes of memory.

There are 2 types of Data Types :

- Primitive Data types : to store simple values
- Non-Primitive Data types : to store complex values

Primitive Data Types

These are the data types of fixed size.

Data Type	Meaning	Size (in Bytes)	Range
byte	2's complement integer	1	-128 to 127
short	2's complement integer	2	-32K to 32K
int	Integer numbers	4	-2B to 2B
long	2's complement integer (larger values)	8	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	Floating-point	4	Upto 7 decimal digits
double	Double Floating-point	8	Upto 16 decimal digits

char	Character	2	a, b, c .. A, B, C .. @, #, $ ..
bool	Boolean	1	True, false

Non-Primitive Data Types
These are of variable size & are usually declared with a 'new' keyword.

Eg : String, Arrays

```
String name = new String("Aman");
int[] marks = new int[3];
marks[0] = 97;
marks[1] = 98;
marks[2] = 95;
```

6. String Class

Strings are immutable non-primitive data types in Java. Once a string is created it's value cannot be changed i.e. if we wish to alter its value then a new string with a new value has to be created.
This class in java has various important methods that can be used for Java objects. These include:

a. Concatenation

```
String name1 = new String("Aman");
String description = new String("is a good boy.");

String sentence = name1 + description;
System.out.println(sentence);
```

b. CharAt

```
String name = new String("Aman");
System.out.println(name.charAt(0));
```

c. Length

```
String name = new String("Aman");
System.out.println(name.length());
```

d. Replace

```
String name = new String("Aman");
System.out.println(name.replace('a', 'b'));
```

e. Substring

```
String name = new String("AmanAndAkku");
System.out.println(name.substring(0, 4));
```

7. Arrays

Arrays in Java are like a list of elements of the same type i.e. a list of integers, a list of booleans etc.

a. Creating an Array (method 1) - with new keyword

```
int[] marks = new int[3];
marks[0] = 97;
marks[1] = 98;
marks[2] = 95;
```

b. Creating an Array (method 2)

```
int[] marks = {98, 97, 95};
```

8. Casting

Casting in java is the assigning values of one type to another. The types being considered here are compatible i.e. we can only assign values of a number type to another type storing numbers (vice-versa is not allowed i.e. floating values cannot be assigned to boolean data types).

Casting in Java is of 2 types:

a. Implicit casting

This casting is done by java implicitly i.e. on its own. It is assigning smaller values to larger data types.

```
float price = 100.00F;
int gst = 18;
float finalPrice = price + gst;
```

b. Explicit casting

This casting is done by the programmer. It is assigning larger values to smaller data types.

```
int price = 100;
float gst = 18.00F;
int finalPrice = price + (int)gst;
```

9. Constants

A constant is a variable in Java which has a fixed value i.e. it cannot be assigned a different value once assigned.

```
package com.apnacollege;
```

```
public class Main {

   public static void main(String[] args) {
  // Constants
       final float PI = 3.14F;
   }
}
```

10. Operators

There are 4 types of operators in Java :

a. **Arithmetic Operators**

Arithmetic operators are just like operators we used in Math. These include:

1. '+' Add
```
int a = 30;
int b = 40;
int sum = a + b;
```

2. '-' Subtract
```
int a = 30;
int b = 40;
int diff = a - b;
```

3. '*' Multiply
```
int a = 30;
int b = 40;
int mul = a * b;
```

4. '/' Divide
```
int a = 30;
int b = 40;
int div = a / b;
```

5. '%' Modulo - Remainder of a/b
```
int a = 30;
int b = 40;
int modulo = a % b;
```

6. Unary Operators
```
int a = 30;
a++;
a--;
```

Pre-incrementer : It increments the value of the operand instantly.

Post-incrementer : It stores the current value of the operand temporarily and only after that statement is completed, the value of the operand is incremented.

Pre-decrementer : It decrements the value of the operand instantly.

Post-decrementer : It stores the current value of the operand temporarily and only after that statement is completed, the value of the operand is decremented.

b. Assignment Operators

Operator	Operation	Example
=	Assigns value of right operand to left operand	A=B will put value of B in A
+=	Adds right operand to the left operand and assigns the result to left operand.	A+=B means A = A+B
-=	Subtracts right operand from the left operand and assigns the result to left operand.	A-=B means A=A-B
*=	Multiplies the right operand with the left operand and assigns the result to the left operand.	A*=B means A=A*B
/=	Divides left operand with the right operand and assigns the result to left operand.	A/=B means A=A/B

c. Comparison/Relational Operators

Relational operators define the relation between 2 entities. They give a boolean value as result i.e true or false.

Suppose : A=5 and B=10

Operator	Operation	Example
==	Gives true if two operands are equal	A==B is not true
!=	Gives true if two operands are not equal	A!=B is true
>	Gives true if left operand is more than right operand	A>B is not true
<	Gives true if left operand is less than right operand	A<B is true
>=	Gives true if left operand is more than right operand or equal to it	A>=B is not true
<=	Gives true if left operand is more than right operand or equal to it	A<=B is true

d. Logical Operators

Logical operators are used to connect multiple expressions or conditions together.
We have 3 basic logical operators.
Suppose : A=0 and B=1

Operator	Operation	Example
&&	AND operator. Gives true if both operands are non zero	(A && B) is false
||	OR operator. Gives true if atleast one of the two operands are non-zero.	(A || B) is true
!	NOT operator. Reverse the logical state of operand	!A is true

11. Math class

Math is an important class in Java that is extensively used and has a lot of interesting functions.
To import - `import java.lang.Math;`

Some functions include:

a. Max

```
int a = 10;
int b = 20;
Math.max(a, b);
```

b. Min

```
int a = 10;
int b = 20;
Math.min(a,b);
```

c. Random

```
int randomNumber = (int)(Math.random()*100);
```

12. Taking Input

We take input using the Scanner class and input various types of data using it.
To import the Scanner class - `import java.util.Scanner;`

Example :

```
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
float a = sc.nextFloat();
String name = sc.next();
String line = sc.nextLine();
```

13. Conditional Statements 'if-else'

The if block is used to specify the code to be executed if the condition specified in if is true, the else block is executed otherwise.

```
int age = 30;
if(age > 18) {
   System.out.println("This is an adult");
} else {
   System.out.println("This is not an adult");
}
```

14. Conditional Statements 'switch'

Switch case statements are a substitute for long if statements that compare a variable to multiple values. After a match is found, it executes the corresponding code of that value case.

```
int n = 1;
switch(n) {
   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;
   default :
       System.out.println("Sunday");
}
```

15. Break & Continue

Jumps in loops are used to control the flow of loops. There are two statements used to implement jump in loops - Continue and Break. These statements are used when we need to change the flow of the loop when some specified condition is met.

Continue statement is used to skip to the next iteration of that loop. This means that it stops one iteration of the loop. All the statements present after the continue statement in that loop are not executed.

```
int i;
for (i=1; i<=20; i++) {

   if (i%3==0) {
       continue;
   }
   System.out.println(i);
}
```

In this for loop, whenever i is a number divisible by 3, it will not be printed as the loop will skip to the next iteration due to the continue statement. Hence, all the numbers except those which are divisible by 3 will be printed.

Break statement is used to terminate the current loop. As soon as the break statement is encountered in a loop, all further iterations of the loop are stopped and control is shifted to the first statement after the end of loop.

```
int i;
for (i=1; i<=20; i++) {

    if (i == 11) {
        break;
    }
    System.out.println(i);
}
```

In this loop, when i becomes equal to 11, the for loop terminates due to break statement, Hence, the program will print numbers from 1 to 10 only.

16. Loops

A loop is used for executing a block of statements repeatedly until a particular condition is satisfied. A loop consists of an initialization statement, a test condition and an increment statement.

For Loop

The syntax of the for loop is :

```
for (initialization; condition; update) {
 // body of-loop
}
```

```
for (int i=1; i<=20; i++) {
    System.out.println(i);
}
```

While Loop

The syntax for while loop is :

```
while(condition) {
 // body of the loop
}
```

```
int i = 0;
while(i<=20) {
    System.out.println(i);
    i++;
}
```

Do-While Loop

The syntax for the do-while loop is :

```
do {
 // body of loop;
}
while (condition);
```

```
int i = 0;
do {
   System.out.println(i);
   i++;
} while(i<=20);
```

17. Exception Handling (try-catch)

Exception Handling in Java is a mechanism to handle the runtime errors so that normal flow of the application can be maintained.

It is done using 2 keywords - ‘try’ and ‘catch’.
Additional keywords like finally, throw and throws can also be used if we dive deep into this concept.

```
int[] marks = {98, 97, 95};
try {
   System.out.println(marks[4]);
} catch (Exception exception) {
   System.out.println("An exception for caught while accessing an index
the 'marks' array");
}
```

```
System.out.println("We tried to print marks & an exception must have
occurred with index >=3");
```

18. Methods/Functions

A function is a block of code that performs a specific task.
Why are functions used?

a. If some functionality is performed at multiple places in software, then rather than writing the same code, again and again, we create a function and call it everywhere. This helps reduce code redundancy.
b. Functions make maintenance of code easy as we have to change at one place if we make future changes to the functionality.
c. Functions make the code more readable and easy to understand.

The **syntax** for function declaration is :

```
return-type function_name (parameter 1, parameter† ..... parameter n){
```

//function_body
}
return-type

The **return type** of a function is the data type of the variable that that function returns.

For eg- If we write a function that adds 2 integers and returns their sum then the return type of this function will be 'int' as we will return a sum that is an integer value.
When a function does not return any value, in that case the return type of the function is 'void'.

function_name
It is the unique name of that function.
It is always recommended to declare a function before it is used.

Parameters
A function can take some parameters as inputs. These parameters are specified along with their data types.
For eg- if we are writing a function to add 2 integers, the parameters would be passed like –
int add (int num1, int num2)

main function
The main function is a special function as the computer starts running the code from the beginning of the main function. Main function serves as the entry point for the program.

Example :

```
package com.apnacollege;

public class Main {
   //A METHOD to calculate sum of 2 numbers - a & b
   public static void sum(int a, int b) {
       int sum = a + b;
       System.out.println(sum);
   }

   public static void main(String[] args) {
      int a = 10;
      int b = 20;
       sum(a, b); // Function Call
```

```
    }
}
```

19. Mini-Project

Let's create a project where we are trying to ask the user to guess a randomly generated number.
The number is in the range of 1 to 100.

If the user guesses a number that is greater, we print "The number is too large".
If the user guesses a number that is smaller, we print "The number is too small".
If the user is able to correctly guess the number, then we print "Correct Number!".

At the end we will print the number that was generated by our Math library.

LET THE GUESSING BEGIN :)

CODE

```
package com.apnacollege;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        //MINI PROJECT
        Scanner sc = new Scanner(System.in);
        int myNumber = (int)(Math.random()*100);
        int userNumber = 0;

        do {
            System.out.println("Guess my number(1-100) : ");
            userNumber = sc.nextInt();

            if(userNumber == myNumber) {
                System.out.println("WOOHOO .. CORRECT NUMBER!!!");
                break;
            }
            else if(userNumber > myNumber) {
                System.out.println("your number is too large");
            }
            else {
                System.out.println("your number is too small");
            }
        } while(userNumber >= 0);

        System.out.print("My number was : ");
        System.out.println(myNumber);
```

```
    }
}
```